Elementos de Teoria da Informação

Mário A. T. Figueiredo

Departamento de Engenharia Electrotécnica e de Computadores
Instituto Superior Técnico
1049-001 Lisboa
Portugal

Versão 1.8

Outubro de 2013

Conteúdo

1 Introdução à Teoria da Informação **5**

1.1 Fontes de Informação sem Memória 5
1.2 Medida de Informação: a Entropia 6
1.3 Propriedades Elementares da Entropia e Quantidades Relacionadas 10
1.3.1 Limites Superior e Inferior para a Entropia 10
1.3.2 Entropia Conjunta 11
1.3.3 Entropia Condicional e Lei de Bayes para Entropias 12
1.3.4 Informação Mútua 14
1.4 Desigualdade da Informação 16
1.5 Corolários da Desigualdade da Informação 18
1.6 A Desigualdade do Processamento de Dados 19

2 Codificação de Fontes Discretas Sem Memória **21**

2.1 Códigos 21
2.1.1 Definições e Notação 21
2.1.2 Códigos Não Singulares 22
2.1.3 Códigos Univocamente Descodificáveis 23
2.1.4 Códigos Instantâneos 24
2.2 Desigualdade de Kraft-McMillan 25
2.3 Códigos Ideais e Códigos Óptimos 28
2.4 Limites para os Códigos Óptimos 30
2.5 Extensões de Fonte 31
2.6 Codificação com Modelo Errado 33
2.7 Codificação de Huffman 34
2.7.1 Algoritmo de Huffman 34
2.7.2 Escrita Recursiva do Algoritmo de Huffman 35
2.7.3 Demonstração de Optimalidade 38
2.7.4 Algoritmo de Huffman para Alfabetos D-ários 41
2.8 Codificação de Shannon-Fano-Elias 41
2.9 Codificação Aritmética 47

3 Fontes Discretas com Memória **51**
3.1 Processos Estocásticos Discretos em Tempo Discreto 51
3.2 Processos Estacionários 52
3.3 Processos de Markov 52
3.3.1 Introdução 52
3.3.2 Processos de Markov Invariantes no Tempo 53
3.3.3 Distribuição dos Estados e Distribuição Estacionária 57
3.4 Taxas de Entropia 59
3.5 Codificação de Fontes com Memória 62

A Demonstração do Teorema da Média de Cesàro **65**

Capítulo 1

Introdução à Teoria da Informação

A teoria da informação dedica-se ao estudo de medidas de *informação* e suas propriedades e aplicações, nomeadamente em problemas de telecomunicações. Não se pode afirmar que existe **a** teoria da informação, mas sim diversas teorias da informação, com fundamentações conceptuais diversas [4]; as mais famosas são a *teoria da informação de Shannon* (TIS) e a *teoria da informação de Kolmogorov* (TIK). A TIS, desenvolvida por Claude Shannon nos anos 40 [10], suporta-se numa perspectiva probabilística, enquanto que a TIK adopta uma perspectiva computacional [8]. Este texto foca exclusivamente a TIS, introduzindo os conceitos teóricos básicos e suas aplicações em problemas de compressão e codificação de dados.

Para além da sua clara importância prática em telecomunicações, a TIS tem influência e aplicabilidade em várias áreas científicas e tecnológicas: biologia (em particular, na biologia molecular [12], na neurobiologia [9], na biologia teórica [2]); física (física estatística, física quântica e cosmologia [11]); química [5]; matemática (por exemplo, teoria das probabilidades e estatística [7], teoria ergódica, sistemas dinâmicos, cálculo combinatório, algebra, optimização); economia (em particular, na análise de estratégias de investimento e no estudo de mercados bolsistas [4]). Este facto reforça a ideia de que um conhecimento básico de teoria da informação deve fazer parte da formação essencial de qualquer engenheiro cuja área de especialidade contemple a manipulação (isto é, aquisição, armazenamento, ou transmissão) de informação, nomeadamente as telecomunicações.

1.1 Fontes de Informação sem Memória

O modelo mais simples para uma fonte de informação sem memória, numa perspectiva probabilística, é simplesmente uma variável aleatória. Por ausência de memória, entende-se a propriedade de que cada símbolo gerado não depende dos símbolos anteriormente gerados. Neste capítulo, apenas se consideram fontes de informação discretas, isto é, que geram símbolos de um alfabeto $\mathcal{X} = \{x_1, ..., x_N\}$. Este alfabeto é perfeitamente abstracto, podendo conter símbolos ASCII (nesse caso, $N = 256$), dígitos binários (com $N = 2$), ou quaisquer outros elementos gerados de forma aleatória. Formalmente, define-se a fonte como uma variável aleatória X que toma valores em $\mathcal{X}$. Dada a ausência de memória, cada símbolo é uma amostra desta variável aleatória, gerada de modo independente das outras amostras.

Uma fonte discreta sem memória é completamente caracterizada pelas probabilidades dos respectivos símbolos, $\{P(X = x_1) = p(x_1), ..., P(X = x_N) = p(x_N)\}$; por vezes, utiliza-se a notação abreviada p_i para representar $p(x_i)$. Dado que são probabilidades, estes números verificam duas propriedades fundamentais:

- $\forall_{i=1,...,N},\ 0 \leq p_i \leq 1$;
- $\sum_{i=1}^{N} p_i = 1$.

Finalmente, interessa recordar que, dada uma função real definida em $\mathcal{X}$, isto é $f : \mathcal{X} \to \mathbb{R}$, o seu valor esperado é dado por

$$E[f(X)] = \sum_{i=1}^{N} p(x_i) f(x_i) = \sum_{i=1}^{N} p_i f(x_i). \tag{1.1}$$

1.2 Medida de Informação: a Entropia

Coloca-se agora a questão de como medir o conteúdo informativo de uma fonte discreta sem memória. Se bem que, de um ponto de vista conceptual, esta questão não é simples, e tem mesmo várias respostas possíveis, este texto aborda a resposta considerada padrão e que está na base da teoria da informação de Shannon. A definição de conteúdo informativo de uma fonte, para a qual se toma como modelo uma variável aleatória, deve depender, naturalmente, das probabilidades dos respectivos símbolos. É consensual que, quanto maior for a incerteza associada a uma fonte, maior é a quantidade de informação que é transmitida a um observador por cada amostra gerada por essa fonte. Assim, a medida de informação procurada pode ser vista como uma uma medida de incerteza. Havendo certamente muitas formas de quantificar incerteza, é necessário restringir a escolha impondo certas propriedades à função em causa. A primeira, e fundamental, é a de que esta medida, que se designará por H, apenas depende das probabilidades dos símbolos da fonte, isto é,

$$H(X) = H(p_1, ..., p_N).$$

Por este motivo, é comum em textos de teoria da informação misturar as duas notações: embora, estritamente, H seja uma função de um conjunto de números (as probabilidades dos símbolos), por vezes escreve-se simplesmente $H(X)$. Consideram-se agora as quatro condições, consideradas naturais, que esta função deve verificar:

- Para duas fontes X e Y, independentes, a incerteza associada ao par (X, Y), que se escreve $H(X, Y)$, deve ser a soma das incertezas, isto é,

$$X \text{ e } Y \text{ independentes } \Rightarrow\ H(X, Y) = H(X) + H(Y). \tag{1.2}$$

  Note-se que o par (X, Y) pode ser visto simplesmente como uma variável aleatória que toma valores no produto cartesiano $\mathcal{X} \times \mathcal{Y}$, em que $\mathcal{X}$ e $\mathcal{Y}$ são os conjuntos (ou alfabetos)

nos quais as variáveis X e Y, respectivamente, tomam valores. Por exemplo, se $\mathcal{X} = \{1,2,3\}$ e $\mathcal{Y} = \{a,b\}$, tem-se $\mathcal{X} \times \mathcal{Y} = \{(1,a),(1,b),(2,a),(2,b),(3,a),(3,b)\}$. Se as variáveis X e Y forem independentes, vem $P[(X,Y) = (1,a)] = P[X = 1] \cdot P[Y = a]$, $P[(X,Y) = (1,b)] = P[X = 1] \cdot P[Y = b]$, etc. Esta factorização escreve-se geralmente (numa notação pouco correcta, mas muito comum) como $p(x,y) = p(x)p(y)$ e constitui, precisamente, a definição de independência.

- Para uma fonte X que gera símbolos equiprováveis, isto é, $p_1 = p_2 = ... = p_N = 1/N$, a incerteza $H(X) = H(1/N, ..., 1/N)$ deve, obviamente, ser uma função monotónica crescente de N. Por outras palavras: "quanto mais símbolos equiprováveis, maior a incerteza".

- A terceira condição é um pouco mais elaborada. Considere-se uma fonte com um alfabeto $\mathcal{X} = \{x_1, ..., x_N\}$, com $N \geq 3$; agrupem-se os símbolos em dois grupos A e B; por exemplo, $A = \{x_1, ..., x_a\}$ e $B = \{x_{a+1}, ..., x_N\}$. A probabilidade da fonte gerar um símbolo do grupo A é, obviamente, $p_A = p_1 + ... + p_a$; a probabilidade de se obter um símbolo do grupo B é $p_B = p_{a+1} + ... + p_N$. A terceira condição a impor à função H é a seguinte:

$$H(p_1, ..., p_N) = H(p_A, p_B) + p_A H\left(\frac{p_1}{p_A}, ..., \frac{p_a}{p_A}\right) + p_B H\left(\frac{p_{a+1}}{p_B}, ..., \frac{p_N}{p_B}\right). \tag{1.3}$$

Por palavras, o que esta condição requer à medida de incerteza/informação é que esta se preserve quando se adopta um esquema hierárquico para comunicar qual dos símbolos foi gerado pela fonte. A quantidade $H(p_A, p_B)$ mede a incerteza associada à escolha entre o grupo A e o grupo B. Note-se que as quantidades[1] $p_1/p_A, ..., p_a/p_A$ são as probabilidades dos símbolos $x_1, ..., x_a$, sob a condição de se saber previamente que o símbolo gerado pertence ao grupo A; de modo similar, $p_{a+1}/p_B, ..., p_N/p_B$ são as probabilidades dos símbolos $x_{a+1}, ..., x_N$, sob a condição de se saber previamente que o símbolo gerado pertence ao grupo B. Assim, $H(p_1/p_A, ..., p_a/p_A)$ é a incerteza associada à geração dos símbolos, sob a condição de que têm de ser símbolos do grupo A. A condição (1.3) é equivalente a exigir que os dois modos seguintes de gerar símbolos de uma fonte de alfabeto $\mathcal{X} = \{x_1, ..., x_N\}$ possuam o mesmo conteúdo informativo (ou incerteza):

Modo 1: Gera-se simplesmente um símbolo, escolhido de acordo com as respectivas probabilidades $p_1, ..., p_N$.

Modo 2: Neste modo, gera-se o símbolo em dois passos; no primeiro passo, escolhe-se um dos grupos, A ou B, de acordo com as respectivas probabilidades p_A e p_B; no segundo passo, gera-se um dos símbolos do grupo escolhido no primeiro passo, de acordo com as respectivas probabilidades condicionadas.

[1] São as probabilidades condicionadas $p(x_i | x_i \in A)$; pela lei de Bayes, sabe-se que estas são dadas por

$$p(x_i | x_i \in A) = \frac{p(x_i, x_i \in A)}{p_A} = \begin{cases} \frac{p(x_i)}{p_A} & \Leftarrow \quad x_i \in A \\ 0 & \Leftarrow \quad x_i \notin A \end{cases}$$

.

- Finalmente, a quarta condição é de natureza técnica, mas perfeitamente natural: a função $H(p_1, ..., p_N)$ deve ser, naturalmente, contínua nos seus argumentos. Por outras palavras, duas fontes cujas probabilidades dos símbolos difiram infinitesimalmente possuem incertezas infinitesimalmente próximas.

É possível provar que a única função que verifica estas quatro condições tem a forma

$$H(p_1, ..., p_N) = -C \sum_{i=1}^{N} p(x_i) \log p(x_i) = C \sum_{i=1}^{N} p_i \log \frac{1}{p_i}, \tag{1.4}$$

em que C é uma constante arbitrária (como tal, assume-se que $C = 1$) e a base do logaritmo é qualquer número real maior que 1 (por uma razão que adiante será tornada clara). Note-se que, como $\log_a x = (\log_b x)/\log_b a$, a adopção de diferentes bases para os logaritmos é equivalente è escolha de diferentes valores para a constante C. A demonstração de que esta função é a única que verifica as quatro condições enumeradas está para além do âmbito deste texto (ver, por exemplo, [1], para a demonstração completa). No entanto, é fácil verificar que H, como definido em (1.4), verifica as condições apresentadas.

- Dada uma fonte X que gera símbolos equiprováveis, isto é, $p_1 = p_2 = ... = p_N = 1/N$, obtém-se

$$H(X) = H(1/N, ..., 1/N) = -\sum_{i=1}^{N} \frac{1}{N} \log \frac{1}{N} = \log N, \tag{1.5}$$

que é, como exigido, uma função monotonicamente crescente de N. Note-se que os logaritmos de base menor que 1 são funções decrescentes, pelo que só podem usar-se (como indicado acima) logaritmos de base maior que 1.

- Sejam X e Y duas variáveis aleatórias independentes, tomando valores, respectivamente, em $\mathcal{X} = \{x_1, ..., x_N\}$ e $\mathcal{Y} = \{y_1, ..., y_M\}$. Dada a independência, tem-se que $\forall_{x \in \mathcal{X}, y \in \mathcal{Y}}$, $p(x, y) = p(x) \cdot p(y)$. Recordando que o par (X, Y) não é mais do que uma variável aleatória que toma valores em $\mathcal{X} \times \mathcal{Y}$, tem-se

$$\begin{aligned}
H(X,Y) &= H(p(x_1)p(y_1),\, p(x_1)p(y_2), ..., p(x_N)p(y_N)) \\
&= -\sum_{i=1}^{N}\sum_{j=1}^{M} p(x_i)p(y_j) \log [p(x_i)p(y_j)] \\
&= -\sum_{i=1}^{N}\sum_{j=1}^{M} p(x_i)p(y_j) [\log p(x_i) + \log p(y_j)] \\
&= \underbrace{-\sum_{i=1}^{N} p(x_i) \log p(x_i) \overbrace{\sum_{j=1}^{M} p(y_j)}^{1}}_{H(X)} \underbrace{-\sum_{j=1}^{N} p(y_j) \log p(y_j) \overbrace{\sum_{i=1}^{N} p(x_i)}^{1}}_{H(Y)} \\
&= H(X) + H(Y),
\end{aligned} \tag{1.6}$$

como exigido pela segunda condição.

- Na verificação da terceira condição, e para manter a notação simples, considere-se o caso particular de um alfabeto com quatro símbolos. Seja $\mathcal{X} = \{1, 2, 3, 4\}$ e considerem-se os sub-conjuntos $A = \{1, 2\}$ e $B = \{3, 4\}$; assim, $p_A = p_1 + p_2$ e $p_B = p_3 + p_4$. Escrevendo o termo da direita da igualdade em (1.3), para este alfabeto $\mathcal{X}$ e esta escolha dos sub-conjuntos A e B, e usando a definição da função H em (1.4),

$$
\begin{aligned}
& H(p_A, p_B) + p_A\, H\left(\frac{p_1}{p_A}, ..., \frac{p_a}{p_A}\right) + p_B\, H\left(\frac{p_{a+1}}{p_B}, ..., \frac{p_N}{p_B}\right) \\
&= H(p_1 + p_2, p_3 + p_4) + (p_1 + p_2)\, H\left(\frac{p_1}{p_1 + p_2}, \frac{p_2}{p_1 + p_2}\right) + (p_3 + p_4)\, H\left(\frac{p_3}{p_3 + p_4}, \frac{p_4}{p_3 + p_4}\right) \\
&= -(p_1 + p_2)\log(p_1 + p_2) - (p_3 + p_4)\log(p_3 + p_4) \\
&\quad -(p_1 + p_2)\left(\frac{p_1}{p_1 + p_2}\log\frac{p_1}{p_1 + p_2} + \frac{p_2}{p_1 + p_2}\log\frac{p_2}{p_1 + p_2}\right) \\
&\quad -(p_3 + p_4)\left(\frac{p_3}{p_3 + p_4}\log\frac{p_3}{p_3 + p_4} + \frac{p_4}{p_3 + p_4}\log\frac{p_4}{p_3 + p_4}\right) \\
&= -(p_1 + p_2)\log(p_1 + p_2) - (p_3 + p_4)\log(p_3 + p_4) \\
&\quad -\left(p_1\log\frac{p_1}{p_1 + p_2} + p_2\log\frac{p_2}{p_1 + p_2}\right) - \left(p_3\log\frac{p_3}{p_3 + p_4} + p_4\log\frac{p_4}{p_3 + p_4}\right) \\
&= -(p_1 + p_2)\log(p_1 + p_2) - (p_3 + p_4)\log(p_3 + p_4) \\
&\quad +(p_1 + p_2)\log(p_1 + p_2) + (p_3 + p_4)\log(p_3 + p_4) \\
&\quad -p_1\log p_1 - p_2\log p_2 - p_3\log p_3 - p_3\log p_3 \\
&= H(p_1, p_2, p_3, p_4), \qquad (1.7)
\end{aligned}
$$

  verificando-se assim a terceira condição.

- Finalmente, dado que o logaritmo é uma função contínua no seu domínio, H é uma função contínua dos seus argumentos.

Após verificar que, de facto, a função H definida em (1.4) verifica as quatro condições impostas, importa acrescentar que, por analogia com uma quantidade formalmente idêntica que surge na física estatística, H é habitualmente designada como **entropia**. Esta função desempenha, como se verá mais adiante, um papel central em toda a teoria da informação. A unidade na qual se expressa a entropia depende da base escolhida para os logaritmos; as escolhas típicas são o logaritmo de base 2, vindo a entropia expressa em *bits/símbolo*, e o logaritmo de base e (ou logaritmo natural, escrito $\log_e$ ou simplesmente ln), vindo neste caso a entropia expressa em *nats/símbolo*. Um valor de entropia expresso em nats/símbolo pode converter-se para bits/símbolo simplesmente multiplicando-o por $\log_2 e$ (ou, equivalentemente, dividindo-o por $\ln 2$). Como a entropia depende apenas das probabilidades dos símbolos, e não dos símbolos, os elementos do alfabeto são, do ponto de vista da teoria da informação, totalmente irrelevantes; apenas as suas probabilidades interessam.

1.3 Propriedades Elementares da Entropia e Quantidades Relacionadas

Apresentam-se, de seguida, algumas das propriedades fundamentais da entropia; introduzem-se outras quantidades fundamentais da teoria da informação (entropia conjunta, entropia condicional, informação mútua) e apresentam-se algumas das suas propriedades. Dada a sua natureza elementar, apresentar-se-ão todas as demonstrações.

1.3.1 Limites Superior e Inferior para a Entropia

A primeira propriedade fundamental da entropia é a sua positividade: para qualquer variável aleatória (fonte) a entropia é não negativa, isto é, $H(X) \geq 0$. A demonstração desta propriedade é elementar; começa por notar-se que a entropia pode ser escrita como o valor esperado de uma função da variável aleatória X,

$$H(X) = H(p_1, ..., p_N) = \sum_{i=1}^{N} p(x_i) \log \frac{1}{p(x_i)} = E[-\log p(X)];$$

dado que, para qualquer símbolo x, se verifica $p(x) \leq 1$, isso implica que $-\log p(x) \geq 0$. O valor esperado de uma função não negativa é, obviamente, não negativo. Pode também apresentar-se uma demonstração directa com base na seguinte cadeia de desigualdades:

$$H(X) = H(p_1, ..., p_N) = \underbrace{-\sum_{i=1}^{N} \underbrace{\underbrace{p_i}_{\geq 0} \underbrace{\log \overbrace{p_i}^{\leq 1}}_{\leq 0}}_{\leq 0}}_{\geq 0}.$$

No tratamento dos símbolos com probabilidade nula ($p_i = 0$), coloca-se a questão acerca de que valor atribuir a $0 \log 0$. Dado que a função logaritmo não está definida em zero, considera-se a extensão por continuidade, usando o limite $\lim_{p \to 0} p \log p$. Embora seja uma indeterminação do tipo $0 \times (-\infty)$, é possível levantar esta indeterminação e verificar que $\lim_{p \to 0} p \log p = 0$. Assim, convenciona-se que sempre que surgir um termo $0 \log 0$, este deve ser entendido como $\lim_{p \to 0} p \log p$ e, como tal, toma o valor zero. Deste modo, os símbolos com probabilidade zero não contribuem para a entropia, tudo se passando como se não existissem no alfabeto.

Em que condições pode a entropia ser zero? Se se observar que cada parcela da soma que define a entropia, $-p_i \log p_i$, é uma quantidade não negativa, conclui-se que a entropia apenas pode ser nula se todas estas parcelas forem zero. Cada uma destas parcelas apenas é zero se a correspondente probabilidade p_i for igual a 0 ou 1. Dado que a soma de todas as probabilidades é igual a 1, apenas uma das probabilidades pode ser igual a 1, concluindo-se que a entropia é nula se um dos símbolos tiver probabilidade 1 e todos os outros probabilidade 0. Este conclusão está de acordo com a interpretação da entropia como medida de incerteza pois, se um símbolo possui probabilidade 1 de ocorrer, a incerteza é claramente inexistente.

A segunda propriedade fundamental da entropia afirma que, para uma fonte X com um alfabeto de N símbolos, $H(X) \leq \log N$. A demonstração desta propriedade será apresentada

mais tarde, dado que se baseia na desigualdade da informação, a qual será enunciada e demonstrada adiante. No entanto, pode desde já verificar-se que esta propriedade está também de acordo com a interpretação da entropia como medida de incerteza: a incerteza (imprevisibilidade) máxima atinge-se na situação em que todos os símbolos são equiprováveis; recorde-se que se mostrou em (1.5) que, ao caso $p_i = 1/N$, para $i = 1, ..., N$, corresponde $H = \log N$.

Considere-se uma fonte binária, ou seja, cujo alfabeto possui apenas dois símbolos, por simplicidade designados simplesmente como 1 e 0. Designando-se como p a probabilidade do símbolo 1, tem-se imediatamente que a probabilidade do símbolo 0 é igual a $1 - p$. A entropia da fonte binária é então dada por

$$H(X) = H(p, 1-p) = -p \log p - (1-p) \log(1-p),$$

que se representa graficamente na figura 1.1, como função do valor de p.

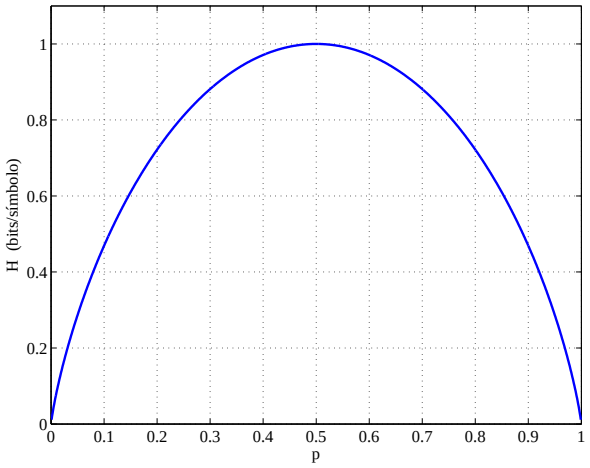

Figura 1.1: Entropia de uma fonte binária sem memória em função da probabilidade de um dos símbolos.

1.3.2 Entropia Conjunta

A *entropia conjunta* de um par de variáveis aleatórias foi já implicitamente usada em (1.2) para o caso de duas variáveis independentes. Para introduzir formalmente o conceito de entropia conjunta, considere-se um par de variáveis aleatórias X e Y, tomando valores nos alfabetos $\mathcal{X} = \{x_1, ..., x_N\}$ e $\mathcal{Y} = \{y_1, ..., y_M\}$. Este par de variáveis aleatórias é caracterizado pelas probabilidades conjuntas $\{p(x, y),\ x \in \mathcal{X},\ y \in \mathcal{Y}\}$. Obviamente, verifica-se que $0 \leq p(x, y) \leq 1$, para qualquer par $(x, y) \in \mathcal{X} \times \mathcal{Y}$, bem como

$$\sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x, y) = 1.$$

A entropia conjunta de X e Y, designada $H(X, Y)$ não é mais do que a entropia da variável aleatória constituída pelo par (X, Y); isto é,

$$H(X, Y) = -\sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x, y) \log p(x, y). \tag{1.8}$$

Sendo uma entropia, $H(X,Y)$ verifica as duas desigualdades acima enunciadas: $0 \leq H(X,Y) \leq \log(MN) = \log N + \log M$. Recorde-se que, dado que X e Y podem tomar N e M valores diferentes, respectivamente, o par (X,Y) pode tomar NM valores diferentes (que é o cardinal do produto cartesiano $\mathcal{X} \times \mathcal{Y}$). Como demonstrado em (1.2), se X e Y forem variáveis aleatórias independentes (isto é, se $p(x,y) = p(x)p(y)$, para qualquer par (x,y)), a entropia conjunta é igual à soma das entropias: $H(X,Y) = H(X) + H(Y)$.

A definição de entropia conjunta pode extender-se a um conjunto arbitrário de variáveis aleatórias. Considere-se um conjunto de L variáveis aleatórias $X_1, ..., X_L$, tomando valores nos conjuntos (ou alfabetos) $\mathcal{X}_1, ..., \mathcal{X}_L$. Este conjunto de variáveis aleatórias é caracterizado pelas probabilidades conjuntas $\{p(x_1, ..., x_L),\ x_1 \in \mathcal{X}_1, ..., x_L \in \mathcal{X}_L\}$. A entropia conjunta define-se, naturalmente, como

$$H(X_1, ..., X_L) = -\sum_{x_1 \in \mathcal{X}_1} \sum_{x_2 \in \mathcal{X}_2} \cdots \sum_{x_L \in \mathcal{X}_L} p(x_1, x_2, ..., x_L) \log p(x_1, x_2, ..., x_L). \tag{1.9}$$

Obviamente, as desigualdades acima apresentadas mantêm-se válidas,

$$0 \leq H(X_1, X_2 ..., X_L) \leq \log(|\mathcal{X}_1| \cdot |\mathcal{X}_2| \cdots |\mathcal{X}_L|) = \sum_{l=1}^{L} \log(|\mathcal{X}_l|),$$

onde $|\mathcal{X}|$ designa o cardinal do conjunto $\mathcal{X}$.

1.3.3 Entropia Condicional e Lei de Bayes para Entropias

A entropia de uma variável aleatória X, condicionada pela presença (ou conhecimento) de uma outra variável Y, mede a incerteza de X quando Y é conhecida. Se se condicionar a um valor específico $Y = y$, as probabilidades condicionais $\{p(x|y),\ x \in \mathcal{X}\}$ podem ser usadas na definição original de entropia pois verificam $0 \leq p(x|y) \leq 1$ e

$$\sum_{x \in \mathcal{X}} p(x|y) = 1,$$

qualquer que seja $y \in \mathcal{Y}$. Surge assim a entropia/incerteza de X, condicionada a que $Y = y$, dada por

$$H(X|Y = y) = -\sum_{x \in \mathcal{X}} p(x|y) \log p(x|y).$$

Para medir a entropia/incerteza de X, na presença de Y, quando esta toma todos os seus possíveis valores com as respectivas probabilidades[2] $p(y)$, é necessário tomar o valor esperado de $H(X|Y = y)$; surge assim a definição de *entropia condicional*:

$$\begin{aligned} H(X|Y) &= \sum_{y \in \mathcal{Y}} p(y)\, H(X|Y = y) && (1.10) \\ &= -\sum_{y \in \mathcal{Y}} p(y) \sum_{x \in \mathcal{X}} p(x|y) \log p(x|y) \\ &= -\sum_{y \in \mathcal{Y}} \sum_{x \in \mathcal{X}} p(x,y) \log p(x|y), && (1.11) \end{aligned}$$

[2]Recorde-se que os valores de $p(y)$ podem ser obtidos das probabilidades conjuntas por marginalização: $p(y) = \sum_{x \in \mathcal{X}} p(x,y)$.

onde se usou o facto de que, de acordo com a lei de Bayes, $p(x|y)p(y) = p(x, y)$.

Na teoria das probabilidades, a lei de Bayes estabelece a relação entre probabilidades conjuntas, condicionais e marginais. Esta lei reflecte-se na teoria da informação dando origem à chamada *lei de Bayes para entropias*:

$$H(X, Y) = H(X|Y) + H(Y). \tag{1.12}$$

A demonstração desta igualdade é simples:

$$\begin{aligned}
H(X|Y) + H(Y) &= -\sum_{y\in\mathcal{Y}}\sum_{x\in\mathcal{X}} p(x, y) \log p(x|y) - \sum_{y\in\mathcal{Y}} p(y) \log p(y) && (1.13)\\
&= -\sum_{y\in\mathcal{Y}}\sum_{x\in\mathcal{X}} p(x, y) \log p(x|y) - \sum_{y\in\mathcal{Y}}\sum_{x\in\mathcal{X}} p(x, y) \log p(y) && (1.14)\\
&= -\sum_{y\in\mathcal{Y}}\sum_{x\in\mathcal{X}} p(x, y) \left[\log p(x|y) + \log p(y)\right] \\
&= -\sum_{y\in\mathcal{Y}}\sum_{x\in\mathcal{X}} p(x, y) \log \left[p(x|y)p(y)\right] && (1.15)\\
&= -\sum_{y\in\mathcal{Y}}\sum_{x\in\mathcal{X}} p(x, y) \log p(x, y) && (1.16)\\
&= H(X, Y).
\end{aligned}$$

Para passar da expressão (1.13) para (1.14) usou-se a definição de probabilidade marginal, $p(y) = \sum_{x\in\mathcal{X}} p(x, y)$; para passar de (1.15) para (1.16), invocou-se a lei de Bayes, ou seja $p(x|y)p(y) = p(x, y)$. Dado que se pode repetir a demonstração, trocando os papeis de X e Y, pode escrever-se também

$$H(X, Y) = H(X|Y) + H(Y) = H(Y|X) + H(X). \tag{1.17}$$

Calcule-se agora a entropia condicional para duas variáveis aleatórias, X e Y, independentes. Neste caso, já se sabe que $H(X, Y) = H(X) + H(Y)$; assim,

$$H(X|Y) = H(X, Y) - H(Y) = H(X) + H(Y) - H(Y) = H(X), \tag{1.18}$$

ou seja, se X e Y forem independentes, as entropias condicional e não condicional são iguais. Esta conclusão é bastante natural pois, se as variáveis são independentes, o conhecimento de uma não altera a incerteza acerca da outra. Naturalmente, do mesmo modo, pode-se escrever que, se X e Y forem independentes, $H(Y|X) = H(Y)$.

No extremo oposto estão os pares de variáveis nos quais uma é uma função determinística da outra, ou seja, para as quais se pode escrever $X = f(Y)$, em que $f : \mathcal{Y} \to \mathcal{X}$ é uma função arbitrária (determinística). Neste caso, pode afirmar-se que, para qualquer $y \in \mathcal{Y}$, se verifica que

$$H(X|Y = y) = 0,$$

pois o valor $x = f(y)$ apresenta probabilidade condicionada igual a um, enquanto todos os outros valores possuem probabilidade condicionada nula. Inserindo esta igualdade na definição

de entropia condicional (1.10) surge $H(X|Y) = 0$. Esta conclusão é bastante natural: se X for uma função determinística de Y, o conhecimento de Y retira toda a incerteza a X.

É importante notar que $H(X|Y) = 0$ não implica que $H(Y|X) = 0$. Considera-se, de seguida, um exemplo ilustrativo.

Exemplo 1.1 *Seja Y uma variável aleatória que toma valores em $\mathcal{Y} = \{a, b, c\}$ e X uma variável aleatória com valores em $\mathcal{X} = \{1, 2\}$ e que é uma função determinística de Y definida do seguinte modo: $f(a) = 1$, $f(b) = 1$, $f(c) = 2$. Claramente, $H(X|Y = a) = 0$, $H(X|Y = b) = 0$, $H(X|Y = c) = 0$, pelo que $H(X|Y) = 0$. No entanto, $H(Y|X = 1) \neq 0$, pois o facto de se saber que $X = 1$ não chega para se saber o valor de Y (pode ser a ou b). Assim, neste caso, embora $H(X|Y) = 0$, verifica-se que $H(Y|X) \neq 0$.*

A lei de Bayes para entropias é bastante útil na obtenção e manipulação de entropias condicionais. Por exemplo, dadas três variáveis aleatórias X_1, X_2 e X_3, podem definir-se todas as possíveis entropias condicionais simplesmente à custa de entropias marginais e conjuntas; por exemplo,

$$\begin{aligned} H(X_1|X_2, X_3) &= H(X_1, X_2, X_3) - H(X_2, X_3) \\ H(X_1, X_3|X_2) &= H(X_1, X_2, X_3) - H(X_2). \end{aligned}$$

Este tipo de igualdades pode também ser usado para decompor entropias conjuntas em somas de entropias condicionais; por exemplo,

$$\begin{aligned} H(X_1, X_2, X_3) &= H(X_1|X_2, X_3) + H(X_2, X_3) \\ &= H(X_1|X_2, X_3) + H(X_2|X_3) + H(X_3). \end{aligned} \tag{1.19}$$

É claro que os mesmos factos podem ser invocados para obter uma decomposição por ordem inversa: $H(X_1, X_2, X_3) = H(X_3|X_2, X_1) + H(X_2|X_1) + H(X_1)$. Este tipo de igualdades pode extender-se em cadeia, dando origem às chamadas *regras de cadeia* ("*chain rules*"). Considere-se um conjunto de L variáveis aleatórias $X_1, ..., X_L$, tomando valores nos conjuntos (ou alfabetos) $\mathcal{X}_1, ..., \mathcal{X}_L$. Recorrendo à lei de Bayes para entropias, pode escrever-se:

$$\begin{aligned} H(X_1, ..., X_L) &= H(X_L|X_{L-1}, ..., X_1) + H(X_{L-1}|X_{L-2}, ..., X_1) + ... + H(X_2|X_1) + H(X_1) \\ &= H(X_1) + \sum_{l=2}^{L} H(X_l|X_{l-1}, ..., X_1). \end{aligned} \tag{1.20}$$

1.3.4 Informação Mútua

A igualdade expressa em (1.17), $H(X|Y) + H(Y) = H(Y|X) + H(X)$, sugere que se considere uma outra quantidade obtida por permutação das parcelas $H(X|Y)$ e $H(Y|X)$ para os membros opostos da igualdade. Daí, surge a igualdade

$$H(Y) - H(Y|X) = H(X) - H(X|Y) \equiv I(X; Y) \tag{1.21}$$

a qual define uma quantidade à qual (por motivos que adiante se tornarão claros) se dá o nome de *informação mútua* e que se escreve $I(X;Y)$. A partir desta definição de $I(X;Y)$ pode facilmente chegar-se a uma expressão directa a partir das probabilidades:

$$\begin{aligned}
I(X;Y) &= H(X) - H(X|Y) \\
&= -\sum_{x\in\mathcal{X}} p(x)\log p(x) + \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\log p(x|y) && (1.22)\\
&= -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\log p(x) + \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\log p(x|y) && (1.23)\\
&= \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\left[\log p(x|y) - \log p(x)\right] \\
&= \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\left[\log \frac{p(x|y)}{p(x)}\right] && (1.24)\\
&= \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\left[\log \frac{p(x,y)}{p(x)p(y)}\right] && (1.25)
\end{aligned}$$

Na passagem de (1.22) para (1.23) usou-se de novo a igualdade $p(x) = \sum_{y\in\mathcal{Y}} p(x,y)$; para obter (1.25) a partir de (1.24), invocou-se a lei de Bayes sob a forma $p(x|y) = p(x,y)/p(y)$.

Usando a lei de Bayes para entropias $H(X|Y) = H(X,Y) - H(Y)$ pode calcular-se a informação mútua sem usar explicitamente entropias condicionais:

$$I(X;Y) = H(X) - H(X|Y) = H(X) + H(Y) - H(X,Y).$$

Esta igualdade sugere que a informação mútua pode ser vista como uma medida de dependência entre variáveis aleatórias, pois quanto "mais independentes" forem X e Y, menor será a diferença entre $H(X)+H(Y)$ e $H(X,Y)$. Esta afirmação será confirmada mais adiante e apresentada mais formalmente.

Uma das propriedades básicas da informação mútua pode obter-se directamente da igualdade $I(X;Y) = H(X)+H(Y)-H(X|Y)$. Se as variáveis aleatórias X e Y forem independentes, tem-se $H(X,Y) = H(X) + H(Y)$ e, como tal,

$$I(X;Y) = H(X) + H(Y) - H(X,Y) = 0;$$

ou seja, a informação mútua entre variáveis aleatórias independentes é nula. Esta observação reforça a sugestão de que a informação mútua pode ser vista como uma medida de dependência entre variáveis aleatórias.

Considere-se agora o caso em que uma variável é uma função determinística da outra, ou seja, pode escrever-se $X = f(Y)$, em que $f : \mathcal{Y} \to \mathcal{X}$ é uma função determinística. Neste caso, como mostrado acima, $H(X|Y) = 0$ e a informação mútua fica $I(X;Y) = H(X) - H(X|Y) = H(X)$. Ou seja, neste caso a informação mútua é igual à entropia da variável cuja entropia condicional é nula.

Finalmente refira-se que a informação mútua, por ser igual a uma diferença entre entropias, se expressa nas mesmas unidades que essas entropias; por exemplo, em bits/símbolo ou nats/símbolo.

1.4 Desigualdade da Informação

Uma dos resultados centrais da teoria da informação é a *desigualdade da informação*, a qual estabelece a não negatividade da informação mútua. Esta desigualdade, a qual conduz a alguns importantes corolários, suporta-se na interpretação da informação mútua como uma *divergência de Kullbak-Leibler*, que de seguida se introduz formalmente.

Considere-se um alfabeto $\mathcal{X} = \{x_1, ..., x_N\}$ e duas funções de probabilidade[3] definidas sobre este alfabeto: $p_1 : \mathcal{X} \to \mathbb{R}$ e $p_2 : \mathcal{X} \to \mathbb{R}$. A *divergência de Kullbak-Leibler* (DKL) entre p_1 e p_2 é uma medida de dissemelhança entre p_1 e p_2 que se define como

$$D_{\mathrm{KL}}(p_1 \| p_2) = \sum_{x \in \mathcal{X}} p_1(x) \log \frac{p_1(x)}{p_2(x)}. \tag{1.26}$$

A divergência de Kullback-Leibler é claramente não simétrica, em geral $D(p_1\|p_2) \neq D(p_2\|p_1)$, pelo que não pode ser considerada uma distância entre funções de probabilidade. A propriedade fundamental da DKL é expressa pela desigualdade da informação:

Desigualdade da informação: Para qualquer par de funções de probabilidade p_1 e p_2 definidas sobre o mesmo alfabeto $\mathcal{X}$, verifica-se

$$D_{\mathrm{KL}}(p_1 \| p_2) \geq 0,$$

com igualdade se e só se $p_1(x) = p_2(x)$, para todos os $x \in \mathcal{X}$.

Demonstração: A demonstração desta desigualdade suporta-se, de modo simples, no facto de a função logaritmo ser côncava. Recorde-se que uma função real de variável real é dita côncava se possuir a seguinte propriedade: sejam a e b dois pontos no domínio de f; então, para qualquer $\lambda \in [0, 1]$, tem-se $f((1-\lambda)a + \lambda b) \geq (1-\lambda)f(a) + \lambda f(b)$. Quando a desigualdade se verifica estritamente, diz-se que a função é estritamente côncava. Uma função duas vezes diferenciável (isto é, que possui segunda derivada em toda a parte) é côncava se e só se a sua segunda derivada for negativa; é imediato verificar que a função logaritmo natural verifica esta propriedade:

$$\frac{d^2 \log x}{d^2 x} = -\frac{1}{x^2} < 0.$$

Uma função côncava, duas vezes diferenciável, é menor ou igual a qualquer das suas tangentes; este facto pode facilmente demonstrar-se considerando o desenvolvimento em série de Taylor da função f em torno de um ponto x_0:

$$f(x) = f(x_0) + (x - x_0) \left.\frac{df(x)}{dx}\right|_{x_0} + \underbrace{\underbrace{\frac{1}{2}(x - x_0)^2}_{\geq 0} \underbrace{\left.\frac{d^2 f(x)}{d^2 x}\right|_{x_1}}_{\leq 0}}_{\leq 0} \leq \underbrace{f(x_0) + (x - x_0) \left.\frac{df(x)}{dx}\right|_{x_0}}_{\text{tangente a } f \text{ em } x_0},$$

[3]Note-se que uma distribuição de probabilidades definida sobre um alfabeto/conjunto pode ser vista como uma função real $p : \mathcal{X} \to \mathbb{R}$ verificando duas restrições: $\forall_{x \in \mathcal{X}},\ 0 \leq p(x) \leq 1$ e $\sum_x p(x) = 1$.

em que x_1 é um ponto entre x_0 e x. Concretizando esta desigualdade para a função logaritmo natural, com $x_0 = 1$, tem-se

$$\ln(x) \leq x - 1, \tag{1.27}$$

com igualdade se e só se $x = 1$, como ilustrado na figura 1.2.

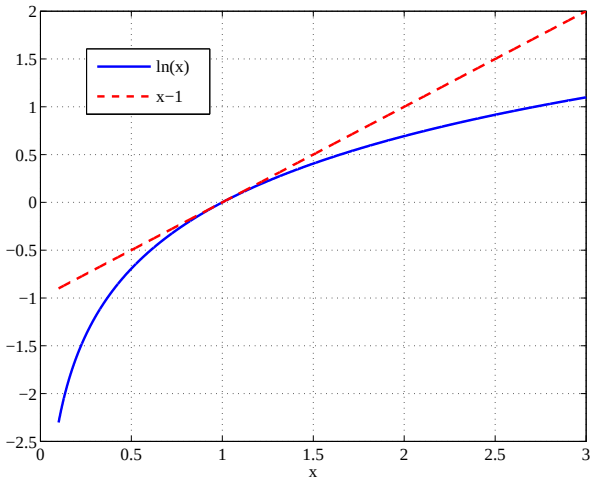

Figura 1.2: A função logaritmo natural é menor do que a sua tangente em $x = 1$, a função $x - 1$.

Armados com a desigualdade anterior, pode agora passar-se à demonstração da desigualdade da informação. Considere-se que os logaritmos usados são numa qualquer base $c > 1$. Seja A o conjunto dos símbolos para os quais p_1 é estritamente positiva: $A = \{x : \ p_1(x) > 0\}$. Para demonstrar que $D_{\mathrm{KL}}(p_1 \| p_2) \geq 0$, demonstra-se a desigualdade equivalente $-D_{\mathrm{KL}}(p_1 \| p_2) \leq 0$:

$$\begin{aligned}
-D_{\mathrm{KL}}(p_1\|p_2) &= -\frac{1}{\ln c}\sum_{x\in\mathcal{X}} p_1(x)\ln\frac{p_1(x)}{p_2(x)} \\
&= \frac{1}{\ln c}\sum_{x\in\mathcal{X}} p_1(x)\ln\frac{p_2(x)}{p_1(x)} && (1.28)\\
&= \frac{1}{\ln c}\sum_{x\in A} p_1(x)\ln\frac{p_2(x)}{p_1(x)} && (1.29)\\
&\leq \frac{1}{\ln c}\sum_{x\in A} p_1(x)\left(\frac{p_2(x)}{p_1(x)} - 1\right) && (1.30)\\
&= \frac{1}{\ln c}\underbrace{\sum_{x\in A} p_2(x)}_{\leq 1} - \frac{1}{\ln c}\underbrace{\sum_{x\in A} p_1(x)}_{=1} \ \leq\ 0. && (1.31)
\end{aligned}$$

A igualdade entre (1.28) e (1.29) justifica-se pelo facto de que os termos com $p_1(x) = 0$ têm uma contribuição nula para o somatório. A passagem de (1.29) para (1.30) usa a desigualdade (1.27).

Finalmente, para demonstrar que $D_{\mathrm{KL}}(p_1\|p_2) = 0$ se e só se $p_1(x) = p_2(x)$, para todos os $x \in \mathcal{X}$, observem-se as desigualdades contidas nas expressões (1.30)-(1.31):

- A desigualdade $\sum_{x\in A} p_2(x) \leq 1$, invocada em (1.31), verifica-se com igualdade se e só se $\{x :\ p_2(x) > 0\} = A = \{x :\ p_1(x) > 0\}$, isto é, se os elementos para os quais p_2 é estritamente positiva são os mesmos para os quais p_1 é estritamente positiva.
- As desigualdades
$$\ln \frac{p_2(x)}{p_1(x)} \leq \frac{p_2(x)}{p_1(x)} - 1,$$
para todos os $x \in A$, são igualdades se e só se $p_2(x)/p_1(x) = 1$, para todos os $x \in A$.

A conjunção destas duas condições implica que $D_{\mathrm{KL}}(p_1\|p_2) = 0$ se e só se $p_1(x) = p_2(x)$, para todos os $x \in \mathcal{X}$.

Finalmente, resta mostrar que a informação mútua é, de facto, uma divergência de Kullback-Leibler. A observação da expressão (1.25) mostra que, de facto,

$$I(X;Y) = D_{\mathrm{KL}}(p_1\|p_2) = \sum_{x\in\mathcal{X}} \sum_{y\in\mathcal{Y}} p_1(x,y) \log \frac{p_1(x,y)}{p_2(x,y)}$$

em que $p_1(x,y) = p(x,y)$ e $p_2(x,y) = p(x)p(y)$. Ou seja, a informação mútua entre duas variáveis aleatórias é igual à divergência de Kullback-Leibler entre a sua função de probabilidade conjunta e uma outra função de probabilidade, sob a qual as variáveis aleatórias são vistas como independentes. Esta conclusão reforça a interpretação da informação mútua como uma medida de dependência entre variáveis aleatórias. Por este facto, a desigualdade da informação pode também escrever-se como

$$I(X;Y) \geq 0,$$

com igualdade se e só se X e Y forem independentes (pois nesse caso $p_1(x,y) = p(x,y) = p_2(x,y) = p(x)p(y)$).

1.5 Corolários da Desigualdade da Informação

Apresentam-se agora alguns corolários imediatos da desigualdade da informação:

- A desigualdade $H(X) \leq \log N$ (enunciada sem demonstração na subsecção 1.3.1) obtém-se do seguinte modo. Considere-se $p_2(x) = 1/N$, para todos os $x \in \mathcal{X} = \{x_1, ..., x_N\}$. Então,

$$\begin{aligned} 0 &\geq -D_{\mathrm{KL}}(p\|p_2) \\ &= \sum_{x\in\mathcal{X}} p(x) \log \frac{1/N}{p(x)} \\ &= -\sum_{x\in\mathcal{X}} p(x) \log p(x) - \sum_{x\in\mathcal{X}} p(x) \log N \\ &= H(X) - \log N. \end{aligned}$$

- O segundo corolário é $H(X|Y) \leq H(X)$, com igualdade se e só se X e Y forem independentes. A demonstração é trivial, recordando a definição de informação mútua em (1.21):

$$0 \leq I(X:Y) = H(X) - H(X|Y).$$

  Esta desigualdade afirma que a incerteza de uma variável aleatória X não pode aumentar pela presença de uma segunda variável aleatória Y; pode apenas manter-se inalterada ou diminuir.

- Finalmente, o terceiro corolário afirma que a entropia conjunta atinge o seu valor máximo na situação de independência. Partindo da regra de cadeia (1.20) e invocando o corolário anterior, segundo o qual $H(X_l|X_{l-1},...,X_1) \leq H(X_l)$,

$$\begin{aligned} H(X_1,...,X_L) &= H(X_1) + \sum_{l=2}^{L} H(X_l|X_{l-1},...,X_1) \\ &\leq \sum_{l=1}^{L} H(X_l), \end{aligned} \tag{1.32}$$

  com igualdade se e só se as variáveis aleatórias X_1,...,X_L forem independentes.

1.6 A Desigualdade do Processamento de Dados

Considerem-se três variáveis aleatórias X, Y e Z, com valores em $\mathcal{X}$, $\mathcal{Y}$ e $\mathcal{Z}$, respectivamente. Diz-se que estas três variáveis formam uma "cadeia de Markov" se e só se verificarem

$$p(Z=z|X=x,Y=y) = p(Z=z|Y=y), \;\; \forall_{x\in\mathcal{X},\, y\in\mathcal{Y},\, z\in\mathcal{Z}}.$$

Esta igualdade afirma que Z apenas depende X através de Y, usando-se por vezes a notação $X \to Y \to Z$. Uma condição equivalente é que quando Y é observada, as variáveis X e Z são independentes; isto pode demonstrar-se simplesmente invocando a lei de Bayes (usando a notação abreviada $p(x) \equiv p(X=x)$):

$$p(x,z|y) = \frac{p(x,y,z)}{p(y)} = \frac{p(z|x,y)\,p(x,y)}{p(y)} = \frac{p(z|y)\,p(x|y)p(y)}{p(y)} = p(z|y)\,p(x|y),$$

desde que $p(y) > 0$, para todos os $y \in \mathcal{Y}$. A igualdade $p(x,z|y) = p(z|y)\,p(x|y)$ estabelece que X e Z são condicionalmente independentes, na presença de Y.

Obviamente, dadas duas variáveis aleatórias X e Y, se uma terceira variável aleatória Z for uma função determinística de uma delas, $Z = f(Y)$, verifica-se que $X \to Y \to Z$.

A chamada "desigualdade do processamento de dados" (DPD) afirma: se $X \to Y \to Z$, então $I(X;Y) \geq I(X;Z)$. Isto é, se Z apenas depende de X através de Y, então, Z possui menos informação acerca de X do que Y. Por outras palavras, qualquer que seja o "processamento" que se aplique a Y, sendo o resultado Z, o resultado deste processamento não pode ter mais informação acerca de X do que o próprio Y. A importante lição é: "nenhum

tipo de processamento aplicado a um conjunto de dados pode aumentar o conteúdo informativo desse conjunto de dados".

A demonstração da DPD é simples e resulta da desigualdade da informação que foi enunciada e demonstrada na secção 1.4. Considerem-se as informação mútua $I(X;Y,Z)$ entre a variável X e o par de variáveis (Y,Z); informação mútua é dada, por definição, por

$$\begin{aligned} I(X;Y,Z) &= H(Y,Z) - H(Y,Z|X) && (1.33)\\ &= H(Y|Z) + H(Z) - [H(Y|Z,X) + H(Z|X)] && (1.34)\\ &= \underbrace{H(Y|Z) - H(Y|Z,X)}_{I(X;Y|Z)} + \underbrace{H(Z) - H(Z|X)}_{I(X;Z)}; && (1.35) \end{aligned}$$

a igualdade entre (1.33) e (1.34) resulta da lei de Bayes para entropias $H(Y,Z) = H(Y|Z) + H(Z)$, a qual, naturalmente, também é válida para entropias condicionais, $H(Y,Z|X) = H(Y|Z,X) + H(Z|X)$. A quantidade $I(X;Y|Z) = H(Y|Z) - H(Y|Z,X)$ designa-se, naturalmente, como informação mútua condicional e tem uma definição idêntica à informação mútua não condicional, mas envolvendo entropias condicionais. Dado que é possível repetir a sequência (1.33) - (1.35) trocando Y com Z, pode escrever-se

$$I(X;Y,Z) = I(X;Z) + I(X;Y|Z) = I(X;Y) + I(X;Z|Y). \quad (1.36)$$

Uma vez que, condicionadas a Y, as variáveis X e Z são independentes, tem-se $I(X;Z|Y) = 0$; por outro lado, por ser uma informação mútua, $I(X;Y|Z) \geq 0$. Introduzindo estes dois factos em (1.36) resulta imediatamente que $I(X;Z) \leq I(X;Y)$, como se pretendia demonstrar.

Capítulo 2

Codificação de Fontes Discretas Sem Memória

Um dos papeis fundamentais das grandezas e propriedades estudadas no Capítulo 1 consiste no estabelecimento de limites teóricos para a codificação de informação. Neste capítulo, estudam-se esses limites bem como técnicas concretas que os aproximam (e, sob certas circunstâncias, os atingem).

2.1 Códigos

2.1.1 Definições e Notação

A formalização do conceito de código é necessária ao seu estudo à luz da teoria da informação. Informalmente, um código é uma forma de representar os símbolos de uma dada fonte; para o efeito, atribui-se a cada símbolo gerado pela fonte uma sequência de símbolos do alfabeto sobre o qual está definido o código. Formalmente, considere-se uma fonte sem memória, discreta, que gera símbolos de um alfabeto $\mathcal{X} = \{x_1, ..., x_N\}$. Um *codificador*, ou simplesmente um *código*, definido sobre o alfabeto $\mathcal{D}$, é uma função

$$C : \mathcal{X} \to \mathcal{D}^*,$$

onde $\mathcal{D}^*$ (chamado *fecho de Kleene – Kleene closure*) denota o conjunto de todas as sequências finitas de símbolos de $\mathcal{D}$, incluindo a sequência vazia, designada como ε. Recorde-se que $\mathcal{D}^k$ representa a k-ésima a potência cartesiana do conjunto $\mathcal{D}$, isto é, o conjunto de todas as sequências de k elementos de $\mathcal{D}$. No caso binário, $\mathcal{D} = \{0, 1\}$, com $k = 3$, tem-se $\mathcal{D}^3 = \{000, 001, 010, 011, 100, 101, 110, 111\}$. A notação $\mathcal{D}^*$ representa o conjunto (infinito) de todas as sequências finitas de elementos de $\mathcal{D}$. Por exemplo, no caso binário, $\mathcal{D} = \{0, 1\}$,

$$\mathcal{D}^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, ..., 111, 0000, 0001, 0010, ...\}.$$

Assim, um código atribui a cada símbolo do alfabeto da fonte, $\mathcal{X}$, uma sequência finita de símbolos de $\mathcal{D}$.

Exemplo 2.1 *Considere-se o alfabeto de fonte* $\mathcal{X} = \{a, b, c, d, e\}$ *e o alfabeto de código* $\mathcal{D} = \{0, 1, 2\}$*; um exemplo de código para* $\mathcal{X}$*, definido sobre* $\mathcal{D}$ *é*

$$C(a) = 0, \quad C(b) = 10, \quad C(c) = 22, \quad C(d) = 2, \quad C(e) = 210. \tag{2.1}$$

Sem qualquer perda de generalidade (dado que os símbolos do alfabeto do código são totalmente arbitrários e abstractos), considera-se-á sempre que $\mathcal{D} = \{0, 1, ..., D-1\}$. Um código definido sobre um alfabeto com D símbolos diz-se um código D-ário. Na quase totalidade dos casos, estudam-se códigos binários, isto é, toma-se $\mathcal{D} = \{0, 1\}$, embora todos os resultados apresentados neste capítulo se possam generalizar sem dificuldade para o caso de códigos D-ários, com qualquer $D \geq 2$.

Denota-se como $l_C(x)$, para $x \in \mathcal{X}$, o comprimento (em número de símbolos de $\mathcal{D}$) de $C(x)$. Isto é, pode ver-se $l_C : \mathcal{X} \to \mathbb{N} = \{1, 2, ...\}$ como uma função que atribui um número natural a cada símbolo do alfabeto da fonte. Considerando que a fonte é caracterizada por uma função de probabilidade $p(x)$, o valor esperado do comprimento do código C, designado como $L(C)$, é dado por

$$L(C) = E[l_C(X)] = \sum_{x \in \mathcal{X}} p(x)\, l_C(x), \tag{2.2}$$

vulgarmente designado como comprimento médio. A unidade na qual se exprime o comprimento médio é, naturalmente, "símbolos de $\mathcal{D}$ por símbolo de $\mathcal{X}$". No caso binário, com $\mathcal{D} = \{0, 1\}$, os comprimentos médios exprimem-se em bits/símbolo, tal como as entropias de base 2.

Exemplo 2.2 *Para o código definido no Exemplo 2.1, vem*

$$l_C(a) = 1, \quad l_C(b) = 2, \quad l_C(c) = 2, \quad l_C(d) = 1, \quad l_C(e) = 3.$$

O comprimento médio correspondente, assumindo que as probabilidades dos símbolos são $\{p(a) = 0.4,\ p(b) = 0.2,\ p(c) = 0.15,\ p(d) = 0.15,\ p(e) = 0.1\}$*, é*

$$0.4 \times 1 + 0.2 \times 2 + 0.15 \times 2 + 0.15 \times 1 + 0.1 \times 3 = 1.55,$$

que se exprime em unidades "símbolos de $\mathcal{D}$ *por símbolo de* $\mathcal{X}$*".*

2.1.2 Códigos Não Singulares

Apresentam-se de seguida várias condições a impor aos códigos por forma a serem utilizáveis. A primeira, e mais fraca, condição a impor a um código é que este seja "não singular"; isto é, que a função $C : \mathcal{X} \to \mathcal{D}^*$ seja injectiva:

$$(x_1 \neq x_2) \Rightarrow (C(x_1) \neq C(x_2)), \tag{2.3}$$

onde x_1 e x_2 são dois símbolos arbitrários de $\mathcal{X}$. Esta condição garante que se se enviar uma palavra de código para um receptor, este pode descodificá-la sem ambiguidade, isto é, pode saber qual o símbolo da fonte que foi codificado. O código definido no Exemplo 2.1 é claramente não singular.

2.1.3 Códigos Univocamente Descodificáveis

A condição de não singularidade, se bem que razoável, é em geral insuficiente se se pretender usar o código para enviar, não um único símbolo, mas uma sequência de símbolos. Ilustre-se esta afirmação com o exemplo seguinte.

Exemplo 2.3 *Considere-se* $\mathcal{X} = \{a, b, c, d\}$, $\mathcal{D} = \{0, 1\}$, *e o código binário* C *definido por*

$$C(a) = 0, \quad C(b) = 1, \quad C(c) = 01, \quad C(d) = 10, \tag{2.4}$$

o qual é claramente não singular. Ao receber-se, por exemplo, a sequência 0110*, não é possível determinar se a sequência de símbolos de fonte codificada foi cd, abba, abd, ou cab.*

Para se evitar este tipo de ambiguidade, deve exigir-se ao código que seja "univocamente descodificável"; esta condição é formalizada no parágrafo seguinte.

Seja $x_1, ..., x_n$ uma sequência de n símbolos de $\mathcal{X}$. Considere-se um código $C : \mathcal{X} \to \mathcal{D}$. A extensão de ordem n do código C, denotada C^n, é uma função de $\mathcal{X}^n$ para $\mathcal{D}^*$ definida pela simples concatenação das palavras de C, isto é,

$$C^n(x_1, ..., x_n) = C(x_1)\, C(x_2)\, C(x_n).$$

Exemplo 2.4 *A extensão de ordem 2 do código* C *definido no Exemplo 2.3 é*

$$\begin{aligned}
C^2(aa) &= C(a)C(a) = 00 \\
C^2(ab) &= C(a)C(b) = 01 \\
C^2(ac) &= C(a)C(c) = 001 \\
C^2(ad) &= C(a)C(d) = 010 \\
C^2(ba) &= C(b)C(a) = 10 \\
C^2(bb) &= C(b)C(b) = 11 \\
C^2(bc) &= C(b)C(c) = 101 \\
C^2(bd) &= C(b)C(d) = 110 \\
C^2(ca) &= C(c)C(a) = 010 \\
C^2(cb) &= C(c)C(b) = 011 \\
C^2(cc) &= C(c)C(c) = 0101 \\
C^2(cd) &= C(c)C(d) = 0110 \\
C^2(da) &= C(d)C(a) = 100 \\
C^2(db) &= C(d)C(b) = 101 \\
C^2(dc) &= C(d)C(c) = 1001 \\
C^2(dd) &= C(d)C(d) = 1010
\end{aligned}$$

O código denotado C^*, a que se chama simplesmente extensão (sem ordem) do código C, é obtido do mesmo modo mas considerando todos as sequências de $\mathcal{X}$ de qualquer comprimento. Um código C é dito *univocamente descodificável* se a sua extensão C^* for não singular.

Exemplo 2.5 *A extensão C^* do código C definido no Exemplo 2.3 é*

$$\begin{aligned}
C^*(a) &= C(a) = 0\\
C^*(b) &= C(b) = 1\\
C^*(c) &= C(c) = 01\\
C^*(d) &= C(d) = 10\\
C^*(aa) &= C(a)C(a) = 00\\
C^*(ab) &= C(a)C(b) = 01\\
&\vdots\\
C^*(dc) &= C(d)C(c) = 1001\\
C^*(dd) &= C(d)C(d) = 1010\\
C^*(aaa) &= C(a)C(a)C(a) = 000\\
C^*(aab) &= C(a)C(a)C(b) = 001\\
&\vdots
\end{aligned} \tag{2.5}$$

Assim, o código C não é univocamente descodificável pois $C^*(c) = 01$ e $C^*(ab) = 01$, pelo que C^* é singular (corresponde a uma função não injectiva).

A verificação formal da condição de descodificabilidade unívoca de um código pode ser feita através do *teste de Sardinas-Patterson*; dado que, como se verificará mais adiante, a propriedade de descodificabilidade unívoca não é suficiente para tornar um código útil, não se incluirá um descrição desse teste neste texto.

2.1.4 Códigos Instantâneos

Embora a propriedade de descodificabilidade unívoca seja claramente desejável, pode ser, na prática, insuficiente: para descodificar um símbolo, pode ser necessário observar muitos símbolos seguintes, o que dá ao processo de descodificação uma grande demora.

Exemplo 2.6 *Considere-se o código univocamente descodificável $C : \{a, b, c, d\} \to \{0, 1\}^*$, definido por $C(a) = 01$, $C(b) = 11$, $C(c) = 00$, $C(d) = 110$. Se um receptor receber, por exemplo, a sequência*

$$11\,\underbrace{0\,0...0...0}_{n\ zeros}\,11,$$

a sua descodificação é simples:

$$\begin{aligned}
b\underbrace{c\,c...c...c}_{\frac{n}{2}\ c's}b &\Leftarrow n\ par\\
d\underbrace{c\,c...c...c}_{\frac{n-1}{2}\ c's}b &\Leftarrow n\ ímpar.
\end{aligned}$$

Observe-se que, para descodificar o primeiro símbolo, pode ser necessário observar um número arbitrariamente grande de palavras de código subsequentes, introduzindo um grande atraso no processo de comunicação. Por este motivo, este código é dito não instantâneo; nem sempre é possível identificar uma palavra de código de forma instantânea. É fácil constatar que este facto se deve a que uma das palavras do código, $C(b) = 11$, é prefixo de uma outra palavra de código, $C(d) = 110$.

É evidente que, se nenhuma palavra de código for prefixo de outra, as palavras são descodificadas instantaneamente; os códigos com esta propriedade são ditos *instantâneos*, ou *de prefixo* (*prefix codes*).

2.2 Desigualdade de Kraft-McMillan

O objectivo de desenho de um código é, naturalmente, obter o menor comprimento médio possível. Como é óbvio, não é possível reduzir arbitrariamente o comprimento de todas as palavras, mantendo o código instantâneo; por exemplo, num código binário, para um alfabeto de fonte com mais de dois símbolos, se 0 for uma palavra de código, não pode existir mais nenhuma palavra com apenas um bit, pois todas as restantes palavras têm de começar por 1 (para que 0 não seja prefixo de nenhuma delas). Este facto é expresso formalmente pela teorema de Kraft-McMillan, mais conhecido como *desigualdade de Kraft-McMillan* (DKM).

DKM: Seja $C : \mathcal{X} \to \mathcal{D}$ um código, com um alfabeto de código $\mathcal{D}$ com D símbolos. Se C é instantâneo, os comprimentos das suas palavras verificam

$$\sum_{x \in \mathcal{X}} D^{-l_C(x)} \leq 1. \tag{2.6}$$

Reciprocamente, dado um conjunto de números naturais $l_1, ..., l_N \in \mathbb{N}$ que verifique

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

existe um código D-ário instantâneo para um alfabeto com N símbolos, cujos comprimentos são esses números.

Demonstração 1: Seja $l_{\max}$ um número maior ou igual ao comprimento máximo das palavras do código, isto é,

$$l_{\max} \geq \max\{l_C(x),\ x \in \mathcal{X}\}.$$

Para uma dada palavra $C(x)$, de comprimento $l_C(x)$, o número de palavras de comprimento $l_{\max}$ que possuem $C(x)$ como prefixo é

$$D^{(l\max - l_C(x))}.$$

Para dois símbolos diferentes x_1 e x_2, dado que o código é instantâneo, os conjuntos das palavras de comprimento $l_{\max}$ que possuem $C(x_1)$ e $C(x_2)$ como prefixos são disjuntos.

Como o número total de palavras de comprimento $l_{\max}$ é $D^{l\max}$, tem-se que

$$\sum_{x\in\mathcal{X}} D^{(l\max - l_C(x))} \leq D^{l\max}; \tag{2.7}$$

dividindo ambos os termos por $D^{l\max}$, obtém-se (2.6).

A implicação reciproca resulta do facto de ser possível associar, a cada código de prefixo, uma árvore D-ádica cujas folhas são as palavras de código. Dado que o código é de prefixo, existe um e um só caminho da raiz da árvore até cada uma das folhas. Para esclarecer esta afirmação, apresenta-se na figura 2.1 a árvore correspondente ao código instantâneo definido por $C(a) = 0$, $C(b) = 10$, $C(c) = 110$, $C(d) = 111$. A descodificação de uma palavra de código corresponde a um trajecto da raiz até uma das folhas, no qual cada símbolo de código indica que ramo seguir a partir de cada nó interior da árvore.

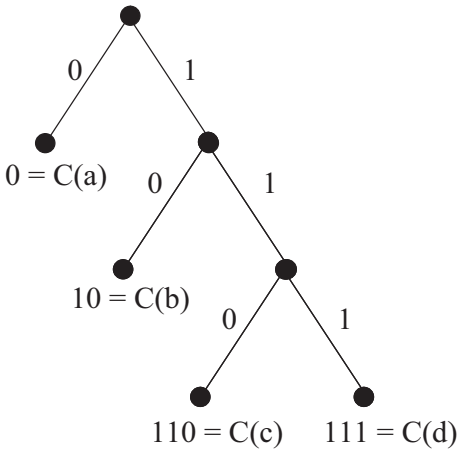

Figura 2.1: Árvore binária associada ao código instantâneo definido por $C(a) = 0$, $C(b) = 10$, $C(c) = 110$, $C(d) = 111$.

A demonstração acima apresentada da DKM suporta-se na existência de um limite superior para o comprimento das palavras de código, que foi designado como $l_{\max}$. Em certas circunstâncias, pode ser necessário desenhar códigos instantâneos para fontes com alfabetos infinitos, $\mathcal{X} = \{x_1, x_2, ..., x_n, ...\}$; por exemplo, é por vezes necessário usar códigos instantâneos para números inteiros arbitrariamente grandes. Em tais casos, não é possível estabelecer *a priori* um limite superior $l_{\max}$ para o comprimento das palavras de código, pelo que a demonstração apresentada no parágrafo anterior não se pode aplicar. Existe uma demonstração alternativa da DKM, sem recurso a $l_{\max}$ e, como tal, aplicável a alfabetos infinitos, que de seguida se apresenta.

Demonstração 2: Considere-se um código instantâneo para uma fonte com alfabeto $\mathcal{X}$, não necessariamente finito. Designem-se os elementos do alfabeto de $\mathcal{D}$, sem perda de generalidade, como $\mathcal{D} = \{0, 1, ..., D-1\}$, ou seja, os números inteiros de 0 a $D-1$. Cada palavra de código $C(x)$ é um elemento de $\mathcal{D}^*$, isto é, uma sequência constituída por $l(x)$

elementos de $\mathcal{D}$. Explicitamente, escreva-se

$$C(x) = d_1(x)\, d_2(x) \ldots d_{l_C(x)}(x),$$

em que $d_i \in \mathcal{D}$. A cada uma destas sequências pode fazer-se corresponder um número $\alpha(x)$ no intervalo $[0,\ 1[$, cuja expansão $D-$ária é dada pelos símbolos $d_i(x)$, para $i = 1, ..., l_C(x)$, isto é,

$$\alpha(x) = 0.d_1(x)\, d_2(x) \ldots d_{l_C(x)}(x) = \sum_{i=1}^{l_C(x)} d_i(x)\ D^{-i}.$$

Claramente, verifica-se que, por construção, $\alpha(x) \in [0,\ 1[$, para todos os $x \in \mathcal{X}$. Considere-se agora, para cada x, o intervalo (fechado à esquerda e aberto à direita)

$$I(x) = \Big[\alpha(x),\ \alpha(x) + D^{-l_C(x)}\Big[,$$

o qual contem todos os números cuja expansão D-ária tem como prefixo a expansão D-ária de $\alpha(x)$, ou seja, $0.d_1(x)\, d_2(x) \ldots d_{l(x)}(x)$.

Os dois exemplos seguintes ajudarão a tornar mais clara esta construção.

Exemplo 2.7 *No caso de um código* 10*-ário (ou decimal), tem-se* $D = 10$ *e* $\mathcal{D} = \{0, 1, ..., 9\}$*; considere-se que a palavra de código* $C(x_n)$*, para um dado símbolo* $x_n \in \mathcal{X}$*, é* $C(x_n) = 2738$*; assim,* $\alpha(x_n) = 0.2738$ *(na habitual escrita em base 10) e* $I(x_n) = [0.2738,\ 0.2739[$*, o qual é o intervalo de todos os números reais cuja escrita decimal começa por* 0.2738*; por exemplo* $0.273845 \in [0.2738,\ 0.2739[$.

Exemplo 2.8 *No caso de um código binário, com* $D = 2$ *e* $\mathcal{D} = \{0, 1\}$*, suponha-se que um dado símbolo* $x_m \in \mathcal{X}$ *tem o código* $C(x_m) = 100101$*; neste caso,* $\alpha(x_m) = 0.100101$ *(em base 2, ou seja, traduzindo para base 10,* $\alpha(x_m) = 1/2 + 1/16 + 1/64 = 0.5781$*). O correspondente intervalo é* $I(x_m) = [0.100101,\ 0.10011[$ *(pois, em base 2,* $0.100101 + 0.000001 = 0.10011$*), o qual contém todos os números cuja escrita em base 2 começa por* 0.100101*; por exemplo* $0.100101101 \in [0.100101,\ 0.10011[$.

As três observações fundamentais que permitem concluir a demonstração são: comprimento de cada intervalo $I(x)$, designado $|I(x)|$ é $D^{-l_C(x)}$; dado que o código é instantâneo, nenhuma palavra é prefixo de outra, pelo que todos os intervalos $I(x)$ são disjuntos; todos os intervalos $I(x)$ estão contidos no intervalo $[0,\ 1[$, cujo comprimento $|[0,\ 1[| = 1$. Assim,

$$1 = \Big|[0,\ 1[\Big| \geq \left|\bigcup_{x\in\mathcal{X}} I(x)\right| = \sum_{x\in\mathcal{X}} |I(x)| = \sum_{x\in\mathcal{X}} D^{-l_C(x)},$$

pois o comprimento da união de intervalos disjuntos é igual à soma dos comprimentos dos intervalos.

2.3 Códigos Ideais e Códigos Óptimos

Como referido acima, o objectivo de desenho de um código é, naturalmente, obter o menor comprimento médio possível, sob a constrição de que o código obtido seja instantâneo. A desigualdade de Kraft-McMillan, que acabou de ser apresentada e demonstrada, permite impor formalmente a restrição de que os códigos considerados possuam comprimentos compatíveis com a propriedade de descodificabilidade instantânea. Formalmente, o desenho de um código óptimo apresenta-se como um problema de optimização com restrições. Para uma fonte de alfabeto $\mathcal{X}$, cujos símbolos são emitidos com probabilidades $\{p(x),\ x \in \mathcal{X}\}$, o código *ideal* (em breve justificar-se-á o uso do termo *ideal*, em vez de *óptimo*) possui comprimentos $\{l^*(x),\ x \in \mathcal{X}\}$ dados por

$$\{l^*(x),\ x \in \mathcal{X}\} = \text{solução de } \begin{cases} \text{minimizar} & \displaystyle\sum_{x\in\mathcal{X}} l(x)\,p(x) \\ \text{sob a restrição} & \displaystyle\sum_{x\in\mathcal{X}} D^{-l(x)} \leq 1. \end{cases} \tag{2.8}$$

Embora este problema se possa facilmente atacar usando a técnica dos multiplicadores de Lagrange (ver, por exemplo, [4]), pode usar-se uma abordagem indirecta baseada na desigualdade da informação e formalizada na seguinte desigualdade.

Desigualdade Fundamental da Codificação de Fonte: Seja uma uma fonte X, de alfabeto $\mathcal{X}$, cujos símbolos são emitidos com probabilidades $\{p(x),\ x \in \mathcal{X}\}$. Qualquer código cujos comprimentos verifiquem a desigualdade de Kraft-McMillan, ao ser usado para codificar essa fonte, apresenta um comprimento médio maior ou igual à entropia da fonte, isto é,

$$\left(\sum_{x\in\mathcal{X}} D^{-l(x)} \leq 1\right) \Rightarrow \left(L(C) = \sum_{x\in\mathcal{X}} p(x)\,l_C(x) \geq H_D(X)\right), \tag{2.9}$$

com igualdade se e só se $l(x) = -\log_D p(x)$, para todos os $x \in \mathcal{X}$, e onde $H_D(X)$ denota simplesmente a entropia calculada usando logaritmos de base D.

Demonstração: Considere-se $D = 2$, por simplicidade; a demonstração é trivialmente modificada para qualquer valor de $D > 1$. Escreva-se $L(C) - H(X)$, que se pretende demonstrar ser maior ou igual que zero,

$$L(C) - H(X) = \sum_{x\in\mathcal{X}} p(x)\,l_C(x) + \sum_{x\in\mathcal{X}} p(x)\log p(x). \tag{2.10}$$

Note-se que se pode escrever $l_C(x) = -\log_2 2^{-l_C(x)}$; introduzindo esta igualdade acima, vem

$$\begin{aligned} L(C) - H(X) &= -\sum_{x\in\mathcal{X}} p(x)\log 2^{-l_C(x)} + \sum_{x\in\mathcal{X}} p(x)\log p(x) \\ &= \sum_{x\in\mathcal{X}} p(x)\log\frac{p(x)}{2^{-l_C(x)}}. \end{aligned}$$

Multiplicando e dividindo o argumento de cada logaritmo por $A = \sum_{x' \in \mathcal{X}} 2^{-l_C(x')}$,

$$\begin{aligned} L(C) - H(X) &= \sum_{x \in \mathcal{X}} p(x) \log \frac{A\, p(x)}{A\, 2^{-l_C(x)}} \\ &= \sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{\frac{2^{-l_C(x)}}{A}} - \sum_{x \in \mathcal{X}} p(x) \log A. \end{aligned} \tag{2.11}$$

Definindo-se $q(x) = 2^{-l(x)}/A$ tem-se que $q(x) \geq 0$, para todos os $x \in \mathcal{X}$, bem como

$$\sum_{x \in \mathcal{X}} q(x) = \frac{1}{A} \sum_{x \in \mathcal{X}} 2^{-l_C(x)} = 1,$$

pelo que pode interpretar-se o primeiro somatório em (2.11) como uma divergência de Kullback-Leibler. Assim,

$$L(C) - H(X) = \underbrace{\sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{q(x)}}_{= D_{KL}(p\|q) \geq 0} \underbrace{- \underbrace{\log A}_{\leq 0} \overbrace{\sum_{x \in \mathcal{X}} p(x)}^{=1}}_{\geq 0} \geq 0,$$

onde $\log A \leq 0$ porque, pela desigualdades de Kraft-McMillan, $A \leq 1$.

Para se ter igualdade é necessário $\log A = 0$, isto é $A = \sum_{x \in \mathcal{X}} 2^{-l_C(x)} = 1$, o que corresponde a um código que verifica a DKM com igualdade, e ainda $D_{KL}(p\|q) = 0$, ou seja (usando o facto que $A = 1$),

$$p(x) = q(x) = \frac{2^{-l_C(x)}}{A} = 2^{-l_C(x)} \quad \Leftrightarrow \quad l_C(x) = -\log p(x), \tag{2.12}$$

para todos os $x \in \mathcal{X}$.

A desigualdade que acabou de ser demonstrada fornece a solução para o problema enunciado em (2.8). O menor valor possível para $\sum_{x \in \mathcal{X}} p(x)\, l_C(x)$, sob a condição que os $l_C(x)$ verifiquem a DKM é dado precisamente por (2.12). Estes comprimentos serão designados ditos *ideais* e denotados como

$$l^*(x) = -\log p(x) = \log \frac{1}{p(x)}. \tag{2.13}$$

Por construção, verificam a DKM, e conduzem a um valor esperado igual à entropia (como se verificou na demonstração anterior). O motivo pelo qual se designam estes comprimentos como *ideais*, e não óptimos, é o seguinte: os valores $l^*(x)$ podem não ser (em geral não são) números inteiros, pelo que não é possível construir palavras de código com esses comprimentos. Ignorando, por momentos, esta restrição, deve observar-se que a interpretação de (2.13) é simples: aos símbolos mais prováveis atribuem-se palavras mais curtas e aos símbolos menos prováveis correspondem palavras de código mais longas. Apenas é possível construir um código instantâneo com comprimentos dados por $l^*(x)$ se estes forem inteiros; isto sucede se e só se todas as probabilidades $p(x)$ forem potências de 2 (ou de D, no caso dum alfabeto de código

D-ário), necessariamente de expoente negativo, pois $p(x) \leq 1$. Uma função de probabilidade em que todos os valores são potências de 2 diz-se diádica (ou D-ádica, no caso geral).

Para obter comprimentos inteiros é necessário impor essa restrição adicional no problema de optimização que conduz aos comprimentos do código *óptimo*:

$$\{l^{\text{opt}}(x),\ x \in \mathcal{X}\} = \text{solução de} \begin{cases} \text{minimizar} & \displaystyle\sum_{x\in\mathcal{X}} l(x)\,p(x) \\ \text{sob as restrições} & \displaystyle\sum_{x\in\mathcal{X}} D^{-l(x)} \\ & l(x) \in \mathbb{N},\ \ \forall_{x\in\mathcal{X}}. \end{cases} \tag{2.14}$$

A introdução da restrição adicional confere a este problema um carácter combinatório, deixando, como tal, de poder ser resolvido com ferramentas de análise de funções reais de variáveis reais (recorde-se que, subjacentes à desigualdade da informação, estão propriedades de convexidade da função logaritmo). A um código instantâneo cujas palavras apresentam os comprimentos óptimos chama-se código óptimo e denota-se como C^{opt}.

2.4 Limites para os Códigos Óptimos

Antes de introduzir a solução de (2.14) (o que será feito na secção 2.7), apresentam-se alguns resultados que se podem obter sem usar explicitamente essa solução.

Uma possibilidade para obter comprimentos inteiros a partir dos valores $l^*(x)$ é considerar os menores inteiros não inferiores a $l^*(x)$; esta é precisamente a definição da função *ceiling* (ou "tecto"):

$$\lceil z \rceil = \text{menor inteiro não inferior a } z.$$

Assim, definem-se os chamados *comprimentos de Shannon*, dados por

$$l^s(x) = \lceil l^*(x) \rceil = \lceil -\log p(x) \rceil\,.$$

É fácil verificar que estes comprimentos verificam a DKM (os logaritmos que surgem são na base D),

$$\sum_{x\in\mathcal{X}} D^{-l^s(x)} = \sum_{x\in\mathcal{X}} D^{-\lceil -\log p(x)\rceil} \leq \sum_{x\in\mathcal{X}} D^{\log p(x)} = \sum_{x\in\mathcal{X}} p(x) = 1.$$

pois, para qualquer número real z, tem-se $\lceil z \rceil \geq z$ e, como tal, $D^{-\lceil z\rceil} \leq D^{-z}$. Assim, é possível construir um código instantâneo, designado C^s, com estes comprimentos.

Embora não seja necessariamente óptimo, este código não se afasta muito do limite inferior dado pela entropia da fonte. De facto,

$$\begin{aligned} E[l^s(X)] &= \sum_{x\in\mathcal{X}} p(x)\lceil -\log p(x)\rceil \\ &< \sum_{x\in\mathcal{X}} p(x)\,(-\log p(x) + 1) \\ &= -\sum_{x\in\mathcal{X}} p(x)\log p(x) + \sum_{x\in\mathcal{X}} p(x) \\ &= H(X) + 1, \end{aligned} \tag{2.15}$$

devido à desigualdade $\lceil z \rceil < z + 1$, válida para qualquer real z.

Finalmente, pode estabelecer-se os limites inferior e superior para o comprimento médio do código óptimo.

Limites Para o Código Óptimo: O valor esperado do comprimento de um código instantâneo óptimo verifica:

$$H(X) \leq L(C^{\text{opt}}) < H(X) + 1. \tag{2.16}$$

Demonstração: A desigualdade da esquerda é uma simples consequência de C^{opt} ser instantâneo, logo verificar a DKM e, consequentemente, possuir um valor esperado do comprimento não inferior à entropia da fonte. A segunda desigualdade é um simples corolário de (2.15): de facto, se C^{opt} é óptimo, o seu comprimento esperado não pode exceder o de C^s (ou não seria óptimo) donde $L(C^{\text{opt}}) \leq L(C^s) < H(X) + 1$.

Em conclusão, quer o código de Shannon C^s, quer o código óptimo C^{opt}, têm um valor esperado de comprimento que se situa menos de 1 bit/símbolo acima da entropia da fonte. Este excesso pode ser desprezável no caso de fontes de entropia elevada (definidas sobre alfabetos grandes), mas pode ser relativamente grave no caso de fontes de baixa entropia (por exemplo, com alfabetos pequenos). Esta observação é relevante pois é fácil demonstrar que existem fontes cuja valor esperado do comprimento dos códigos óptimos estão arbitrariamente próximos de $H(X) + 1$.

Exemplo 2.9 *Considere-se uma fonte com um alfabeto de apenas dois símbolos,* $\mathcal{X} = \{a, b\}$. *Existem apenas dois códigos binários para esta fonte: código* C_1, *definido por* $C_1(a) = 0$ *e* $C_1(b) = 1$; *código* C_2, *definido por* $C_2(a) = 1$ *e* $C_2(b) = 0$. *Dado que ambos têm valor esperado do comprimento igual a 1, tem-se* $L(C^{opt}) = 1$. *Como a entropia de uma fonte com dois símbolos pode ser arbitrariamente próxima de zero (ver Figura 1.1), pode ter-se* $H(X) + 1$ *arbitrariamente próximo de* $L(C^{opt})$.

2.5 Extensões de Fonte

O limite superior para o valor esperado do comprimento de um código óptimo, apresentado na secção anterior, sugere que a codificação óptima pode ser pouco eficaz para fontes de muito baixa entropia. A forma de contornar esta dificuldade em codificar fontes de entropia muito baixa (por exemplo, com alfabetos muito pequenos) consiste em codificar os símbolos, não individualmente, mas sim em grupos. Esta ideia formaliza-se usando o conceito de *extensão da fonte*. Considere-se uma fonte X, sem memória, emitindo símbolos de um alfabeto $\mathcal{X}$, com função de probabilidade $p(x)$; a extensão de ordem n dessa fonte, designada $X_{(n)}$, obtém-se agrupando os símbolos gerados pela fonte em grupos de n, ou seja $X_{(n)} = (X_1, ..., X_n)$, onde todos os X_i são independentes (a fonte não possui memória) e identicamente distribuídos de acordo com $p(x)$. Note-se que agrupar n amostras de uma fonte sem memória é equivalente a considerar n cópias independentes da fonte original. A nova fonte $X_{(n)}$ gera símbolos no alfabeto estendido $\mathcal{X}^n$ (n-ésima potência cartesiana de $\mathcal{X}$).

Um código óptimo para este fonte estendida, designado C_n^{opt}, apresenta comprimento médio

$$L(C_n^{\text{opt}}) = \sum_{x_1 \in \mathcal{X}} \cdots \sum_{x_n \in \mathcal{X}} p(x_1, ..., x_n)\; l_n^{\text{opt}}(x_1, ..., x_n),$$

onde $p(x_1, ..., x_n)$ é a probabilidade da sequência de símbolos $(x_1, ..., x_n)$ e $l_n^{\text{opt}}(x_1, ..., x_n)$ é o comprimento da palavra de código óptimo para a sequência de símbolos $(x_1, ..., x_n)$. Como qualquer outro código óptimo, C_n^{opt} verifica (2.16), ou seja

$$H(X_{(n)}) \leq L(C_n^{\text{opt}}) < H(X_{(n)}) + 1, \tag{2.17}$$

onde $H(X_{(n)}) = H(X_1, ..., X_n)$ é a entropia da fonte extendida. Pelo facto da fonte original não possuir memória, e de todos os X_i possuírem a mesma função de probabilidade, verifica-se que

$$H(X_{(n)}) = H(X_1, ..., X_n) = \sum_{i=1}^{n} H(X_i) = nH(X),$$

e, assim,

$$n\,H(X) \leq L(C_n^{\text{opt}}) < n\,H(X) + 1. \tag{2.18}$$

Note-se, no entanto, que $L(C_n^{\text{opt}})$ é o valor esperado do comprimento do código, por símbolo da fonte estendida. Dado que cada símbolo da fonte estendida é constituído por n símbolos da fonte original, o comprimento médio do código estendido, por símbolo da fonte original, designado como $L_n(C_n^{\text{opt}})$, é dado por $L_n(C_n^{\text{opt}}) = (1/n)L(C_n^{\text{opt}})$. Assim, dividindo todos os termos de (2.18) por n, obtém-se

$$H(X) \leq L_n(C_n^{\text{opt}}) < H(X) + \frac{1}{n}, \tag{2.19}$$

o que mostra que, usando extensões de fonte, podem obter-se códigos cujo valor esperado do comprimento médio por símbolo se aproxima arbitrariamente do valor da entropia da fonte.

Exemplo 2.10 *Para uma fonte com um alfabeto de apenas dois símbolos, $\mathcal{X} = \{a, b\}$, tem-se (como foi visto na secção anterior) que $L(C^{opt}) = 1$, independentemente do valor da entropia da fonte. Assuma-se que $p(a) = 15/16$ e $p(b) = 1/16$, o que corresponde a um valor da entropia $H(X) = 0.3373$ bits/símbolo. Obviamente, verifica-se $H(X) = 0.3373 \leq 1 < H(X) + 1 = 1.3373$ (todas as quantidades em bits/símbolo). Nesta caso, o código óptimo está a $(1 - 0.3373) = 0.6627$ bits/símbolo da entropia. Considere-se agora a extensão de segunda ordem, cujo alfabeto é $\mathcal{X}^2 = \{(a, a), (a, b), (b, a), (b, b)\}$, e cujas probabilidades são, respectivamente, $\{(15/16)^2, 15/16^2, 15/16^2, 1/16^2\} \simeq \{0.8798, 0.0586, 0.0586, 0.0039\}$. Um código instantâneo óptimo para esta fonte estendida pode ser obtido por simples inspecção das probabilidades: $C_n^{opt}(a, a) = 0$, $C_n^{opt}(a, b) = 10$, $C_n^{opt}(b, a) = 110$, $C_n^{opt}(b, b) = 111$. O comprimento médio deste código é*

$$L(C_2^{opt}) = 1 \times 0.8798 + 2 \times 0.0586 + 3 \times 0.0586 + 3 \times 0.0039 \simeq 1.1836 \;\; \textit{bits/símbolo},$$

medido em bits por símbolo da fonte estendida, isto é, por cada par de símbolos da fonte original. Calculando o número de bits usados em média por cada símbolo da fonte original, obtém-se

$$L_2(C_2^{opt}) = \frac{1}{2}\, L(C_2^{opt}) = \frac{1.1836}{2} \simeq 0.5918 \;\; \textit{bits/símbolo},$$

medido em bits por símbolo da fonte original. Note-se que este valor verifica $H(X) = 0.3373 \leq 0.5918 < H(X) + 1/2 = 0.8373$. *Com a extensão, conseguiu-se passar a diferença para a entropia de* $(1 - 0.3373) = 0.6627$ *bits/símbolo para apenas* $(0.5918 - 0.3373) = 0.2545$ *bits/símbolo.*

2.6 Codificação com Modelo Errado

Nesta secção, estuda-se o impacto sobre o valor esperado do comprimento de codificação do uso de uma função de probabilidade errada. Considere-se uma fonte de alfabeto $\mathcal{X}$ cuja função de probabilidade é $p(x)$. Desenha-se um código C com comprimentos de Shannon baseados numa função de probabilidade $q(x)$, não necessariamente igual a $p(x)$, ou seja $l_C(x) = \lceil -\log q(x) \rceil$. Usando o facto de que, para qualquer número a, tem-se $\lceil a \rceil \geq a$, pode concluir-se que

$$\begin{aligned}
L_p(C) &= \sum_{x\in\mathcal{X}} p(x)\, \lceil -\log q(x) \rceil \\
&\geq \sum_{x\in\mathcal{X}} p(x)\, \log \frac{1}{q(x)} \\
&= \sum_{x\in\mathcal{X}} p(x)\, \log \frac{p(x)}{q(x)\, p(x)} \\
&= \sum_{x\in\mathcal{X}} p(x)\, \log \frac{p(x)}{q(x)} + \sum_{x\in\mathcal{X}} p(x)\, \log \frac{1}{p(x)} \\
&= H(p) + D_{\mathrm{KL}}(p\|q), \qquad (2.20)
\end{aligned}$$

onde $L_p(C)$ denota o valor esperado do comprimento do código C, sob a função de probabilidade p. Usando a outra desigualdade para a função "ceiling", isto é, $\lceil a \rceil < a + 1$, obtém-se

$$\begin{aligned}
L_p(C) &= \sum_{x\in\mathcal{X}} p(x)\, \lceil -\log q(x) \rceil \\
&< \sum_{x\in\mathcal{X}} p(x)\, \left(\log \frac{1}{q(x)} + 1 \right) \\
&= \sum_{x\in\mathcal{X}} p(x)\, \left(\log \frac{p(x)}{q(x)\, p(x)} + 1 \right) \\
&= \sum_{x\in\mathcal{X}} p(x)\, \log \frac{p(x)}{q(x)} + \sum_{x\in\mathcal{X}} p(x)\, \log \frac{1}{p(x)} + \sum_{x\in\mathcal{X}} p(x) \\
&= H(p) + D_{\mathrm{KL}}(p\|q) + 1 \qquad (2.21)
\end{aligned}$$

Resumindo (2.20) e (2.21) numa só expressão, obtém-se

$$H(p) + D_{\mathrm{KL}}(p\|q) \leq L_p(C) < H(p) + D_{\mathrm{KL}}(p\|q) + 1.$$

Em conclusão, o facto de se usar uma função de probabilidade errada no desenho de um código com comprimentos de Shannon conduz a um custo adicional, em termos de valor esperado do comprimento, de $D_{\mathrm{KL}}(p\|q)$.

2.7 Codificação de Huffman

Nest secção apresenta-se a solução do problema de optimização conducente aos comprimentos do código óptimo. Dado que se trata de um problema combinatório, pelo facto de se estar sob a condição de que os comprimentos devem ser números inteiros, não é possível aplicar as habituais ferramentas do cálculo (derivadas, multiplicadores de Lagrange, etc.) para deduzir uma solução. É, no entanto, possível apresentar uma solução e demonstrar que é óptima.

2.7.1 Algoritmo de Huffman

O algoritmo apresentado em seguida foi proposto por Huffman em 1952 [6], para resolver o problema da codificação óptima de uma fonte sem memória, isto é, para resolver (2.14). Por agora, considerar-se-ão apenas códigos binários, isto é, com $\mathcal{D} = \{0, 1\}$; mais adiante serão indicadas as diferenças para o caso de alfabetos de código de dimensão D arbitrária.

Considera-se uma fonte X emitindo símbolos de um alfabeto $\mathcal{X} = \{x_1, ..., x_N\}$, com probabilidades $\{p_1, ..., p_N\}$. O algoritmo de Huffman pode ser dividido em duas partes.

Parte A: Repetem-se os seguintes passos:

- **Passo A.1:** Ordenam-se os símbolos por ordem decrescente de probabilidade.
- **Passo A.2:** Agrupam-se os dois símbolos menos prováveis num "super-símbolo" cuja probabilidade é a soma das probabilidades dos dois símbolos agrupados (o alfabeto resultante possui um símbolo a menos).
- **Passo A.3:** Se o alfabeto resultante possui dois ou mais símbolos, volta-se ao Passo A.1; caso contrário, está concluída a parte A.

Parte B: A Parte A do algoritmo produziu uma arvore binária na qual as folhas (nós sem descendentes) correspondem aos símbolos da fonte. Basta agora percorrer a árvore da raiz atá cada uma das folhas, atribuindo (de forma arbitrária) os símbolos "0" e "1" a cada par de ramos com origem em cada nó interno (não folha) da árvore.

No caso de existirem, em algum ponto do algoritmo, dois ou mais símbolos (ou "super-símbolos") com o mesmo valor de probabilidade a ordenação destes símbolos é arbitrária, sem que isso tenha qualquer impacto no comprimento médio do código resultante.

Apresenta-se de seguida um exemplo. Considere-se o alfabeto $\mathcal{X} = \{a, b, c, d\}$, com probabilidades, respectivamente, $\{0.1, 0.25, 0.2, 0.45\}$. O primeiro passo consiste em ordenar o alfabeto por ordem decrescente de probabilidades; o resultado desta ordenação é $\{d, b, c, a\}$, com probabilidades ordenadas $\{0.45, 0.25, 0.2, 0.1\}$. No passo A.1 agrupam-se os dois símbolos menos prováveis, c e a, num "super-símbolo" (a, c) com probabilidade $p(a)+p(c) = 0.1+0.2 = 0.3$. Este passo está ilustrado na figura 2.2 (a); nesta figura, junto a cada nó da árvore binária que

vai sendo construída está indicado qual o conjunto de símbolos que lhe corresponde e qual a correspondente probabilidade total. O alfabeto resultante é $\{d,\ b,\ (a,c)\}$, com probabilidades $\{0.45,\ 0.25,\ 0.3\}$, no qual os símbolos a e c do alfabeto original foram substituídos pelo "super-símbolo" (a,c). Dado que este alfabeto possui dois ou mais (neste caso três) símbolos, volta-se ao passo A.1. Reordenando o alfabeto obtém-se $\{d,\ (a,c),\ b\}$, com probabilidades ordenadas $\{0.45,\ 0.3,\ 0.25\}$. Procede-se agora ao passo A.2, no qual se agrupam os símbolos de menor probabilidade: (a,c) e b, num novo "super-símbolo" $((a,c),b)$. Este passo está ilustrado na figura 2.2 (b). O alfabeto resultante é $\{d,((a,c),b)\}$, com probabilidades $\{0.45,\ 0.55\}$. Reordenando e agrupando, obtém-se finalmente um alfabeto com apenas um símbolo $\{((a,c),b,d)\}$, naturalmente com probabilidade 1. A árvore final que se obtém está representada na figura 2.2 (c); nesta árvore, a raiz corresponde ao "super-símbolo" final $((a,c),b,d)$.

Na parte B do algoritmo, toma-se a árvore produzida na parte A e etiquetam-se os ramos que emanam de cada bifurcação, nos caminhos que vão da raiz até às folhas, com o símbolo "1" para um dos ramos e o símbolo "0" para o outro. Para se obter a palavra de código para cada símbolo, basta registar as etiquetas binárias dos ramos percorridos no caminho da raiz até esse símbolo. Este procedimento está representado na figura 2.3.

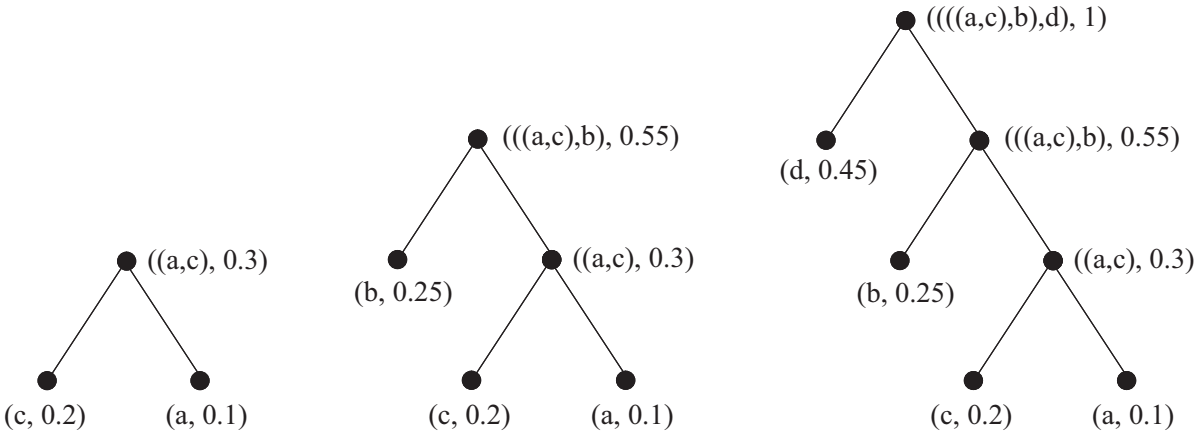

Figura 2.2: Sequência de árvores binárias produzidas pela parte A do algoritmo de Huffman, para o exemplo apresentado no texto.

2.7.2 Escrita Recursiva do Algoritmo de Huffman

É possível escrever o algoritmo de Huffman de forma recursiva. Para tal, é necessário começar por constatar que, de facto, o algoritmo é intrinsecamente recursivo: após proceder á criação do "super-símbolo", por agregação dos dois símbolos menos prováveis, o algoritmo obtém o código de Huffman para o alfabeto reduzido. As palavras de código para os símbolos que deram origem ao "super-símbolo" obtêm-se simplesmente acrescentando um "1" e um "0" à palavra de código atribuída ao "super-símbolo" no código para o alfabeto de dimensão reduzida. Resumindo, para obter um código de Huffman para um alfabeto de M símbolos, é necessário obter um código de Huffman para um alfabeto de $M-1$ símbolos. A recursão termina quando se pretende obter um código de Huffman para um alfabeto de 2 símbolos, pois neste caso a resposta é trivial.

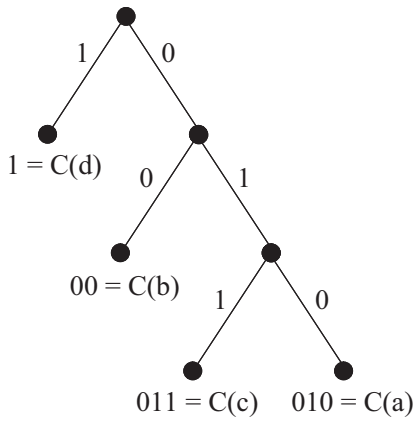

Figura 2.3: Árvore binária, com etiquetas nos ramos e correspondentes palavras de código.

Apresenta-se na figura 2.4 uma função, a que se chamou `huff`, escrita numa pseudo-linguagem, inspirada na linguagem MATLAB, que implementa o algoritmo de Huffman usando recursividade. A função recebe como argumento um vector de probabilidades $\mathtt{p} = \{\mathtt{p}_1, ..., \mathtt{p}_N\}$ e devolve um vector de sequências binárias $\mathtt{C} = \{\mathtt{C}_1, ..., \mathtt{C}_N\}$. As funções auxiliares usadas, bem como outros aspectos da pseudo-linguagem, são de seguida apresentados:

- `length`: devolve o número de elementos de um vector.
- `sort`: ordena um vector por ordem crescente, devolvendo também as respectivas posições que os elementos ordenados ocupavam no vector original. Por exemplo, se $\mathtt{p} = \{12, 9, 4, 7\}$, o resultado de [q,s]=sort(p) é $\mathtt{q} = \{4, 7, 9, 12\}$ e $\mathtt{s} = \{3, 4, 2, 1\}$.
- `min`: devolve o menor dos dois argumentos.
- `max`: devolve o maior dos dois argumentos.
- `strcat`: (abreviatura de *string concatenation*) devolve a concatenação dos seus dois argumentos. Por exemplo, `strcat("ab", "cd")` devolve `"abcd"`.
- O acesso a elementos individuais ou sequências de elementos de um vector é efectuado da forma habitual. Por exemplo, se `C = {"010","011","00","1"}`, então, `C(2)` é `"011"` e `C(2:4)` é `{"011","00","1"}`.
- As linhas de código começadas por `%` são apenas comentários.

Note-se que a função possui apenas 18 linhas de código efectivo (as restantes são comentários para facilitar a sua leitura). Deve, no entanto, referir-se que esta não é necessariamente a implementação mais eficiente do algoritmo de Huffman; é, no entanto, extremamente compacta e ilustra bem a sua natureza recursiva, a qual desempenha um papel central na demonstração da optimalidade dos códigos resultantes.

```
function C = huff(p)
N = length(p)
if N=2
   C(1) = "0";
   C(2) = "1";
else
   [psorted , indices] = sort(p);
   % the two smallest probabilities in p are
   % psorted(1) and psorted(2) or, equivalently,
   % p(indeces(1)) and p(indeces(2)).
   % Their locations are indeces(1) and indeces(2).

   % Now, we need to find which of these two positions
   % is the leftmost, that is, appears first in p.
   first = min(indeces(1),indeces(2));
   second = max(indeces(1),indeces(2));

   % Now we build a new vector of probabilities, called paux,
   % with N-1 elements.  The two smallest probabilities in p are added
   % and stored in paux(first)
   % Example:  if p={0.2 0.1 0.2 0.05 0.2 0.25},
   %           the two smallest probabilities are 0.1 and 0.05, and
   %           paux = {0.2 0.15 0.2 0.2 0.25}
   paux(first) = psorted(1) + psorted(2);
   paux(1:first-1) = p(1:first-1);
   paux(first+1:second-1) = p(first+1:second-1);
   paux(second:N-1) = p(second+1:N);

   % Now we ask for the Huffman code for the probabilities in paux.
   % by calling the function huff itself.  Here's the recursiveness!
   Caux = huff(paux);

   % Now, we have the Huffman code for the vector of probabilities paux.
   % To obtain the Huffman code for the full vector p, we simply
   % split the "super-symbol" into its two original components,
   % and append "0" and "1" to their codewords.
   C(1:second-1) = Caux(1:second-1);
   C(first) = strcat(C(first),"0"));
   C(second) = strcat(Caux(first),"1"));
   C(second+1:N) = Caux(second:N-1);
endif
```

Figura 2.4: Função `huff` que implementa o algoritmo de Huffman explorando a sua natureza recursiva; note-se que a função tem apenas 18 linhas de código.

2.7.3 Demonstração de Optimalidade

A demonstração de optimalidade dos códigos de Huffman suporta-se num lema auxiliar que de seguida se apresenta e demonstra.

Lema 1: Considere-se um código C desenhado para uma fonte com um alfabeto de N símbolos, $\mathcal{X}$, emitidos com probabilidades $\{p_1, ..., p_N\}$. Sejam $\{l_1, ..., l_N\}$ os comprimentos das palavras desse código. Sem perda de generalidade (pois pode sempre reordenar-se os símbolos à partida), considere-se que as probabilidades estão ordenadas por ordem decrescente, isto é, $p_1 \geq p_2 \geq ... \geq p_N$. Se existir um grupo de m (com $m \geq 2$) símbolos com igual probabilidade, isto é, $p_i = p_{i+1} = ... = p_{i+m}$, assume-se que os comprimentos das palavras estão ordenados por ordem crescente, isto é, $l_i \leq l_{i+1} \leq ... \leq l_{i+m}$; note-se que isto não afecta o comprimento médio do código nem a sua optimalidade. Então, se C for um código instantâneo óptimo, tem de verificar as seguintes propriedades:

a) A símbolos mais prováveis não podem corresponder palavras de código mais longas, isto é, $(p_i > p_j) \Rightarrow (l_i \leq l_j)$.

Demonstração: Na demonstração nega-se a implicação verificando-se que isso contrariaria a optimalidade de C. Negar a implicação é equivalente a admitir a existência de um código óptimo C com $(p_i > p_j)$ e $(l_i > l_j)$; o comprimento médio de C seria

$$L(C) = \sum_{n=1}^{N} l_n\, p_n = K + l_i\, p_i + l_j\, p_j,$$

onde K representa todos os termos com $n \neq i$ e $n \neq j$. Pode construir-se um outro código C' trocando a palavras i e j, cujo comprimento médio é

$$L(C') = \sum_{n=1}^{N} l_n\, p_n = K + l_j\, p_i + l_i\, p_j.$$

A diferença $L(C') - L(C)$ é

$$\begin{aligned} L(C') - L(C) &= K + l_j\, p_i + l_i\, p_j - K - l_i\, p_i - l_j\, p_j \\ &= l_j(p_i - p_j) + l_i(p_j - p_i) \\ &= \underbrace{(p_i - p_j)}_{>0}\,\underbrace{(l_j - l_i)}_{<0} < 0, \end{aligned}$$

mostrando que $L(C') < L(C)$ o que negaria a optimalidade de C, provando assim a validade da implicação.

b) Aos dois últimos símbolos (na lista ordenada) correspondem palavras de igual comprimento, isto é, $l_{N-1} = l_N$.

Demonstração: Começa por mostrar-se que, necessariamente, $l_{N-1} \leq l_N$; de facto, se $p_{N-1} > p_N$, então $l_{N-1} \leq l_N$, de acordo com a parte (a) do lema acima demonstrada; se $p_{N-1} =$

p_N, então $l_{N-1} \leq l_N$, de acordo com as hipótese do lema relativamente à ordem das palavras de código associadas a símbolos de igual probabilidade. Uma vez demonstrado que $l_{N-1} \leq l_N$, basta agora demonstrar que não é possível ter-se $l_{N-1} < l_N$. Ora se $l_{N-1} < l_N$, como o código é, por hipótese, instantâneo, a palavra $N-1$ não é prefixo da palavra N; como tal, podem truncar-se os bits em excesso mantendo o carácter instantâneo do código, mas reduzindo o comprimento médio e assim negando a sua optimalidade. Isto demonstra que não se pode ter $l_{N-1} < l_N$, o que conjugado com $l_{N-1} \leq l_N$ mostra que $l_{N-1} = l_N$.

c) No conjunto de todas as palavras de comprimento máximo (l_N), há pelo menos um par de palavras que difere apenas no último bit.

Demonstração: Se no conjunto de todas as palavras de comprimento máximo não existir, pelo menos, um par de palavras diferindo apenas no último bit, é obviamente possível truncar esse bit mantendo as palavras todas diferentes. O código obtido tem menor comprimento médio do que C, negando assim a optimalidade de C.

Com base no Lema 1 pode agora apresentar-se a demonstração de que o procedimento de Huffman conduz a um código óptimo.

Demonstração da optimalidade dos códigos de Huffman: Recorde-se que se pretende desenhar um código C_N para um alfabeto com N símbolos $\mathcal{X}_N = \{x_1, ..., x_N\}$; assume-se, sem perda de generalidade, que os símbolos foram previamente ordenados por ordem decrescente de probabilidades: $p_1 \geq p_2 \geq ... \geq p_N$. De acordo com a escrita recursiva do algoritmo de Huffman, obtém-se C_N do seguinte modo.

- Cria-se um alfabeto reduzido, substituindo-se os símbolos x_{N-1} e x_N pelo "super-símbolo" $x_{N-1,N}$, com probabilidade $p_{N-1,N} = p_N + p_{N-1}$. Tem-se assim um alfabeto com $N-1$ símbolos $\mathcal{X}_{N-1} = \{x_1, ..., x_{N-1,N}\}$ com probabilidades $\{p_1, ..., p_{N-1,N}\}$.

- Obtém-se um código de Huffman C_{N-1} para este alfabeto de $N-1$ símbolos $\mathcal{X}_{N-1} = \{x_1, ..., x_{N-1,N}\}$ com probabilidades $\{p_1, ..., p_{N-1,N}\}$.

- As palavras do código de Huffman C_N obtêm-se a partir das palavras do código de Huffman C_{N-1} de acordo com

$$\begin{aligned} C_N(x_1) &= C_{N-1}(x_1) \\ C_N(x_2) &= C_{N-1}(x_2) \\ &\vdots \\ C_N(x_{N-2}) &= C_{N-1}(x_{N-2}) \\ C_N(x_{N-1}) &= C_{N-1}(x_{N-1,N}) + \text{“0”} \\ C_N(x_N) &= C_{N-1}(x_{N-1,N}) + \text{“1”}, \end{aligned}$$

onde $C(x_i) + \text{“0”}$ denota a operação de acrescentar um "0" à palavra de código $C(x_i)$.

A demonstração prossegue adoptando o princípio da indução, de acordo com o qual basta demonstrar as duas seguintes proposições:

- C_2 é óptimo.
- Se C_{N-1} é óptimo, então C_N é óptimo.

Que C_2 é óptimo não carece de demonstração. Para um alfabeto com dois símbolos $\{x_1, x_2\}$, os dois códigos de Huffman possíveis, $\{C_1(x_1) =$ "1", $C_1(x_2) =$ "0"$\}$ e $\{C_2(x_1) =$ "0", $C_2(x_2) =$ "1"$\}$ são ambos claramente óptimos, com comprimento médio igual a 1.

Para demonstrar a implicação expressa pelo passo de indução, demonstra-se a sua equivalente[1]: C_N não é óptimo implica que C_{N-1} não é óptimo. Se C_N não é óptimo, existe um outro código instantâneo C'_N que é óptimo, como tal verificando $L(C'_N) < L(C_N)$. Se C'_N é óptimo, verifica o Lema 1; assim, pela parte (b) do Lema 1, aos dois símbolos menos prováveis correspondem palavras de igual comprimento, isto é, $l'_{N-1} = l'_N$; de acordo com a alínea (c) do Lema 1, pelo menos duas das palavras de comprimento l'_N diferem apenas no último bit. Pode assumir-se, sem perda de generalidade que são as palavras $N-1$ e N, pois caso contrário podem permutar-se palavras de igual comprimento sem afectar o comprimento médio do código. Truncando este último bit a estas duas palavras, obtém-se um código C'_{N-1}. Mostra-se no parágrafo seguinte que $L(C'_{N-1}) < L(C_{N-1})$, o que nega a optimalidade de C_{N-1}, demonstrando assim o passo de indução.

O comprimento médio de C'_{N-1} é

$$\begin{aligned} L(C'_{N-1}) &= p_1 l'_1 + p_2 l'_2 + \cdots + (p_N + p_{N-1})\,(l'_{N-1} - 1) \\ L(C'_{N-1}) &= \underbrace{p_1 l'_1 + p_2 l'_2 + \cdots + p_{N-1} l'_{N-1} + p_N l'_N}_{L(C'_N)} - (p_N + p_{N-1}), \end{aligned}$$

onde se usou o facto de que $l'_{N-1} = l'_N$. Se se repetir o procedimento com C_N obtém-se precisamente o código C_{M-1} pois está-se simplesmente a inverter o processo pelo qual, no algoritmo de Huffman, se obteve C_N a partir de C_{N-1}, ou seja

$$\begin{aligned} L(C_{N-1}) &= p_1 l_1 + p_2 l_2 + \cdots + (p_N + p_{N-1})\,(l_{N-1} - 1) \\ L(C_{N-1}) &= \underbrace{p_1 l_1 + p_2 l_2 + \cdots + p_{N-1} l_{N-1} + p_N l_N}_{L(C_N)} - (p_N + p_{N-1}). \end{aligned}$$

Finalmente,

$$L(C'_{N-1}) - L(C_{N-1}) = L(C'_N) - (p_N + p_{N-1}) - L(C_N) + (p_N + p_{N-1}) = L(C'_N) - L(C_N) < 0,$$

o que significa que se C_N não é óptimo, C_{N-1} também não o é, concluindo-se assim a demonstração do passo indução.

[1]Recorde-se que $(A \Rightarrow B) \Leftrightarrow (\tilde{B} \Rightarrow \tilde{A})$, onde $\tilde{P}$ denota a negação da proposição P.

2.7.4 Algoritmo de Huffman para Alfabetos D-ários

A modificação do algoritmo de Huffman para alfabetos D-ários é simples. Em vez de se agruparem os dois símbolos menos prováveis, agrupam-se os D símbolos menos prováveis. O resultado é uma arvore D-ária da qual é possível obter as palavras do código instantâneo D-ário óptimo. No entanto, existe um pequeno detalhe ao qual é importante prestar atenção: é necessário dispor, até ao final do algoritmo (quando se atinge a raiz da arvore D-ária), de D símbolos para agrupar; se assim não for, desperdiçam-se palavras curtas (ou seja, palavras junto à raiz da árvore).

Ilustra-se o problema referido no parágrafo anterior com um pequeno exemplo. Considere-se uma fonte que emite símbolos do alfabeto $\{a, b, c, d\}$, com probabilidades $\{1/2,\ 1/4,\ 1/8,\ 1/8\}$, para a qual se pretende desenhar um código de Huffman ternário, isto é, com $\mathcal{D} = \{0, 1, 2\}$. Na aplicação directa do algoritmo de Huffman, agrupam-se os três símbolos menos prováveis $\{b, c, d\}$ num "super-símbolo" (b, c, d) com probabilidade $1/2$. O alfabeto resultante possui apenas dois símbolos $\{a, (b, c, d)\}$, pelo que o código óptimo é trivial. Finalmente, o código resultante é $\{C(a) = 1,\ C(b) = 00,\ C(c) = 01,\ C(d) = 02\}$. Ora, este código é claramente não óptimo, pois o código alternativo $\{C'(a) = 1,\ C'(b) = 2,\ C'(c) = 01,\ C'(d) = 02\}$ é ainda instantâneo e tem menor comprimento médio. A origem do problema reside no facto de se ter atingido o estágio final do algoritmo apenas com 2 símbolos para agrupar; isto teve como consequência que uma palavra curta, neste exemplo simplesmente "2", não pôde ser usada.

A solução para este problema é simples. Acrescentam-se ao alfabeto um conjunto de palavras com probabilidade zero, por forma a garantir que seja possível construir uma árvore D-ária completa (isto é, na qual todos os nós internos têm D descendentes) com esse número de folhas. No exemplo anterior, considere-se o novo alfabeto $\{a, b, c, d, e\}$, com probabilidades $\{1/2,\ 1/4,\ 1/8,\ 1/8,\ 0\}$. Da aplicação directa do algoritmo de Huffman resulta agora $\{C(a) = 0,\ C(b) = 1,\ C(c) = 20,\ C(d) = 21,\ C(e) = 22\}$. A palavra de código $C(e) = 22$ pode ser descartada, pois a sua probabilidade de utilização é zero. O código restante é claramente óptimo.

Resta estudar qual o número de palavras que é necessário acrescentar a um alfabeto com N símbolos para se obter um código D-ário óptimo com o algoritmo de Huffman. Em cada passo do algoritmo de Huffman produz-se um alfabeto reduzido com $D - 1$ símbolos a menos do que o alfabeto anterior. Assim, é necessário que o número inicial de símbolos seja da forma $1 + k\,(D - 1)$, onde k é o número de níveis da árvore. Por exemplo, no caso ternário, $D - 1 = 2$, pelo que o número de símbolos deve ser da forma $1 + k2$, ou seja, um número ímpar. Considerando outro exemplo, para um alfabeto decimal, $D = 10$, $D - 1 = 9$, pelo que o número de símbolos deve ser da forma $1 + k9$, ou seja, pertencer a $\{10,\ 19,\ 28,\ 37,\ ...\}$. Obviamente, no caso binário, $D - 1 = 1$, e qualquer número maior que um se pode escrever como $1 + k$, ou seja, é possível construir árvores binárias completas com qualquer número de folhas.

2.8 Codificação de Shannon-Fano-Elias

A codificação aqui designada como de Shannon-Fano-Elias (SFE), de acordo com a designação adoptada em [4], é um híbrido de várias propostas apresentadas separadamente por Shannon,

Fano e Elias nas décadas de 1940 e 1950. O seu interesse presente é quase exclusivamente histórico, por estar na raiz da codificação aritmética, apresentada na secção seguinte. A descrição aqui apresentada segue de perto a referência [4], com pequenas alterações de notação e alguns detalhes adicionais.

Como visto na Subsecção 2.5, para que o comprimento médio de codificação se aproxime do limite teórico inferior imposto pela entropia da fonte, pode ser necessário recorrer a extensões de ordem elevada. Esta opção, no entanto, pode tornar-se pouco prática pelo seguinte motivo: o alfabeto para uma extensão de ordem n de uma fonte com um alfabeto original de N símbolos possui N^n símbolos. Se se adoptar codificação de Huffman para os símbolos extendidos, é necessário desenhar um código de Huffman para um alfabeto com N^n símbolos, isto é, que cresce exponencialmente com a ordem da extensão da fonte. Esta via torna-se rapidamente impraticável, mesmo para extensões de ordem moderada; por exemplo, se o alfabeto original for constituído pelos 256 símbolos ASCII, o alfabeto da extensão de ordem 3 possui mais de 16 milhões de símbolos ($256^3 = 2^{24} = 16777216$). Seria pois necessário desenhar um código de Huffman para um alfabeto com este enorme número de símbolos, a maioria dos quais acabariam mesmo por nunca ser utilizados.

A codificação de SFE (e a codificação aritmética, como se verá mais adiante) constitui uma alternativa, quase óptima (o sentido desta afirmação ficará claro mais adiante), ao uso de códigos de Huffman para fontes extendidas. A característica fundamental da codificação de SFE é a possibilidade de obter a palavra de código para um único símbolo, sem necessidade de criar palavras de código para todos os símbolos do alfabeto. É claro que a codificação de Huffman não possui esta característica.

Para descrever o procedimento de codificação de SFE, considere-se um alfabeto $\mathcal{X} = \{x_1, ..., x_N\}$, cujos símbolos são emitidos com probabilidades $\{p_1, ..., p_N\}$. Sem perda de generalidade, assume-se que $p_i > 0$, para $i = 1, ...N$; se existir algum $p_i = 0$, pode simplesmente retirar-se o respectivo símbolo do alfabeto pois nunca vai ser necessário codificá-lo. Focar-se-á o caso dos códigos binários, $C : \mathcal{X} \to \mathcal{D} = \{0, 1\}$, mas a generalização para outras alfabetos de código é trivial. Considere-se a função de distribuição cumulativa $F(x)$ que se define como

$$F(x_i) = F_i = \sum_{j=1}^{i} p_j, \quad \text{para } i = 1, 2, ..., N, \tag{2.22}$$

e $F_0 = 0$. Dado que todos os p_i são estritamente positivos, a sequência F_0, F_1, F_2, ..., F_N é estritamente monotónica, isto é, $F_0 < F_1 < F_2 < ... < F_N$. Note-se ainda que $F_1 = p_1$ e $F_N = 1$.

Dado que a sequência $F_1,\ F_2,\ ..., F_N$ é estritamente monotónica, todos os F_i são diferentes, isto é,

$$(i \neq j) \Rightarrow (F_i \neq F_j). \tag{2.23}$$

Observando a equação (2.3) na Subsecção 2.1.2, verifica-se que isto é precisamente a definição de código não singular. Esta observação sugere que se utilizem os números $F_1,\ F_2,\ ..., F_N$ para codificar os símbolos $\{x_1,\ x_2,\ ...,\ x_N\}$. No entanto, uma questão se levanta: em geral, os números $F_1,\ F_2,\ ..., F_N$ são reais arbitrários, no intervalo $]0, 1]$, pelo que a sua escrita (em base 2, ou em qualquer outra base) pode exigir um número infinito de dígitos. Para se

obter um código útil, é necessário truncar as representações dos números $F_1,\ F_2,\ ..., F_N$ para comprimentos finitos; ao fazer essa truncatura, pode exigir-se que o código obtido, mais do que simplesmente não singular, seja instantâneo.

No procedimento de SFE não se utilizam os números $F_1,\ F_2,\ ..., F_N$, mas sim um outro conjunto de números, com estes relacionados. Considere-se que a cada símbolo x_i se faz corresponder o intervalo $[F_{i-1},\ F_i[$ (fechado à esquerda e aberto à direita) cuja largura é p_i (pois, como é óbvio de (2.22), $F_i - F_{i-1} = p_i$). Considerem-se agora os pontos centrais de cada um destes intervalos, que serão designados como $\overline{F}_i$, para $i = 1, 2, ..., N$; dado que a largura do i-ésimo intervalo é p_i, tem-se que

$$\begin{aligned} \overline{F}_i &= F_{i-1} + \frac{p_i}{2} \\ &= F_i - p_i + \frac{p_i}{2} \\ &= F_i - \frac{p_i}{2}. \end{aligned} \tag{2.24}$$

Dado que todos os p_i são estritamente positivos,

$$\begin{aligned} \overline{F}_{i+1} - \overline{F}_i &= F_i + \frac{p_{i+1}}{2} - F_{i-1} - \frac{p_i}{2} \\ &= \underbrace{F_i - F_{i-1}}_{p_i} - \frac{p_i}{2} + \frac{p_{i+1}}{2} \\ &= \frac{p_i + p_{i+1}}{2} > 0. \end{aligned} \tag{2.25}$$

$$\tag{2.26}$$

Assim, a sequência $\overline{F}_1,\ \overline{F}_2,\ ...,\ \overline{F}_N$ também é estritamente monotónica e, como tal, também verifica a propriedade de não singularidade (2.23), podendo ser usada para construir um código para os símbolos $\{x_1,\ x_2, ..., x_N\}$. Resta encontrar uma forma de truncar as representações dos números $\overline{F}_1,\ \overline{F}_2,\ ...,\ \overline{F}_N$ que não destrua a propriedade de não singularidade e que, adicionalmente, corresponda a um código instantâneo.

A resposta à questão de como truncar as representações dos números $\overline{F}_1,\ \overline{F}_2,\ ...,\ \overline{F}_N$ por forma a obter um código instantâneo é simples. Tome-se

$$l_i = \lceil -\log_2 p_i \rceil + 1 \tag{2.27}$$

como comprimento da palavra de código $C(x_i)$. As palavras de código são dadas simplesmente por

$$C(x_i) = \text{primeiros } l_i \text{ digitos binários de } \overline{F}_i. \tag{2.28}$$

Para obter uma escrita mais formal, considere-se a notação

$$\lfloor a \rfloor_l = 2^{-l} \lfloor 2^l a \rfloor \tag{2.29}$$

que representa o número a truncado para possuir apenas l dígitos fraccionários [2]. Assim, pode

[2] Por exemplo, no caso decimal, $\lfloor \pi \rfloor_2 = 3.14$ e $\lfloor 1.28389 \rfloor_3 = 1.283$. Ainda no caso decimal, observe-se que, de facto, $\lfloor a \rfloor_l = 10^{-l} \lfloor 10^l a \rfloor$; por exemplo, $10^{-2} \lfloor 10^2\ 41.48597 \rfloor = 10^{-2} \lfloor 4148.597 \rfloor = 10^{-2} 4148 = 41.48 = \lfloor 41.48597 \rfloor_2$.
Numa base arbitrária b (por exemplo 2), a definição (2.29) generaliza-se para $\lfloor a \rfloor_l = b^{-l} \lfloor b^l a \rfloor$. Por exemplo, $\lfloor 0.1001001010 \rfloor_4 = 0.1001$.

escrever-se

$$C(x_i) = \text{digitos de } \lfloor \overline{F}_i \rfloor_{l_i}. \tag{2.30}$$

Recorde-se que todos os números $\overline{F}_i$ são menores que 1, pelo que todos possuem escritas da forma $\overline{F}_i = 0.d_1\, d_2 \cdots$, possivelmente com um número infinito de dígitos; naturalmente, na palavra de código não se inclui o zero à esquerda do ponto, pelo que $C(x_i) = d_1\, d_2 \cdots d_{l_i}$.

Antes de demonstrar que esta escolha conduz, de facto, a um código instantâneo, apresentam-se de seguida dois exemplos: um para um código decimal e outro para um código binário.

Exemplo 2.11 *Considere-se uma fonte que gera símbolos do alfabeto* $\mathcal{X} = \{x_1, x_2, ..., x_{10}, x_{11}\}$ *com probabilidades* $\{1/2, 1/4, 1/8, 1/16, 1/32, 1/64, 1/128, 1/256, 1/512, 1/1024, 1/1024\}$. *Na tabela seguinte, apresentam-se os valores de* $\overline{F}_i$, *os comprimentos* l_i *e as correspondentes palavras do código de SFE decimal. Note-se que, contrariamente ao que se passa com a codificação de Huffman, o código de SFE para uma dada fonte é único.*

i	p_i	$-\log_{10} p_i$	l_i	$\overline{F}_i$	$C(x_i)$
1	*0.50000000000*	*0.3010*	*2*	*0.250000000000*	*25*
2	*0.25000000000*	*0.6020*	*2*	*0.625000000000*	*62*
3	*0.12500000000*	*0.9030*	*2*	*0.812500000000*	*81*
4	*0.06250000000*	*1.2041*	*3*	*0.906250000000*	*906*
5	*0.03125000000*	*1.5051*	*3*	*0.953125000000*	*953*
6	*0.01562500000*	*1.8061*	*3*	*0.976562500000*	*976*
7	*0.00781250000*	*2.1072*	*4*	*0.988281250000*	*9882*
8	*0.00390625000*	*2.4082*	*4*	*0.994140625000*	*9941*
9	*0.00195312500*	*2.7092*	*4*	*0.997070312500*	*9970*
10	*0.00097656250*	*3.0102*	*5*	*0.998535156250*	*99853*
11	*0.00097656250*	*3.0102*	*5*	*0.999511718750*	*99951*

O comprimento médio deste código é de 2.14257 *dígitos decimais por símbolo, enquanto que a entropia de base 10 da fonte é* 0.60147 *dígitos decimais por símbolo.*

Exemplo 2.12 *Na tabela que se segue, repete-se o exemplo anterior, agora para codificação de SFE binária. Na sexta coluna da tabela,* $(\overline{F}_i)_{(2)}$ *representa o valor de* $\overline{F}_i$ *escrito em representação de base 2.*

i	p_i	$-\log_2 p_i$	l_i	$\overline{F}_i$	$(\overline{F}_i)_{(2)}$	$C(x_i)$
1	*0.50000000000*	*1*	*2*	*0.250000000000*	*0.0100000000000*	*01*
2	*0.25000000000*	*2*	*3*	*0.625000000000*	*0.1010000000000*	*101*
3	*0.12500000000*	*3*	*4*	*0.812500000000*	*0.1101000000000*	*1101*
4	*0.06250000000*	*4*	*5*	*0.906250000000*	*0.1110100000000*	*11101*
5	*0.03125000000*	*5*	*6*	*0.953125000000*	*0.1111010000000*	*111101*
6	*0.01562500000*	*6*	*7*	*0.976562500000*	*0.1111101000000*	*1111101*
7	*0.00781250000*	*7*	*8*	*0.988281250000*	*0.1111110100000*	*11111101*
8	*0.00390625000*	*8*	*9*	*0.994140625000*	*0.1111111010000*	*111111101*
9	*0.00195312500*	*9*	*10*	*0.997070312500*	*0.1111111101000*	*1111111101*
10	*0.00097656250*	*10*	*11*	*0.998535156250*	*0.1111111110100*	*11111111101*
11	*0.00097656250*	*10*	*11*	*0.999511718750*	*0.1111111111100*	*11111111111*

O comprimento médio deste código é de 2.9980 *bits por símbolo, enquanto que a entropia binária da fonte é* 1.9980 *bits por símbolo.*

A título de comparação, apresenta-se na tabela seguinte um código binário óptimo (de Huffman) para esta fonte, cujo comprimento médio é 1.9980 *bits/símbolo, precisamente igual à entropia, pois a distribuição de probabilidadades da fonte é diádica (todas as probabilidades são potências de 2).*

i	p_i	$C^{opt}(x_i)$
1	*0.50000000000*	*0*
2	*0.25000000000*	*10*
3	*0.12500000000*	*110*
4	*0.06250000000*	*1110*
5	*0.03125000000*	*11110*
6	*0.01562500000*	*111110*
7	*0.00781250000*	*1111110*
8	*0.00390625000*	*11111110*
9	*0.00195312500*	*111111110*
10	*0.00097656250*	*1111111110*
11	*0.00097656250*	*1111111111*

Finalmente, apresenta-se a demonstração de que a escolha

$$C(x_i) = \text{digitos de } \lfloor \overline{F}_i \rfloor_{l_i}. \tag{2.31}$$

com

$$l_i = \lceil -\log_2 p_i \rceil + 1 \tag{2.32}$$

conduz, de facto, a um código instantâneo. Para tal, usa-se a relação entre palavras de código e sub-intervalos de $[0,1[$ (anteriormente usado na Secção 2.2, na Demonstração 2). No caso presente, o intervalo associado à palavra de código $C(x_i)$ é

$$I(x_i) = \left[\lfloor \overline{F}_i \rfloor_{l_i},\ \lfloor \overline{F}_i \rfloor_{l_i} + 2^{-l_i} \right[,$$

pois $C(x_i)$ tem l_i dígitos (bits). Para que o código seja instantâneo, é necessário e suficiente que todos os intervalos $I(x_i),\ I(x_2),\ ..., I(x_N)$ sejam disjuntos. Para confirmar que os intervalos são, de facto, disjuntos, considerem-se dois intervalos genéricos consecutivos: $I(x_i)$ e $I(x_{i+1})$. Estes intervalos são disjuntos se o limite direito de $I(x_i)$ (isto é, $\lfloor \overline{F}_i \rfloor_{l_i} + 2^{-l_i}$) for estritamente menor do que o limite esquerdo de $I(x_{i+1})$ (isto é, $\lfloor \overline{F}_{i+1} \rfloor_{l_{i+1}}$), ou seja, é necessário mostrar que

$$\lfloor \overline{F}_i \rfloor_{l_i} + 2^{-l_i} - \lfloor \overline{F}_{i+1} \rfloor_{l_{i+1}} < 0. \tag{2.33}$$

Considerando a primeira parcela de (2.33), pode verificar-se que

$$\begin{aligned}
\lfloor \overline{F}_i \rfloor_{l_i} + 2^{-l_i} &\leq \overline{F}_i + 2^{-l_i} && (2.34)\\
&= \overline{F}_i + 2^{-(\lceil -\log_2 p_i \rceil + 1)} && (2.35)\\
&= \overline{F}_i + \frac{1}{2}\, 2^{-\lceil -\log_2 p_i \rceil} && (2.36)\\
&\leq \overline{F}_i + \frac{1}{2}\, 2^{-(-\log_2 p_i)} && (2.37)\\
&= \overline{F}_i + \frac{p_i}{2}, && (2.38)
\end{aligned}$$

onde a desigualdade em (2.34) resulta de, para qualquer número não negativo a e qualquer l, se verificar $\lfloor a \rfloor_l \leq a$; a desigualdade em (2.37) resulta de, para qualquer número não negativo a, se verificar $\lceil a \rceil \geq a$ e, consequentemente, $2^{-\lceil a \rceil} \leq 2^{-a}$. Observando agora o termo da direita em (2.33), constata-se que

$$\begin{aligned}
\lfloor \overline{F}_{i+1} \rfloor_{l_{i+1}} &> \overline{F}_{i+1} - 2^{-l_{i+1}} && (2.39)\\
&\geq \overline{F}_{i+1} - \frac{p_{i+1}}{2}, && (2.40)
\end{aligned}$$

onde a desigualdade em (2.39) resulta de, para qualquer número não negativo a e qualquer l, se verificar[3] $\lfloor a \rfloor_l > a - 2^{-l}$. Finalmente, combinando (2.38) e (2.40), pode escrever-se (recorde-se de (2.25) que $\overline{F}_{i+1} - \overline{F}_i = (p_i + p_{i+1})/2$)

$$\lfloor \overline{F}_i \rfloor_{l_i} + 2^{-l_i} - \lfloor \overline{F}_{i+1} \rfloor_{l_{i+1}} < \overline{F}_i + \frac{p_i}{2} - \overline{F}_{i+1} + \frac{p_{i+1}}{2} = \underbrace{\overline{F}_i - \overline{F}_{i+1}}_{-(p_i+p_{i+1})/2} + \frac{p_i + p_{i+1}}{2} = 0, \tag{2.41}$$

o que demonstra (2.33), condição suficiente para o código de SFE seja instantâneo, como se pretendia demonstrar.

[3] Veja-se, no caso decimal, que $\lfloor a \rfloor_l > a - 10^{-l}$; por exemplo, a diferença entre qualquer número da forma "$0.314 * **$" e 0.314 é menor que 0.001.

Finalmente, pode verificar-se que o código de SFE, que se designará como C^{sfe} é sub-óptimo. De facto, devido ao uso dos comprimentos $l_i^{\text{sfe}} = \lceil -\log p_i \rceil + 1$, em vez dos comprimentos de Shannon $l_i^s = \lceil -\log p_i \rceil$, o comprimento médio dos códigos de SFE verifica uma desigualdade semelhante a (2.16), mas com um excesso de 1 bit/símbolo:

$$H(X) + 1 \leq L(C^{\text{sfe}}) < H(X) + 2. \tag{2.42}$$

No caso de se utilizarem extensões de ordem n, com n elevado (que é a motivação principal para o use de codificação de SFE), pode facilmente verificar-se um par de desigualdades semelhante a (2.19):

$$H(X) + \frac{1}{n} \leq L_n(C_n^{\text{sfe}}) < H(X) + \frac{2}{n}. \tag{2.43}$$

Em conclusão, usando extensões de ordem arbitrariamente elevada, o comprimento médio do código de Shannon-Fano-Elias aproxima-se arbitrariamente do valor da entropia da fonte. É neste sentido que se deve entender a afirmação avançada no início desta secção de que a codificação de SFE é quase óptima.

2.9 Codificação Aritmética

A codificação aritmética[4] pode ser vista simplesmente como uma forma eficiente de implementar a codificação de Shannon-Fano-Elias, no caso de extensões de ordem elevada de fontes sem memória.

Considere-se uma fonte sem memória, emitindo símbolos do alfabeto $\mathcal{X}$, com probabilidades $\{p(x),\ x \in \mathcal{X}\}$. Pretende codificar-se uma sequência de n símbolos gerados por esta fonte, $(x_1,\ x_2, ...,\ x_n)$, usando codificação de Shannon-Fano-Elias. Para tal, como descrito na secção anterior, basta calcular $p(x_1,\ x_2, ...,\ x_n)$ (ou seja p_i, na notação da secção anterior) e $F(x_1,\ x_2, ...,\ x_n)$ (ou seja, F_i, na notação da secção anterior). Com F_i e p_i, pode calcular-se $\overline{F}_i$, usando (2.24) e $l_i = \lceil -\log p_i \rceil + 1$; conhecidos $\overline{F}_i$ e l_i, pode obter-se a palavra de código de SFE através de (2.30). No caso de uma fonte sem memória, $p(x_1,\ x_2, ...,\ x_n) = p(x_1)\, p(x_2) \cdots p(x_n)$.

Note-se que na definição de $F(x_1,\ x_2, ...,\ x_n)$ está implícita uma ordenação de conjuntos de símbolos do alfabeto, pois, por definição (ver (2.22))

$$F(x_1,\ ...,\ x_n) = \sum_{(x_1', ..., x_n') \leq (x_1,\ ...,\ x_n)} p(x_1', ...,\ x_n').$$

A relação de ordem usada em $(x_1', ...,\ x_n') \leq (x_1,\ ...,\ x_n)$ é a chamada ordem lexicográfica; esta, por sua vez, suporta-se numa relação de ordem para os símbolos do alfabeto, ou seja, o alfabeto $\mathcal{X}$ deve ser visto como um conjunto ordenado de símbolos, para os quais a relação $a < b$ significa que a surge primeiro no alfabeto do que b. A ordem lexicográfica é uma generalização da habitual ordem alfabética, definida por

$$\left[(x_1', ...,\ x_n') < (x_1,\ ...,\ x_n)\right] \Leftrightarrow \exists_{m \in \{1,...,n-1\}} : \begin{cases} x_i' = x_i \Leftarrow i < m \\ x_m' < x_m \end{cases}$$

[4]Para uma introdução detalhada, mas acessível, à codificação aritmética, bem como uma breve história da sua origem, veja-se o artigo de Glen Langdon, "An introduction to arithmetic coding", disponível em `http://www.research.ibm.com/journal/rd/282/ibmrd2802C.pdf`

Por outras palavras, uma sequência de símbolos $(x'_1, ..., x'_n)$ é dita "menor"que (ou que está à esquerda de) um outra $(x_1, ..., x_n)$ se forem idênticas até uma determinada posição $m-1$ e, na primeira posição, m, em que diferem, o símbolo x'_m for "menor"(na ordem do alfabeto) do que x_m; a ordem entre os símbolos seguintes é irrelevante.

Exemplo 2.13 *Um pequeno exemplo ajudará a tornar o conceito mais claro. Considere-se o alfabeto $\mathcal{X} = \{a, b, c, d\}$, com a ordem implícita $a < b < c < d$. A sequência $(ababcbcbda)$ é "menor"que $(abadaadccb)$ pois os 3 primeiros símbolos são iguais, (aba); no primeiro símbolo em que as duas sequências diferem (o quarto), o símbolo da primeira sequência, b, é "menor"do que o símbolo na mesma posição na segunda sequência, d.*

A ideia central da codificação aritmética é de que é possível calcular $F(x_1, x_2, ..., x_n)$ e $p(x_1, x_2, ..., x_n)$ através de um procedimento (aritmético) simples de sucessivas partições de subintervalos de $[0, 1[$. Para simplificar a escrita formal do algoritmo, considera-se que o alfabeto é $\{1, 2, ..., N\}$, sem qualquer perda de generalidade, dado que se tinha já assumido que o alfabeto estava "ordenado". O algoritmo de codificação aritmética procede do seguinte modo:

Dados: O alfabeto $\mathcal{X} = \{1, 2, ..., N\}$, as probabilidades dos símbolos $\{p(1), p(2), ..., p(N)\}$, e uma sequência $(x_1, x_2, ..., x_n)$ de n símbolos de $\mathcal{X}$, a codificar.

Inicialização: Tome-se $t = 1$ e considere-se o intervalo $[L, R[$, com $L = 0$ e $R = 1$.

Passo 1: Parte-se o intervalo actual $[L, R[$ em N subintervalos (todos fechados à esquerda e abertos à direita) com tamanhos proporcionais às probabilidades dos símbolos do alfabeto: $(R-L)p(1)$, $(R-L)p(2)$, ..., $(R-L)p(N)$. Os intervalos obtidos são

$$\begin{aligned}
I_1 &= [L, L + (R-L)p(1)[, \\
I_2 &= [L + (R-L)p(1), L + (R-L)p(1) + (R-L)p(2)[\\
&\vdots \quad \vdots \quad \vdots \\
I_i &= [L + (R-L)(p(1) + \cdots + p(i-1)), L + (R-L)(p(1) + \cdots + p(i))[\\
&\vdots \quad \vdots \quad \vdots \\
I_N &= [L + (R-L)(p(1) + \cdots + p(N-1)), R[
\end{aligned}$$

Note-se que os N subintervalos são definidos por $N+1$ pontos, pois o limite esquerdo de um subintervalo é igual ao limite direito do subintervalo seguinte. Assim, se se convencionar que $p(0) = 0$, pode escrever-se uma expressão genérica para o subintervalo I_i, da forma $I_i = [L_i, R_i[= [T_i, T_{i+1}[$, com

$$T_i = L + (R-L)\sum_{j=0}^{i-1} p(j).$$

Confirme-se que a largura de I_i é de facto $(R-L)p(i)$:

$$|I_i| = T_{i+1} - T_i = R_i - L_i = L + (R-L)\sum_{j=0}^{i} p(j) - L - (R-L)\sum_{j=0}^{i-1} p(j) = (R-L)p(i).$$

Passo 2: Escolhe-se o x_t-ésimo intervalo, isto é, faz-se $L = L_{x_t}$ e $R = R_{x_t}$.

Passo 3: Se $t < n$ (ainda há símbolos para codificar), faz-se $t = t + 1$ e volta-se ao passo 1.

É imediato verificar que, após a conclusão do algoritmo, a largura do intervalo final $[L,\, R[$ é precisamente $(R - L) = p(x_1,\, x_n,\, ...,\, x_n) = p(x_1)\, p(x_2) \cdots p(x_n)$; de facto, a largura do intervalo inicial $[0,\, 1[$ é 1, do segundo intervalo é $p(x_1)$, do terceiro intervalo é $p(x_1)\, p(x_2)$, e assim sucessivamente. Consequentemente, o limite esquerdo do intervalo final, L, é a soma das probabilidades de todas as sequências "menores ou iguais" a $(x_1,\, x_n,\, ...,\, x_n)$, isto é, $F(x_1,\, x_n,\, ...,\, x_n)$. Para obter o código, basta agora calcular a expansão binária de $\overline{F}(x_1,\, x_n,\, ...,\, x_n) = F(x_1,\, x_n,\, ...,\, x_n) - p(x_1,\, x_n,\, ...,\, x_n)/2$ e usar os primeiros $l = \lceil -\log p(x_1,\, x_n,\, ...,\, x_n) \rceil + 1$ bits dessa expansão.

Exemplo 2.14 *Considere-se uma fonte sem memória com alfabeto* $\{1,\, 2,\, 3,\, 4\}$*; as probabilidades dos símbolos são, respectivamente,* $\{0.4,\, 0.35,\, 0.15,\, 0.1\}$*. Pretende obter-se a palavra de código aritmético binário para sequência* $(1,\, 1,\, 2,\, 1,\, 4,\, 3,\, 2)$*. A probabilidade desta sequência é* $p(1)p(1)p(2)p(1)p(4)p(3)p(2) = 0.4^3 \cdot 0.35^2 \cdot 0.15 \cdot 0.1 = 0.0001176$*. Inserindo esse valor em* $l = \lceil -\log_2 p(x_1,\, x_n,\, ...,\, x_n) \rceil + 1$*, pode desde já afirmar-se que a palavra de código possui 15 bits.*

Dado que o alfabeto possui 4 símbolos, em cada iteração o intervalo actual $[L,\, R[$ *é subdividido em 4 subintervalos* I_1*,* I_2*,* I_3 *e* I_4*, definidos por 5 pontos* $T_1 = L$*,* T_2*,* T_3*,* T_4 *e* $T_5 = R$*. Na tabela seguinte, apresenta-se a evolução do algoritmo, listando-se os sucessivos valores dos pontos que delimitam os subintervalos, o subintervalo escolhido por cada símbolo e a respectiva largura.*

t	$[L,\, R[$	T_1	T_2	T_3	T_4	T_5	x_t	*novo* $[L,\, R[$	$R - L$
1	[0, 1[	0	.4	.75	.85	1.0	1	[0, .4[	.4
2	[0, .4[	0	.16	.3	.36	0.4	1	[0, .16[	.16
3	[0, .16[	0	.064	.12	.144	.16	2	[.064, .12[	.056
4	[.064, .12[	.064	.0864	.106	.1144	.12	1	[.064, .0864[	.0224
5	[.064, .0864[	.064	.07296	.0808	.08461	.0864	4	[.08461, .0864[	.00224
6	[.08461, .0864[	.08461	.085056	.08584	.086176	.0864	3	[.08584, .086176[	.000336
7	[.08584, .086176[	.08584	.0859744	.086092	.0861424	.086176	2	[.0859744, .086092[	0.0001176

Observe-se que a largura do intervalo final é de facto a probabilidade da sequência, 0.0001176. Quanto a $\overline{F}$*, é simplesmente dado pelo ponto central do intervalo final:*

$$\overline{F}(1,\, 1,\, 2,\, 1,\, 4,\, 3,\, 2) = .0859744 + \frac{0.0001176}{2} = 0.0860332.$$

Em base 2, este número escreve-se

$$\overline{F}(1,\, 1,\, 2,\, 1,\, 4,\, 3,\, 2) = 0.0860332_{(10)} = 0.00010110000000110010001..._{(2)},$$

pelo que a palavra de código se obtém tomando os primeiros 15 dígitos,

$$C(1,\, 1,\, 2,\, 1,\, 4,\, 3,\, 2) = 000101100000011.$$

Note-se que, para obter a palavra de Huffman para esta sequência, seria necessário desenhar um código de Huffman para um alfabeto extendido com $4^7 = 16384$ símbolos.

Capítulo 3

Fontes Discretas com Memória

3.1 Processos Estocásticos Discretos em Tempo Discreto

O Capítulo 1 foi dedicado à introdução dos conceitos de teoria da informação associados à fontes discretas sem memória, isto é, que podem ser descritas como variáveis aleatórias gerando símbolos de forma independente uns dos outros. Para estudar fontes nas quais os símbolos emitidos em instantes diferentes não são mutuamente independentes, é necessário usar um modelo formal mais geral. Assim, uma fonte discreta com memória deve ser descrita como um processo estocástico discreto (dado que se consideram apenas fontes emitindo símbolos de alfabetos discretos) em tempo discreto. Um processo estocástico discreto em tempo discreto não é mais do que uma sequência de variáveis aleatórias,

$$\mathbf{X} = \{X_1, X_2, ..., X_t, ...\}, \tag{3.1}$$

com $X_t \in \mathcal{X}_t$, em que $\mathcal{X}_t$ é o conjunto de símbolos que a fonte pode emitir no instante t. Por simplicidade, geralmente considera-se que $\mathcal{X}_t = \mathcal{X}$, isto é, que o conjunto de símbolos possíveis não varia de instante para instante. Como já foi feito anteriormente, e sem perda de generalidade, neste capítulo adopta-se a convenção de se associar os elementos do alfabeto aos números de 1 a N, isto é, $\mathcal{X} = \{1, ..., N\}$.

Podem também considerar-se processos estocásticos definidos desde um passado infinitamente remoto, isto é,

$$\mathbf{X} = \{..., X_{-1}, X_0, X_1, ..., X_t, ...\}, \tag{3.2}$$

ou processos estocásticos definidos em intervalos de tempo (discreto) de duração finita,

$$\mathbf{X} = \{X_1, ..., X_t, ..., X_T\}. \tag{3.3}$$

Enquanto que a caracterização de uma variável aleatória fica completa com o conhecimento da probabilidade de cada elemento de $\mathcal{X}$ (ver Secção 1.1), o caso dos processos estocásticos é bastante mais complexo. A caracterização completa de um processo estocástico exige o conhecimento da função de probabilidade conjunta de qualquer sub-conjunto finito das variáveis aleatórias que o compõem; isto é, para qualquer inteiro $K \geq 0$ e para qualquer conjunto de K

instantes $\{t_1, ..., t_K\}$, deve conhecer-se

$$P\left(X_{t_1} = x_1, ..., X_{t_K} = x_K\right), \tag{3.4}$$

para todas as possíveis sequências $(x_1, ..., x_K) \in \mathcal{X}^K$.

No caso de um processo definido num intervalo finito (ver (3.3)) com T instantes, se nenhuma estrutura particular for assumida, a caracterização completa exige o conhecimento de $N^T - 1$ probabilidades, pois o conjunto de todas as possíveis sequências de T símbolos de um alfabeto de dimensão N é N^T. No caso de processos com um conjunto infinito de instantes, a caracterização completa, na ausência de qualquer estrutura adicional, exige o conhecimento de um número infinito de probabilidades, pelo que não é aplicável na prática. Assim, o estudo do processos estocásticos concentra-se geralmente em processos com alguma estrutura temporal mais particular, como apresentado nos parágrafos seguintes.

3.2 Processos Estacionários

Um processo diz-se estacionário se e só se verificar a seguinte condição: para qualquer inteiro $K \geq 0$, para qualquer conjunto de K instantes $\{t_1, ..., t_K\}$, e para qualquer inteiro k,

$$P\left(X_{t_1} = x_1, ..., X_{t_K} = x_K\right) = P\left(X_{t_1+k} = x_1, ..., X_{t_K+k} = x_K\right), \tag{3.5}$$

para todas as possíveis sequências $(x_1, ..., x_K) \in \mathcal{X}^K$. Isto é, a probabilidade de se observar um determinado padrão de símbolos num determinado conjunto de instantes, não depende de forma absoluta da localização temporal desses instantes, mas apenas das suas localizações relativas. Embora tenha uma estrutura claramente mais simples, a caracterização completa de um processo estacionário definido num intervalo de tempo infinito continua a exigir o conhecimento de um número infinito de probabilidades

3.3 Processos de Markov

3.3.1 Introdução

O conceito de processo de Markov vai finalmente permitir caracterizar processos definidos num intervalo de tempo infinito com um conjunto finito de probabilidades. Por este motivo, esta classe de processos é extremamente utilizada em muitas áreas técnicas e científicas, tais o controlo, o processamento de sinais, a física, a economia, a biologia; ao estudante interessado num estudo mais aprofundado dos processos de Markov, sugere-se o excelente livro [3]. Os processos de Markov discretos em tempo discreto (os únicos focados neste texto, pois são os de interesse como modelos de fontes discretas com memória) são geralmente designados de cadeias de Markov (*Markov chains*).

Um processo $\mathbf{X} = \{X_1, X_2, ..., X_t, ...\}$ é dito de Markov (ou markoviano) de ordem n se verificar a seguinte propriedade:

$$P\left(X_t = x_t | X_{t-1} = x_{t-1}, X_{t-2} = x_{t-2}, ..., X_1 = x_1\right) = P\left(X_t = x_t | X_{t-1} = x_{t-1}, ..., X_{t-n} = x_{t-n}\right), \tag{3.6}$$

para qualquer sequência $(x_1, ..., x_t) \in \mathcal{X}^t$. Por palavras, um processo é markoviano de ordem n se a probabilidade do símbolo emitido num instante t, dado todo o passado, for apenas função de um passado recente de duração n. O caso particular de $n = 1$, dito processo markoviano de primeira ordem, no qual

$$P\left(X_t = x_t | X_{t-1} = x_{t-1}, X_{t-2} = x_{t-2}, ..., X_1 = x_1\right) = P\left(X_t = x_t | X_{t-1} = x_{t-1}\right), \tag{3.7}$$

é o mais clássico e estudado. Uma das razões para este facto é que, como se verá adiante, um processo de ordem n pode ser reescrito como um processo de primeira ordem.

O conjunto de instantes anteriores, dos quais depende a probabilidade do símbolo que vai ser emitido no instante seguinte, designa-se habitualmente como **estado** do processo (ou cadeia) de Markov. O estado de uma fonte markoviana de ordem n contem os n últimos símbolos emitidos.

3.3.2 Processos de Markov Invariantes no Tempo

Um processo de Markov de ordem n no qual as probabilidades não dependem explicitamente do instante de tempo, ou seja, para o qual, para qualquer t,

$$P\left(X_{n+1} = x_{n+1} | X_n = x_n, ..., X_1 = x_1\right) = P\left(X_t = x_{n+1} | X_{t-1} = x_n, ..., X_{t-n} = x_1\right), \tag{3.8}$$

qualquer que seja a sequência $(x_1, ..., x_{n+1}) \in \mathcal{X}^{(n+1)}$, designa-se um processo de Markov **invariante no tempo**.

Exemplo 3.1 *Considere-se um processo de Markov de ordem 3, definido no alfabeto* $\mathcal{X} = \{a, b, c\}$. *Se o processo for invariante no tempo, verifica-se que*

$$\begin{aligned} P(X_4 = a | X_3 = c,\, X_2 = b,\, X_1 = a) &= P(X_{34} = a | X_{33} = c,\, X_{32} = b,\, X_{31} = a) \\ &= P(X_{269} = a | X_{268} = c,\, X_{267} = b,\, X_{266} = a) \\ &= P(X_{k+4} = a | X_{k+3} = c,\, X_{k+2} = b,\, X_{k+1} = a), \end{aligned}$$

para qualquer valor de k. Ou seja, a probabilidade de a fonte emitir o símbolo "a" após ter emitido a sequência "abc" é a mesma em qualquer instante de tempo.

Um processo de Markov de ordem 1 invariante no tempo fica completamente caracterizado por um conjunto de $N \times N$ probabilidades (designadas **probabilidades de transição**)

$$P(X_2 = j | X_1 = i), \tag{3.9}$$

para $i, j \in \mathcal{X} = \{1, ..., N\}$, e pelas probabilidades iniciais $P(X_1 = k)$, para $k \in \mathcal{X}$. Habitualmente escreve-se este conjunto de probabilidades sob a forma de uma matriz $\mathbf{P}$, dita **matriz de transição**, cujo elemento (i, j) é dado por

$$P_{i,j} = P(X_2 = j | X_1 = i). \tag{3.10}$$

Esta matriz possui a propriedade de que todos os elementos pertencem ao intervalo [0, 1] (pois são probabilidades) e os elementos de qualquer linha somam 1,

$$\sum_{j=1}^{N} P_{i,j} = 1,$$

pois

$$\sum_{j=1}^{N} P_{i,j} = \sum_{j=1}^{N} P(X_2 = j | X_1 = i) = 1.$$

Uma matriz que verifica estas propriedades designa-se uma **matriz estocástica.** As probabilidades iniciais agrupam-se num vector $\mathbf{p}(1)$, cujos elementos são

$$\begin{bmatrix} p_1(1) \\ p_2(1) \\ \vdots \\ p_N(1) \end{bmatrix} = \begin{bmatrix} P(X_1 = 1) \\ P(X_1 = 2) \\ \vdots \\ P(X_1 = N) \end{bmatrix},$$

verificando, obviamente,

$$\sum_{i=1}^{N} p_i(1) = 1.$$

Qualquer função de probabilidade conjunta, para um conjunto de instantes consecutivos com $\{1, 2, ..., t\}$ se pode escrever com base nestas probabilidades. De facto, uma simples aplicação da lei de Bayes e da propriedade de markovianidade de ordem 1 (ver (3.7)) permite escrever

$$\begin{aligned} P(X_t = x_t, ..., X_1 = x_1) &= P(X_t = x_t | X_{t-1} = x_{t-1}, ..., X_1 = x_1)\, P(X_{t-1} = x_{t-1}, ..., X_1 = x_1) \\ &= P(X_t = x_t | X_{t-1} = x_{t-1})\, P(X_{t-1} = x_{t-1}, ..., X_1 = x_1). \end{aligned}$$

Repetindo o procedimento de modo recursivo, obtém-se

$$P(X_t = x_t, ..., X_1 = x_1) = p(X_1 = x_1) \prod_{u=2}^{t} P(X_u = x_u | X_{u-1} = x_{u-1}).$$

Finalmente, invocando a propriedade de invariância no tempo (3.8), tem-se $P(X_u = x_u | X_{u-1} = x_{u-1}) = P(X_2 = x_u | X_1 = x_{u-1}) = P_{x_{u-1}, x_u}$, e logo

$$P(X_t = x_t, ..., X_1 = x_1) = p(X_1 = x_1) \prod_{u=2}^{t} P(X_2 = x_u | X_1 = x_{u-1}) = p_{x_1}(1) \prod_{u=2}^{t} P_{x_{u-1}, x_u}.$$

Exemplo 3.2 *Considere-se um processo de Markov de primeira ordem, definido num alfabeto* $\mathcal{X} = \{1, 2, 3\}$*, com matriz de transição* $\mathbf{P}$*, de dimensão* 3×3*, e vector de probabilidades iniciais* $\mathbf{p}(1)$*, de dimensão* 3×1*. A probabilidade de se observar a sequência* (3, 1, 3, 3, 2, 1)*, a partir do instante* 1*, é*

$$P(X_1 = 3, X_2 = 1, X_3 = 3, X_4 = 3, X_5 = 2, X_6 = 1) = p_3(1)\, P_{3,1}\, P_{1,3}\, P_{3,3}\, P_{3,2}\, P_{2,1}.$$

Para escrever a probabilidade conjunta para um conjunto de instantes não consecutivos $\{t_1, t_2, ..., t_K\}$, basta calcular $P(X_{t_K} = x_{t_K}, ..., X_1 = x_1)$ (isto é, a probabilidade conjunta para todos os instantes de 1 a t_K) e em seguida marginalizar em relação aos instantes que não surgem em $\{t_1, t_2, ..., t_K\}$. O exemplo que se segue ilustra esta ideia.

Exemplo 3.3 *Considere-se um processo de Markov de primeira ordem, definido num alfabeto* $\mathcal{X} = \{1, 2, 3\}$*, com matriz de transição* $\mathbf{P}$*, de dimensão* 3×3*, e vector de probabilidades iniciais* $\mathbf{p}(1)$*, de dimensão* 3×1*. Pretende calcular-se a probabilidade de se observar um* 1 *no instante* 4 *e um* 3 *no instante* 6*; esta probabilidade é dada por*

$$\begin{aligned} P(X_4{=}\,1, X_6 = 3) &= \sum_{x_1}\sum_{x_2}\sum_{x_3}\sum_{x_5} P(X_1 = x_1, X_2 = x_2, X_3 = x_3, X_4 = 1, X_5 = x_5, X_6 = 3) \\ &= \sum_{x_1}\sum_{x_2}\sum_{x_3}\sum_{x_5} p_{x_1}(1)\; P_{x_1,x_2}\; P_{x_2,x_3}\; P_{x_3,1}\; P_{1,x_5}\; P_{x_5,3}, \end{aligned}$$

em que todas as somas se extendem, naturalmente, a todos os possíveis símbolos do alfabeto $\mathcal{X}$*.*

Pode então afirmar-se que, no caso de processos de Markov de ordem finita e invariantes no tempo, é possível escrever probabilidades conjuntas relativas a conjuntos arbitrariamente grandes de instantes, apenas com base num conjunto finito de probabilidades: a matriz de transição $\mathbf{P}$ e a distribuição inicial $\mathbf{p}(1)$.

É comum representar-se uma cadeia de Markov de ordem 1 com auxilio de um grafo, em que cada nó corresponde a um dos símbolos da fonte (que no caso de ordem 1, coincide com o estado da fonte). Entre cada par de nós, existe um arco dirigido, etiquetado com a probabilidade da respectiva transição. Habitualmente, omitem-se os arcos associados a probabilidades nulas. O exemplo seguinte ilustra a construção deste grafo para um caso simples.

Exemplo 3.4 *Considere-se uma fonte de Markov de primeira ordem, com quatro estados/símbolos* $\mathcal{X} = \{1, 2, 3, 4\}$*, cuja matriz de transição é*

$$\mathbf{P} = \begin{bmatrix} 0.1 & 0.3 & 0.4 & 0.2 \\ 0.45 & 0.2 & 0 & 0.35 \\ 0 & 0.2 & 0 & 0.8 \\ 0.5 & 0.5 & 0 & 0 \end{bmatrix}.$$

O grafo associado a este processo de Markov está representado na Figura 3.1.

Para fontes de Markov de ordem superior, $n > 1$, o estado contém os últimos n símbolos emitidos. Neste caso, a matriz de transição não é quadrada, mas sim de dimensão $N^n \times N$, isto é, possui uma linha por cada possível configuração do estado.

Exemplo 3.5 *Uma fonte markoviana de ordem 2, que emite símbolos de um alfabeto com 3 símbolos* $\mathcal{X} = \{1, 2, 3\}$*, possui uma matrix de transição de dimensão* 9×3*. Adoptando uma ordenação lexicográfica, o conjunto de estados possíveis é*

$$\{(1,1), (1,2), (1,3), (2,1), (2,2), (2,3), (3,1), (3,2), (3,3)\},$$

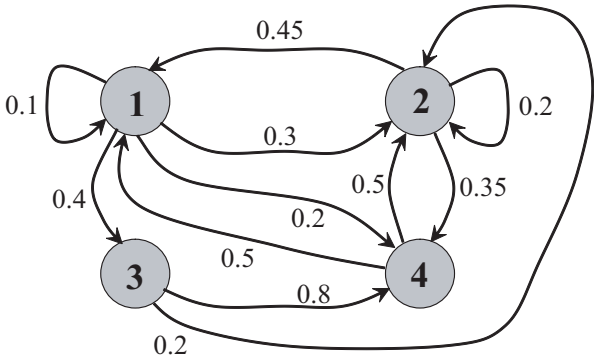

Figura 3.1: Grafo associado ao processo de Markov definido no Exemplo 3.4.

onde se assume que os símbolos de cada par estão ordenados por ordem cronológica. Assim, por exemplo,

$$P(X_t = 2|X_{t-2} = 3, X_{t-1} = 1) = P_{7,2},$$

pois a configuração $(3, 1)$ *surge na posição* 7 *na lista das configurações de estado possíveis.*

Com base no conjunto de estados possíveis, qualquer processo de Markov pode ser visto como um processo de primeira ordem, com algumas restrições nas transições possíveis. Este ideia é mais facilmente apresentada por através do exemplo que se segue.

Exemplo 3.6 *Considere-se uma fonte que emite símbolos do alfabeto* $\{1, 2\}$*, de acordo com um processo de ordem 2 definido pela seguinte matriz de transição:*

$$\mathbf{P} = \begin{bmatrix} 0.1 & 0.9 \\ 0.6 & 0.4 \\ 0.3 & 0.7 \\ 1.0 & 0.0 \end{bmatrix}.$$

Recordar que as linhas da matriz correspondem a uma ordenação lexicográfica do estado; por exemplo, nesta fonte, a probabilidade de se emitir um "1" após a sequência "1,2" é igual a 0.6; a probabilidade de se emitirem três símbolos "2" consecutivos é nula. Pode olhar-se para esta fonte como um processo de ordem 1 definido no conjunto de estados possíveis $\mathcal{X}^2 = \{(1, 1), (1, 2), (2, 1), (2, 2)\}$*, em que as transições de* $(1, 1)$ *para* $(2, 2)$ *e vice-versa são, por construção (e independentemente da matriz* $\mathbf{P}$*) impossíveis. Sendo o primeiro elemento do par* (x_1, x_2) *o penúltimo símbolo emitido, e o segundo elemento o último símbolo emitido, de* (x_1, x_2) *apenas se pode transitar para um par da forma* (x_2, x_3)*; ou seja, o último passa a ser o penúltimo e o lugar do último é tomado pelo novo símbolo. O grafo deste processo de ordem 1 definido em* $\mathcal{X}^2 = \{(1, 1), (1, 2), (2, 1), (2, 2)\}$*, equivalente ao processo de ordem 2 definido em* $\mathcal{X} = \{1, 2\}$ *com a matriz de transição* $\mathbf{P}$ *é apresentado na figura 3.2. A matriz de transição deste processo é*

$$\mathbf{P}' = \begin{bmatrix} 0.1 & 0.9 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.6 & 0.4 \\ 0.3 & 0.7 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 \end{bmatrix}.$$

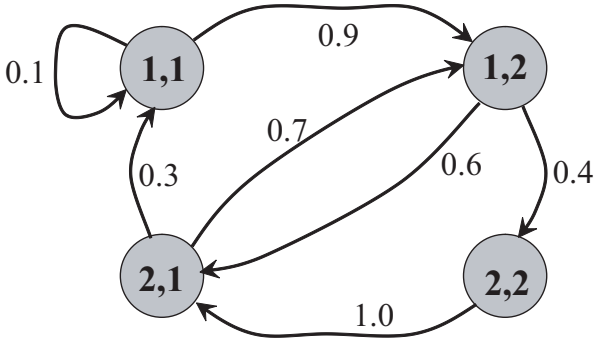

Figura 3.2: Grafo associado ao processo de Markov de ordem 1 definido em $\mathcal{X}^2 = \{(1,1),(1,2),(2,1),(2,2)\}$ equivalente ao processo de ordem 2 de alfabeto $\mathcal{X} = \{1,2\}$ com a matriz de transição do Exemplo 3.6.

A possibilidade de descrever uma fonte markoviana invariante no tempo de qualquer ordem, como um processo de primeira ordem sobre um alfabeto extendido permite focar a atenção sobre os processos de primeira ordem.

3.3.3 Distribuição dos Estados e Distribuição Estacionária

Como se viu na sub-secção anterior, um processo (ou fonte) de Markov de primeira ordem, definido num alfabeto (conjunto de estados) de dimensão N, $\mathcal{X} = \{1,...,N\}$, fica completamente definido pela matriz de transição $\mathbf{P}$ e pela distribuição de probabilidades inicial $\mathbf{p}(1)$.

A partir da distribuição inicial, e da matriz de transição, pode obter-se a distribuição relativa a qualquer instante. Para o instante 2, é imediato concluir que

$$p_i(2) = p(X_2 = i) = \sum_{j=1}^{N} p(X_1 = j)\ P(X_2 = i | X_1 = j) = \sum_{j=1}^{N} p_j(1)\ P_{j,i},$$

que tem o seguinte significado intuitivamente óbvio: a probabilidade de se encontrar a cadeia de Markov no estado i, no instante 2, é igual soma das probabilidades de todos os possíveis estados anteriores, $p(X_1 = j)$, multiplicadas pelas respectivas probabilidades de transitarem desses estados para o estado i, ou seja $P(X_2 = i|X_1 = j) = P_{j,i}$. Agrupando todas as probabilidades $p_i(2)$ no vector $\mathbf{p}(2)$, pode escrever-se

$$\mathbf{p}(2) = \begin{bmatrix} p_1(2) \\ p_2(2) \\ \vdots \\ p_N(2) \end{bmatrix} = \begin{bmatrix} \sum_{j=1}^{N} p(X_1 = j)\ P(X_2 = 1|X_1 = j) \\ \sum_{j=1}^{N} p(X_1 = j)\ P(X_2 = 2|X_1 = j) \\ \vdots \\ \sum_{j=1}^{N} p(X_1 = j)\ P(X_2 = N|X_1 = j) \end{bmatrix} = \begin{bmatrix} \sum_{j=1}^{N} p_j(1)\ P_{j,1} \\ \sum_{j=1}^{N} p_j(1)\ P_{j,2} \\ \vdots \sum_{j=1}^{N} p_j(1)\ P_{j,N} \end{bmatrix} = \mathbf{P}^T \mathbf{p}(1),$$

ou seja, a distribuição no instante 2 obtém-se multiplicando o vector da distribuição no instante 1 pela transposta da matriz de transição. Naturalmente, este facto pode generalizar-se para qualquer par de instantes consecutivos,

$$\mathbf{p}(t+1) = \mathbf{P}^T \mathbf{p}(t). \tag{3.11}$$

Aplicando esta igualdade recursivamente, obtém-se

$$\begin{aligned} \mathbf{p}(t+1) &= \mathbf{P}^T\, \mathbf{p}(t) \\ &= \mathbf{P}^T\, \mathbf{P}^T\, \mathbf{p}(t-1) \\ &= \underbrace{\mathbf{P}^T\, \mathbf{P}^T \cdots \mathbf{P}^T}_{t \text{ vezes}}\, \mathbf{p}(1) = (\mathbf{P}^T)^t\, \mathbf{p}(1) = (\mathbf{P}^t)^T\, \mathbf{p}(1). \end{aligned} \tag{3.12}$$

Quando o processo de Markov apresenta uma distribuição de estados que é invariante sob a acção da matriz $\mathbf{P}^T$, isto é, quando,

$$\mathbf{P}^T\, \mathbf{p}(t) = \mathbf{p}(t), \tag{3.13}$$

diz-se que o processo está em estado estacionário, e designa-se esta distribuição como estacionária. Denota-se esta distribuição por $\mathbf{p}(\infty)$, para salientar que é a distribuição que se mantém indefinidamente. Por inspecção de (3.13), verifica-se que $\mathbf{p}(\infty)$ é o vector próprio de $\mathbf{P}^T$ associado ao valor próprio 1 e cuja soma dos elementos é igual a 1 (para se tratar de uma distribuição de probabilidades válida).

Exemplo 3.7 *Considere-se o processo com dois estados $\mathcal{X} = \{1, 2\}$, com matriz de transição*

$$\mathbf{P} = \begin{bmatrix} 1-\alpha & \alpha \\ \beta & 1-\beta \end{bmatrix}.$$

A matriz $\mathbf{P}^T$ possui valores próprios 1 e $(1-\alpha-\beta)$. O vector próprio associado ao valor próprio unitário (normalizado para que a soma dos elementos seja 1) é

$$\mathbf{p}(\infty) = \begin{bmatrix} \dfrac{\beta}{\beta+\alpha} \\ \dfrac{\alpha}{\beta+\alpha} \end{bmatrix}$$

Uma cadeia de Markov diz-se *irredutível* se for possível transitar, num intervalo de tempo (discreto) finito, de qualquer estado, para qualquer estado. Formalmente, uma cadeia de Markov diz-se irredutível se, para qualquer par de estados $i \neq j \in \{1, ..., N\}$, existe um inteiro t, finito, tal que

$$\left(\mathbf{P}^t\right)_{i,j} > 0.$$

Note-se que a t-ésima potência da matriz de transição é a matriz de transição a t passos:

$$\left(\mathbf{P}^t\right)_{i,j} = P(X_t = j | X_1 = i).$$

Pode demonstrar-se[1] que, se um processo de Markov for irredutível, a matriz de transição correspondente possui um só valor próprio igual a 1 e todos os outros são menores que 1 em módulo. Assim, a distribuição estacionária $\mathbf{p}(\infty)$ é única. Mais, independentemente da distribuição inicial, a distribuição $\mathbf{p}(t)$ converge para a distribuição estacionária.

Finalmente, note-se que um processo de Markov invariante no tempo só é um processo estacionário se possuir distribuição estacionária única e se a distribuição inicial for igual è distribuição estacionária.

3.4 Taxas de Entropia

Os conceitos de taxa de entropia (como se verá, existem dois) generalizam o conceito de entropia de variáveis aleatórias para os processos estocásticos. Para um processo estocástico $\mathbf{X} = \{X_1, X_2, ..., X_i, ...\}$, a taxa de entropia define-se como

$$H(\mathbf{X}) = \lim_{t\to\infty} \frac{1}{t}\, H(X_1, X_2, ..., X_t), \tag{3.14}$$

quando o limite existe.

É importante notar que esta definição contém a entropia de uma fonte sem memória como caso particular. Para uma fonte sem memória, as variáveis $X_1, X_2, ..., X_t$ são todas independentes e identicamente distribuídas; assim, $H(X_1, X_2, ..., X_t) = H(X_1) + H(X_2) + \cdots + H(X_t)$. Designando por X uma variável aleatória com a mesma distribuição que $X_1, X_2, ..., X_t$, tem-se $H(X_1) = H(X_2) = ... = H(X_t) = H(X)$; logo,

$$\begin{aligned} H(\mathbf{X}) &= \lim_{t\to\infty} \frac{1}{t}\, H(X_1, X_2, ..., X_t) \\ &= \lim_{t\to\infty} \frac{1}{t}\, (H(X_1) + H(X_2) + \cdots + H(X_t)) \\ &= \lim_{t\to\infty} \frac{1}{t}\, t\, H(X) \\ &= H(X). \end{aligned}$$

Em conclusão, no caso de uma fonte sem memória, a taxa de entropia coincide com a entropia da variável aleatória que define a fonte.

Um outro conceito de entropia para processos estocásticos é a taxa de entropia condicional, designada $H'(\mathbf{X})$ e definida como

$$H'(\mathbf{X}) = \lim_{t\to\infty} H(X_t | X_{t-1}, ..., X_1), \tag{3.15}$$

quando o limite existe. Também este conceito de entropia coincide com a definição de entropia de uma variável aleatória, no caso de uma fonte sem memória. Numa fonte sem memória, pela propriedade de independência, tem-se $H(X_t|X_{t-1}, ..., X_1) = H(X_t)$; como todas as variáveis

[1] A demonstração deste resultado, baseado no famoso teorema de Perron-Frobenius, está para lá do âmbito deste texto; o leitor interessado pode encontrar mais detalhes em [3], ou em qualquer bom livro sobre processos de Markov.

aleatórias X_t são igualmente distribuídas, vem $H(X_t|X_{t-1},...,X_1) = H(X_t) = H(X)$, pelo que $H'(\mathbf{X}) = H(X)$.

Uma propriedade fundamental destes dois conceitos de entropia é a sua existência e igualdade, no caso dos processos estacionários:

$$\mathbf{X} \text{ é estacionário} \Rightarrow H'(\mathbf{X}) \text{ existe e } H'(\mathbf{X}) = H(\mathbf{X}).$$

A demonstração desta propriedade divide-se em dois passos:

Existência de $H'(\mathbf{X})$: Para demonstrar este facto (para $\mathbf{X}$ estacionário), começa por verificar-se que, pelo facto de o condicionamento reduzir a entropia,

$$H(X_t|X_{t-1},...,X_1) \leq H(X_t|X_{t-1},...,X_2); \tag{3.16}$$

invocando a estacionaridade do processo, tem-se

$$H(X_t|X_{t-1},...,X_2) = H(X_{t-1}|X_{t-2},...,X_1); \tag{3.17}$$

pelo que

$$H(X_t|X_{t-1},...,X_1) \leq H(X_{t-1}|X_{t-2},...,X_1). \tag{3.18}$$

Assim, a sequência $H(X_t|X_{t-1},...,X_1)$ é monotonicamente decrescente com t; como também se verifica que $H(X_t|X_{t-1},...,X_1) \geq 0$, tem-se uma sequência decrescente e limitada por baixo, logo convergente. Em conclusão, o limite que define $H'(\mathbf{X})$ existe.

Igualdade de $H(\mathbf{X})$ e $H'(\mathbf{X})$: A demonstração deste resultado suporta-se no teorema da média de Cesàro (demonstrado no Apêndice A), o qual afirma o seguinte: seja a_n uma sequência de números reais e b_n uma sequência definida a partir de a_n como

$$b_n = \frac{1}{n}\sum_{i=1}^{n} a_i,$$

isto é, a média dos n primeiros termos da sequência a_n; então,

$$\lim_{n\to\infty} a_n = a \quad \Rightarrow \quad \lim_{n\to\infty} b_n = a.$$

Invocando a regra de cadeia (1.20), escreve-se

$$\frac{1}{t}\, H(X_t,...,X_1) = \frac{1}{t}\sum_{s=1}^{t} H(X_s|X_{s-1},...,X_1)$$

(onde $H(X_1|X_0)$ significa simplesmente $H(X_1)$). Usando esta decomposição na definição de $H(\mathbf{X})$,

$$\begin{aligned} H(\mathbf{X}) &= \lim_{t\to\infty} \frac{1}{t}\, H(X_t,...,X_1) \\ &= \lim_{t\to\infty} \frac{1}{t}\sum_{s=1}^{t} H(X_s|X_{s-1},...,X_1) \\ &= \lim_{s\to\infty} H(X_s|X_{s-1},...,X_1) \\ &= H'(\mathbf{X}), \end{aligned} \tag{3.19}$$

em que a terceira igualdade resulta directamente to teorema da média de Cesàro.

A taxa de entropia para um processo de Markov de primeira ordem e estacionário (isto é, para o qual a distribuição inicial é igual à distribuição estacionária, $\mathbf{p}(1) = \mathbf{p}(\infty)$) tem uma forma particularmente simples. Da estacionaridade, decorre que $H(\mathbf{X}) = H'(\mathbf{X})$, sendo fácil obter $H'(\mathbf{X})$,

$$\begin{aligned} H'(\mathbf{X}) &= \lim_{t\to\infty} H(X_t|X_{t-1},...,X_1) \\ &= \lim_{t\to\infty} H(X_t|X_{t-1}) \qquad (3.20) \\ &= \lim_{t\to\infty} H(X_2|X_1) \qquad (3.21) \\ &= H(X_2|X_1), \end{aligned}$$

onde a igualdade (3.20) resulta da propriedade de Markov de primeira ordem e a igualdade (3.21) resulta do facto do processo ser estacionário. A taxa de entropia condicional de um processo de Markov de primeira ordem estacionário pode então escrever-se em termos da matriz de transição e da distribuição estacionária. Usando a definição de entropia condicional (ver (1.10)),

$$\begin{aligned} H'(\mathbf{X}) \;=\; H(X_2|X_1) &= \sum_{i\in\mathcal{X}} H(X_2|X_1=i)\; P(X_1=i) \\ &= \sum_{i=1}^{N} H(X_2|X_1=i)\; p_i(1) \\ &= -\sum_{i=1}^{N} p_i(1) \sum_{j=1}^{N} P(X_2=j|X_1=i) \log P(X_2=j|X_1=i) \\ &= -\sum_{i=1}^{N} p_i(\infty) \sum_{j=1}^{N} P_{i,j} \log P_{i,j} \qquad (3.22) \end{aligned}$$

onde se usou $\mathbf{p}(1) = \mathbf{p}(\infty)$, pois o processo é estacionário. Note-se ainda que, como $\mathbf{p}(\infty)$ depende exclusivamente da matriz de transição $\mathbf{P}$, também a taxa de entropia condicional $H'(\mathbf{X})$ de um processo de Markov estacionário depende exclusivamente de $\mathbf{P}$.

Exemplo 3.8 *Retomando o exemplo 3.7, pode escrever-se a taxa de entropia condicional usando a expressão (3.22) com a distribuição estacionária obtida nesse exemplo. Assim,*

$$H'(\mathbf{X}) = \frac{\beta}{\alpha+\beta}\, H(\alpha, 1-\alpha) + \frac{\alpha}{\alpha+\beta}\, H(\beta, 1-\beta),$$

em que $H(p, 1-p)$ denota a entropia de uma variável binária de probabilidades p e $1-p$.

Exemplo 3.9 *Neste exemplo estuda-se uma fonte de segunda ordem. Considere-se uma fonte, com alfabeto $\mathcal{X} = \{a, b, c\}$, que tem a seguinte característica: nunca emite 3 símbolos iguais seguidos; em cada instante, emite com equiprobabilidade os símbolos permitidos. É claro que, neste caso, o estado da cadeia de Markov é constituído pelos dois últimos símbolos; assim, como*

o conjunto de estados possíveis é $\mathcal{X}^2 = \{(a,a),(a,b),(a,c),(b,a),(b,b),(b,c),(c,a),(c,b),(c,c)\}$*, as probabilidades condicionais que caracterizam esta fonte são*

$$\mathbf{P} = \begin{bmatrix} 0 & 1/2 & 1/2 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/2 & 0 & 1/2 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/2 & 1/2 & 0 \end{bmatrix}$$

Note-se que os zeros que surgem na matriz indicam que, após a sequência "a,a*" não pode ser emitido um novo* a*, após a sequência "*b,b*" não pode ser emitido um novo* b*, e após a sequência "*c,c*" não pode ser emitido um novo* c*. Como visto acima (Exemplo 3.6), este processo pode ser escrito como um processo de primeira ordem, definido no novo conjunto de estados* $\mathcal{X}^2 = \{(a,a),(a,b),(a,c),(b,a),(b,b),(b,c),(c,a),(c,b),(c,c)\}$*, com matriz de transição*

$$\mathbf{P}' = \begin{bmatrix} 0 & 1/2 & 1/2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/3 & 1/3 & 1/3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/2 & 0 & 1/2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/3 & 1/3 & 1/3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1/2 & 1/2 & 0 \end{bmatrix}$$

A matriz $\mathbf{P}'^T$ *possui, como vector próprio associado ao seu valor próprio 1, o vector* $\mathbf{p}(\infty) = (1/9)[1,1,1,1,1,1,1,1,1]^T$*, o que significa que, em estado estacionário, os nove estados possíveis (elementos de* $\mathcal{X}^2$*) são equiprováveis. Quanto à taxa de entropia condicional,*

$$H'(\mathbf{X}) = \frac{1}{9}\left(3\,H(1/2,1/2) + 6H(1/3,1/3,1/3)\right) = \frac{3+6\log_2 3}{9} = 1.39 \text{ } \textit{bits/símbolo.}$$

3.5 Codificação de Fontes com Memória

O desenho de códigos óptimos para fontes com memória resume-se ao desenho de códigos óptimos para a distribuição de símbolos associada a cada estado da cadeia de Markov que modela a fonte. É óbvio que o limite inferior para o comprimento médio do código assim obtido é a taxa de entropia condicional, como ilustrado no exemplo seguinte.

Exemplo 3.10 *Retomando o exemplo 3.9, considere-se o problema de desenhar um esquema de codificação óptimo para a fonte descrita. Por observação da matriz* $\mathbf{P}$*, constata-se que, em*

6 dos 9 estados, a fonte se comporta como uma fonte ternária na qual os três símbolos $\{a, b, c\}$ são equiprováveis. Nos restantes 3 estados, a fonte comporta-se como uma fonte binária, na qual os dois símbolos possíveis são equiprováveis. Assim, a codificação óptima consiste em usar o seguinte conjunto de códigos.

Códigos óptimos	*símbolo*		
símbolos anteriores	*a*	*b*	*c*
aa	·	*0*	*1*
ab	*0*	*10*	*11*
ac	*0*	*10*	*11*
ba	*0*	*10*	*11*
bb	*0*	·	*1*
bc	*0*	*10*	*11*
ca	*0*	*10*	*11*
cb	*0*	*10*	*11*
cc	*0*	*1*	·

O símbolo "·" significa que não é necessário ter uma palavra de código para o símbolo respectivo, pois este tem probabilidade zero de ser emitido. O comprimento médio dos códigos com três palavras (condicionalmente equiprováveis) é

$$\frac{1+2+2}{3} = \frac{5}{3} \simeq 1.6667 \text{ } \textit{bits/símbolo},$$

enquanto que o dos códigos com apenas duas palavras é 1 bit/símbolo. Dado que todos os 9 estados são equiprováveis, o comprimento médio global é

$$\frac{1}{9}\left(6\,\frac{5}{3}+3\right) \simeq 1.4444 \text{ } \textit{bits/símbolo},$$

ligeiramente acima da taxa de entropia condicional $H'(\mathbf{X}) = 1.39$ bits/símbolo. Se se ignorasse a memória da fonte e se desenhasse em código ajustado para a distribuição não condicional dos símbolos (sob a qual os 3 símbolos são obviamente equiprováveis), obtinha um comprimento médio de $\frac{5}{3} \simeq 1.6667$ bits/símbolo, pior do que o que se obtém com o código desenhado para as probabilidades condicionais.

Apêndice A

Demonstração do Teorema da Média de Cesàro

Seja a_n uma sequência de números reais e b_n uma sequência definida a partir de a_n como

$$b_n = \frac{1}{n}\sum_{i=1}^{n} a_i,$$

isto é, a média dos n primeiros termos da sequência a_n. O teorema da média de Cesàro afirma que

$$\lim_{n\to\infty} a_n = a \;\Rightarrow\; \lim_{n\to\infty} b_n = a.$$

Demonstração: O facto de a_n convergir para a é equivalente a

$$\forall_{\varepsilon>0}\ \exists_{N(\varepsilon)}\ :\ n \geq N(\varepsilon) \Rightarrow |a_n - a| < \varepsilon,$$

isto é, para qualquer vizinhança de a, arbitrariamente pequena, existe um ponto da sequência a_n a partir do qual todos os termos pertencem a esta vizinhança.

Para demonstrar o teorema da média de Cesàro é necessário demonstrar uma implicação semelhante para a sequência b_n. Para tal, toma-se um valor arbitrariamente pequeno $\varepsilon > 0$ e o correspondente $N(\varepsilon)$ e escreve-se, para $n > N(\varepsilon)$,

$$\begin{aligned} |b_n - a| &= \frac{1}{n}\left|\sum_{i=1}^{n}(a_i - a)\right| \\ &\leq \frac{1}{n}\sum_{i=1}^{n}|a_i - a| \\ &= \frac{1}{n}\sum_{i=1}^{N(\varepsilon)}|a_i - a| + \frac{1}{n}\sum_{N(\varepsilon)+1}^{n}|a_i - a|. \end{aligned}$$

Mas, $n \geq N(\varepsilon) \Rightarrow |a_n - a| < \varepsilon$, pelo que

$$\begin{aligned} |b_n - a| &< \frac{1}{n}\sum_{i=1}^{N(\varepsilon)} |a_i - a| + \frac{1}{n}\sum_{N(\varepsilon)+1}^{n} \varepsilon. \\ &= \frac{1}{n}\sum_{i=1}^{N(\varepsilon)} |a_i - a| + \frac{n - N(\varepsilon)}{n}\,\varepsilon. \\ &\leq \frac{1}{n}\sum_{i=1}^{N(\varepsilon)} |a_i - a| + \varepsilon. \end{aligned} \tag{A.1}$$

Note-se que, para um dado $N(\varepsilon)$, a quantidade $A(\varepsilon) = \sum_{i=1}^{N(\varepsilon)} |a_i - a|$ é uma constante independente de n. Assim, dado um valor $\delta > 0$ arbitrariamente pequeno, tome-se $\varepsilon = \delta/2$; daqui resulta

$$|b_n - a| < \frac{1}{n}A(\varepsilon) + \delta/2. \tag{A.2}$$

Finalmente,

$$n \geq \frac{2}{\delta}A(\varepsilon) \;\Rightarrow\; |b_n - a| < \delta.$$

Isto é, definindo $M(\delta) = 2\ A(\delta/2)\ /\delta$,

$$\forall_{\delta>0},\ \ n \geq M(\delta) \;\Rightarrow\; |b_n - a| < \delta,$$

o que significa que $\lim_{n\to\infty} b_n = a$, como se queria demonstrar.

Bibliografia

[1] R. Ash, *Information Theory.* New York: Dover Publications, 1965.

[2] R. Ayres, *Information, Entropy, and Progress: A New Evolutionary Paradigm.* New York: Springer Verlag, 1994.

[3] P. Brémaud, *Markov Chains.* Springer Verlag, 1999.

[4] T. Cover and J. Thomas, *Elements of Information Theory.* New York: John Wiley & Sons, 1991.

[5] K. Eckschlager, *Information Theory in Analytical Chemistry.* New York: Wiley, 1994.

[6] D. Huffman, "A method for the construction of minimum redundancy codes,"Proceedings of the IRE, vol. 40, pp. 1098–1101, 1952.

[7] S. Kullback, *Information Theory and Statistics.* New York: John Wiley & Sons, 1959.

[8] M. Li and P. Vitanyi, *An Introduction to Kolmogorov Complexity and its Applications.* New York: Springer Verlag, 1997.

[9] F. Rieke, *Exploring the Neural Code.* Cambridge, M.A.: MIT Press, 1997.

[10] C. E. Shannon, "A mathematical theory of communication," *Bell Systems Technical Journal*, vol. 27, pp. 379–423, 623–656, 1948.

[11] T. Stonier, *Information and the Internal Structure of the Universe: an Exploration into Information Physics.* New York: Springer Verlag, 1990.

[12] H. Yockey, *Information Theory and Molecular Biology.* Cambridge: Cambridge University Press, 1992.