Códigos compactos

- Para comparar diferentes códigos, precisamos de saber com que fonte eles vão ser usados, ou seja precisamos de uma distribuição de probabilidades para os símbolos a serem codificados.
- Sejam $\{P_i : i = 1, \ldots, q\}$ as probabilidades dos símbolos fonte e sejam $\ell_1, \ldots, \ell_q$ os comprimentos das palavras-código correspondentes no código C. Definimos o *comprimento médio* do código C como sendo $L = \sum_{i=1}^{q} P_i \, \ell_i$.
- Fixando a fonte, L pode ser considerada função (linear) do q-uplo $(\ell_1, \ldots, \ell_q) \in \mathbb{Z}_+^q$. Seja r o cardinal do alfabeto do código. Se $r \geqslant q$, então é claro que o mínimo de L é 1. Por isso, passamos também a supor que r é fixado.

- Um código diz-se um *código compacto* se minimizar o valor de L para os códigos sobre um alfabeto com r símbolos.
- Por exemplo, consideremos os dois códigos e a fonte dados pela seguinte tabela:

Fonte	P_i	**A**	**B**
s_1	0.5	00	1
s_2	0.1	01	000
s_3	0.2	10	001
s_4	0.2	11	01

Calculando L para cada um destes códigos, obtém-se $L_{\mathbf{A}} = 2$ e $L_{\mathbf{B}} = (0.5)1 + (0.1)3 + (0.2)3 + (0.2)2 = 1.8$, pelo que **B** tem um comprimento médio inferior ao de **A**.

- Mais geralmente, graças aos Teoremas de Kraft e McMillan, o problema da determinação de códigos compactos com q elementos sobre um alfabeto com r elementos fica reduzido a calcular o mínimo da função

$$L = \sum_{i=1}^{q} P_i \, \ell_i$$

das variáveis inteiras positivas $\ell_1, \ldots, \ell_q$ sujeitas à desigualdade de Kraft

$$\sum_{i=1}^{q} r^{-\ell_i} \leqslant 1.$$

Trata-se de um problema de *programação inteira* (*não-linear*, pois a desigualdade não é linear nos ℓ_i).

Um minorante para o comprimento médio de um código

- Recorde-se que a entropia $H(S)$ (na base 2) da fonte mede o número de bits por símbolo da fonte. Codificando a fonte S em código binário, obtemos um número médio de bits por símbolo. A própria terminologia sugere que o segundo número deve ser maior ou igual ao primeiro:

Teorema 3.5 *Seja $S = \{s_1, \ldots, s_q\}$ uma fonte com entropia na base r dada por $H_r(S)$ e seja L o comprimento médio das palavras-código numa codificação de S por um código num alfabeto com r letras. Então tem-se*

$$L \geqslant H_r(S)$$

e a igualdade verifica-se se e só se $P_i = r^{-\ell_i}$ para $i = 1, \ldots, q$.

Prova. Pelo Teorema de McMillan, tem-se $\sum_{i=1}^{q} r^{-\ell_i} \leqslant 1$. Sem alterar os valores de L e $H_r(S)$, podemos completar a lista $(\ell_1, \ldots, \ell_q)$ acrescentando inteiros positivos de forma que $\sum_{i=1}^{q} r^{-\ell_i} = 1$ e estender a distribuição de probabilidades $(P_1, \ldots, P_q)$ acrescentando zeros. Ficamos então com duas distribuições de probabilidades e o resultado segue da forma habitual por aplicação do Lema 1.1. □

Eficiência de um código

- Note-se que, pelo Teorema 3.5, o minorante $H_r(S)$ só é atingido no caso da distribuição de probabilidade da fonte ser da forma $P_i = r^{-\ell_i}$, sendo em geral $L > H_r(S)$.
- A *eficiência* do código $C : S \to A^*$ para uma fonte S, com $|A| = r$ e comprimento médio L, é dada por

$$\eta = \frac{H_r(S)}{L} \times 100\%.$$

Exemplo

Consideremos o seguinte exemplo de novo:

Fonte	P_i	**A**	**B**
s_1	0.5	00	1
s_2	0.1	01	000
s_3	0.2	10	001
s_4	0.2	11	01

Aqui, temos $r = 2$, e $H(S) = (0.5)\log\frac{1}{0.5} + (0.1)\log\frac{1}{0.1} + 2(0.2)\log\frac{1}{0.2} \simeq 1.761$ bits por símbolo, pelo que o comprimento médio de um código binário para esta fonte tem de ser pelo menos 1.761.

A eficiência dos códigos **A** e **B** é, respetivamente, $\eta_{\mathbf{A}} \simeq \frac{1.761}{2} \simeq 88\%$ e $\eta_{\mathbf{B}} \simeq \frac{1.761}{1.8} \simeq 98\%$.

Fontes especiais

- Recorde-se que a condição para que a igualdade $L = H_r(S)$ seja atingida para algum código é que $P_i = r^{-\ell_i}$ para todo o i, ou seja $\ell_i = \log_r \frac{1}{P_i}$ $(i = 1, \ldots, q)$.
- Uma fonte com distribuição de probabilidades $\{P_i : i = 1, \ldots, q\}$ diz-se uma *fonte especial para os códigos em r letras* se os números $\log_r \frac{1}{P_i}$ $(i = 1, \ldots, q)$ forem todos inteiros.
- Assim, para as fontes especiais S para códigos em r letras, existem códigos compactos 100% eficientes, i.e., com comprimento médio $L = H_r(S)$.

Exemplos

- Considere-se a fonte dada pela tabela

Fonte	s_1	s_2	s_3	s_4
P_i	0.125	0.25	0.5	0.125

Note-se que $P_i = 2^{-\ell_i}$ com $\ell_i = 3, 2, 1, 3$, respetivamente. Logo existe um código compacto binário 100% eficiente para esta fonte, nomeadamente o código dado pela tabela

Fonte	s_1	s_2	s_3	s_4
Código	110	10	0	111

- Considere-se em seguida a fonte dada pela tabela

Fonte	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9
P_i	$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{3}$	$\frac{1}{27}$	$\frac{1}{27}$	$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{27}$	$\frac{1}{9}$

onde as probabilidades são da forma $P_i = 3^{-\ell_i}$, respetivamente para $\ell_i = 2, 2, 1, 3, 3, 2, 2, 3, 2$.

Um código ternário compacto 100% eficiente para esta fonte é dado pela tabela

Fonte	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9
Código	10	11	0	220	221	12	20	222	21

Majorante para o comprimento médio de códigos compactos

- Recorde-se que o comprimento médio L de um código r-ário para uma fonte S satisfaz a desigualdade $L \geqslant H_r(S)$ e que a igualdade só pode ser atingida para fontes especiais.

Teorema 3.6 *Seja L o comprimento médio de um código compacto r-ário para a fonte S. Então verificam-se as desigualdades*

$$H_r(S) \leqslant L < H_r(S) + 1.$$

Prova. Seja q o número de símbolos da fonte S e seja ℓ_i o único inteiro que satisfaz as desigualdades

$$\log_r \frac{1}{P_i} \leqslant \ell_i < \log_r \frac{1}{P_i} + 1.$$

Note-se que os ℓ_i satisfazem a desigualdade de Kraft, pelo que existe um código (prefixo) C com $|C(s_i)| = \ell_i$:

$$\sum_{i=1}^{q} r^{-\ell_i} \leqslant \sum_{i=1}^{q} r^{\log_r P_i} = \sum_{i=1}^{q} P_i = 1.$$

Por outro lado, temos

$$\sum_{i=1}^{q} P_i \log_r \frac{1}{P_i} \leqslant \sum_{i=1}^{q} P_i \ell_i < \sum_{i=1}^{q} P_i \log_r \frac{1}{P_i} + \sum_{i=1}^{q} P_i$$

o que mostra que, para o código C, tem-se $H_r(S) \leqslant L_C < H_r(S) + 1$.
Finalmente, como, para um código compacto, se tem $H_r(S) \leqslant L \leqslant L_C < H_r(S) + 1$, o resultado segue. □

Melhoramento da eficiência por extensão

- Seja $S^n = \{\sigma_1^n, \sigma_2^n, \ldots, \sigma_{q^n}^n\}$ a n-ésima extensão da fonte $S = \{s_1, s_2, \ldots, s_q\}$ onde $\sigma_i^n = s_{i1}s_{i2}\cdots s_{in}$.
- Consideremos para S^n um código compacto r-ário e seja L_n o seu comprimento médio. Pelo Teorema 3.6, tem-se

$$H_r(S^n) \leqslant L_n < H_r(S^n) + 1.$$

- Supondo que S não tem memória, do Teorema 1.14 segue que $H_r(S^n) = nH_r(S)$, pelo que

$$H_r(S) \leqslant \frac{L_n}{n} < H_r(S) + \frac{1}{n}.$$

- Note-se que $\frac{L_n}{n}$ é o *comprimento médio das palavras-código por símbolo* s_i quando as mensagens da fonte S são codificadas não símbolo a símbolo mas por blocos de comprimento n.

O teorema de Shannon para fontes sem memória

Teorema 3.7 *Seja L_n o comprimento médio de um código compacto r-ário para a n-ésima extensão de uma fonte S sem memória. Então verificam-se as desigualdades*

$$H_r(S) \leqslant \frac{L_n}{n} < H_r(S) + \frac{1}{n},$$

pelo que se tem $\lim_{n\to\infty} \frac{L_n}{n} = H_r(S)$.□

- Em contrapartida ao aumento de eficiência do código, em vez de trabalharmos com um código com q palavras, passamos a trabalhar com um código com q^n palavras.

Exemplo

Considere-se a fonte sem memória S com dois símbolos s_1, s_2 e respetivas probabilidades $0.8, 0.2$. Como só temos dois símbolos, temos o seguinte código binário compacto:

Fonte	P_i	Código compacto
s_1	0.8	0
s_2	0.2	1

A eficiência deste código é $\eta = \frac{H(S)}{L}$, onde $L = 1$ é o comprimento médio e $H(S) \simeq 0.722$, ambos em bits por símbolo, ou seja eficiência aproximada de $\eta \simeq 72\%$.

Para a segunda extensão da fonte, temos o seguinte código binário compacto:

Fonte	P_i	Código compacto
$s_1 s_1$	0.64	0
$s_1 s_2$	0.16	10
$s_2 s_1$	0.16	110
$s_2 s_2$	0.04	111

cujo comprimento médio é $L_2 = (0.64)1 + (0.16)2 + (0.16)3 + (0.04)3 = 1.56$ bits por (par de) símbolos e $\frac{L_2}{2}$ bits por símbolo (de S).

A eficiência melhorou para $\eta = \frac{H(S)}{L_2/2} \simeq 92.6\%$.

O teorema de Shannon para fontes de Markov

- Enquanto o Teorema 3.6 foi estabelecido para fontes sem memória, esta hipótese só interveio para mostrar que $H_r(S^n) = nH_r(S)$.
- Ora esta igualdade também vale para extensões de fontes de Markov de ordem m.

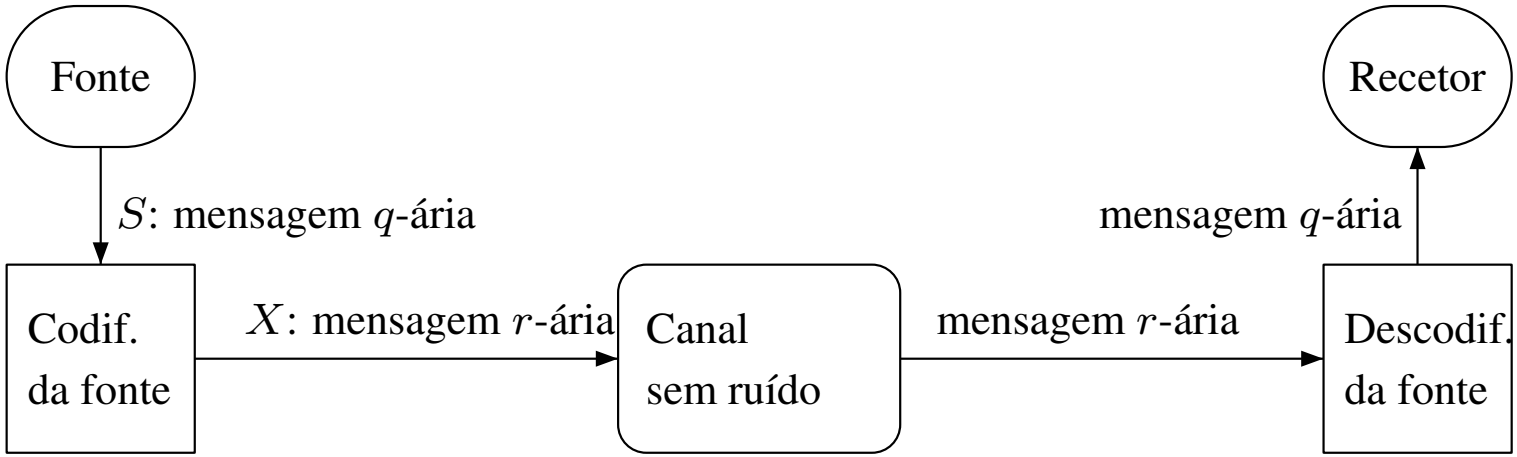
Teorema 3.8 *Seja L_n o comprimento médio de um código compacto r-ário para a n-ésima extensão de uma fonte de Markov M de ordem m. Então verificam-se as desigualdades*

$$H_r(M) \leqslant \frac{L_n}{n} < H_r(M) + \frac{1}{n},$$

pelo que se tem $\lim_{n\to\infty} \frac{L_n}{n} = H_r(M)$.□

Eficiência do código vs capacidade do canal

No caso de um canal sem ruído, recorde-se que temos o seguinte esquema de comunicação:



- Para a fonte S, a entropia $H(S)$ é o *número médio de bits transmitidos por símbolo da fonte*.
- Cada símbolo da fonte é codificado por uma palavra-código cujo comprimento médio é L.
- O quociente $H(X) = \frac{H(S)}{L}$ é o *número médio de bits transmitidos por símbolo do código*, pelo que diz a *entropia do código*.
- Sendo $H_r(S) = \frac{H(S)}{\log_2 r}$ e $\eta = \frac{H_r(S)}{L}$, temos $H(X) = \eta \log_2 r$. Assim, o código será 100% eficiente se e só se $H(X) = \log_2 r$, o que corresponde precisamente ao caso de X ter entropia máxima para uma fonte com r símbolos.
- Logo, num código 100% eficiente todos os símbolos do código ocorrem com a mesma probabilidade.

Exemplo

Considere-se a seguinte sequência binária, que se assume ser produzida por uma fonte sem memória: $0\,0\,0\,1\,1\,0\,0\,1\,0\,0$.

Temos então as seguintes probabilidades observadas: $P(0) = 0.7$, $P(1) = 0.3$, do que resulta $H(S) \simeq 0.88$ (bits por símbolo).

Para a segunda extensão da fonte temos o seguinte código compacto (para a distribuição de probabilidades resultante das observadas para os símbolos individuais, na hipótese de não haver memória):

$s_i s_j$	$P(s_i, s_j)$	Código
00	0.49	0
01	0.21	10
10	0.21	110
11	0.09	111

O comprimento médio do código por símbolo da fonte é $\frac{L_2}{2} = 0.905$ e a eficiência (por símbolo da fonte) é $\eta = \frac{H(S)}{L_2/2} \simeq 0.972$. A fonte obtida por codificação da sequência inicial é $0\,10\,110\,10\,0$, onde os símbolos 0 e 1 têm frequências respetivas $P(0) = \frac{5}{9}$ e $P(1) = \frac{4}{9}$, pelo que a entropia correspondente é $H \simeq 0.99$, obtendo assim uma entropia superior à entropia da fonte inicial.

Esquema de codificação de (Shannon-)Fano

- Seja S uma fonte com símbolos $s_1, \ldots, s_q$ e consideremos as respetivas probabilidades que supomos satisfazerem $P(s_1) \geqslant P(s_2) \geqslant \cdots \geqslant P(s_q)$.
  Pretendemos construir um sistema de codificação em r símbolos.
- Atribuímos inicialmente a cada símbolo a palavra-código vazia.
- Dividimos os símbolos s_i em $(\leqslant)\, r$ grupos, respeitando a ordem das probabilidades, agrupando-os de forma que as probabilidades conjuntas fiquem "próximas".
- Acrescentamos à palavra-código parcial já construída para cada um dos símbolos do i-ésimo grupo o símbolo-código x_i.
- Procedemos de forma idêntica com cada grupo de símbolos que contenha mais do que um símbolo.

Exemplo

Tomemos uma fonte com as seguintes probabilidades:

s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
$1/2$	$1/8$		$1/16$			$1/32$	

Procedendo sucessivamente conforme o esquema de Fano descrito acima, obtemos, de acordo com os agrupamentos indicados:

Prob.	Cód.	Prob.	Cód.	Prob.	Cód.	Prob.	Cód.	Prob.	Cód.
$1/2$	0	$1/2$	0	$1/2$	0	$1/2$	0	$1/2$	0
$1/8$	1	$1/8$	10	$1/8$	100	$1/8$	100	$1/8$	100
$1/8$	1	$1/8$	10	$1/8$	101	$1/8$	101	$1/8$	101
$1/16$	1	$1/16$	11	$1/16$	110	$1/16$	1100	$1/16$	1100
$1/16$	1	$1/16$	11	$1/16$	110	$1/16$	1101	$1/16$	1101
$1/16$	1	$1/16$	11	$1/16$	111	$1/16$	1110	$1/16$	1110
$1/32$	1	$1/32$	11	$1/32$	111	$1/32$	1111	$1/32$	11110
$1/32$	1	$1/32$	11	$1/32$	111	$1/32$	1111	$1/32$	11111

Trata-se de um código compacto, 100% eficiente ($L = H(S) = 2.3125$).

Exemplo

Tomemos uma fonte com as seguintes probabilidades:

s_1	s_2	s_3	s_4	s_5	s_6
$4/9$	$1/9$				

Aplicando o método de Fano, teremos no primeiro passo duas formas de proceder ao agrupamento dos símbolos: $\{s_1|s_2, s_3, s_4, s_5, s_6\}$, com probabilidades $\frac{4}{9}$, $\frac{5}{9}$, ou $\{s_1, s_2|s_3, s_4, s_5, s_6\}$, com probabilidades $\frac{5}{9}$, $\frac{4}{9}$.
Conforme a escolha inicial, obtemos os seguintes códigos:

	Prob.	X
1	$4/9$	0
4	$1/9$	1000
3	$1/9$	1001
2	$1/9$	101
3	$1/9$	110
	$1/9$	111

	Prob.	Y
2	$4/9$	00
1	$1/9$	01
3	$1/9$	100
2	$1/9$	101
3	$1/9$	110
	$1/9$	111

Aqui, obtemos $L_X = 21/9$ e $L_Y = 22/9$, e portanto Y não pode ser um código compacto.
Em particular, um código de Fano não é necessariamente compacto.

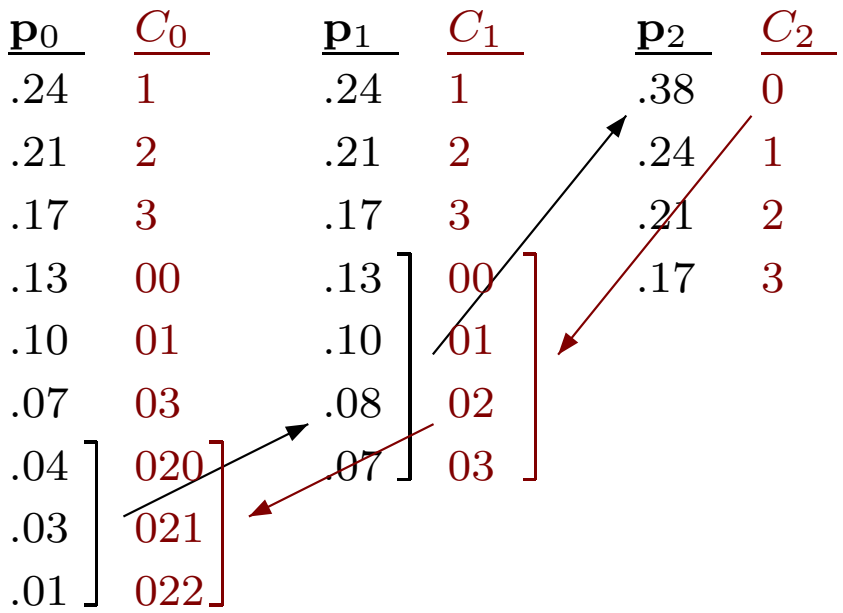
Códigos de Huffman

- O *algoritmo de Huffman*, introduzido em 1952 e durante muito tempo utilizado para comprimir informação, produz um código compacto r-ário para uma dada fonte q-ária.
- Antes de formalizar o algoritmo e provar que efetivamente produz um código compacto, vejamos como funciona num exemplo.

Exemplo

Seja $\mathbf{p}_0 = (.24, .21, .17, .13, .10, .07, .04, .03, .01)$ ($q = 9$) a distribuição de probabilidades de uma fonte que pretendemos codificar num código com símbolos $0, 1, 2, 3$ ($r = 4$).
Começamos por agrupar os últimos $r - 1 = 3$ símbolos, o que conduz a uma nova distribuição de probabilidades (por ordem decrescente)
$\mathbf{p}_1 = (.24, .21, .17, .13, .10, .08, .07)$.
Repetimos este processo até que o número de probabilidades seja reduzido a r, agrupando nas restantes etapas r elementos: $\mathbf{p}_2 = (.38, .24, .21, .17)$.
Para esta última distribuição de probabilidades, uma codificação com 4 símbolos é obtida simplesmente associando-lhes univocamente os símbolos $0, 1, 2, 3$.
A seguir, recuamos no processo de agrupamento das probabilidades, considerando o código para $\mathbf{p}_1$ dado por $1, 2, 3, 00, 01, 02, 03$ e, para $\mathbf{p}_0$ o código $1, 2, 3, 00, 01, 03, 020, 021, 022$.

O seguinte diagrama resume este procedimento:



Não é contudo claro neste exemplo exactamente quantas probabilidades devem ser agrupadas em cada etapa...