



Sessão nº6

Codificadores Estatísticos



Como reduzir a redundância?

- Para reduzir a redundância na representação dos símbolos da fonte deve-se estabelecer o código tal que **aos símbolos com maior probabilidade associa-se palavras de código com menor comprimento e vice-versa.**
- Qual a técnica que permite desenhar de forma sistemática um código que se aproxime do ideal, ou seja, que $N \approx H(X)$ para um dado valor de $H(X)$?



Codificação de repetições: *Run Length Encoding*



- ***Run Length Encoding*** (RLE) é uma técnica simples usada na compressão de repetições de símbolos num “bloco” de dados. **Tipicamente codifica uma repetição como um símbolo e o número de vezes que este se repete.**

– A **simplicidade** do RLE e a **facilidade de implementação** resulta na sua frequente utilização em compressores, apesar da sua **eficácia relativamente pobre**.



Código de Shannon-Fano (depois de 1948)



x_i	$p(x_i)$	$-\log_2 p(x_i)$	$C(x_i)$
a	1/2	1	0
b	1/4	2	10
c	1/8	3	110
d	1/8	3	111

Algoritmo: dividir o conjunto de probabilidades em subconjuntos de igual probabilidade, atribuindo a cada subconjunto os dígitos **0** e **1**; assim sucessivamente para cada subconjunto.

- Entropia da fonte

$$\mathbf{H(X)} = \sum_{x_i \in X} p(x_i) \times -\log_2 p(x_i) = 1,75 \text{ [bit/símbolo]}$$

- Comprimento médio do código (**prefixo** porque verifica $\sum_i 2^{-n(x_i)} \leq 1$)

$$\mathbf{N} = \sum_{x_i \in X} p(x_i) \times n(x_i) = 1,75 \text{ dígitos}$$

– note-se que $\mathbf{N} = \mathbf{H(X)} + \mathbf{0}$ sendo $\epsilon = \mathbf{0}$ logo **código ótimo!**

- Redundância na representação dos símbolos da fonte

$$\mathbf{R} = 1 - (1,75 / (1,75 \times \log_2 2)) = \mathbf{0}$$



Codificação de Huffman (1952) - algoritmo



1. Seja \mathbf{L} a lista de probabilidades de uma fonte de símbolos que se consideram estar associados às folhas de uma árvore binária.
 2. Considerem-se as duas menores probabilidades de \mathbf{L} e tomem-se como dois descendentes de um nó. Gere-se um nó intermédio para suportar esses descendentes e atribua-se a um dos ramos o dígito $\mathbf{0}$ e ao outro o dígito $\mathbf{1}$.
 3. Substitua-se as duas probabilidades e seus nós associados por um novo nó intermédio com a soma das duas probabilidades. Se a lista resultante ficar com um único nó terminar. Senão repetir a partir do passo 2.
- Para se obter o código associado a cada símbolo percorre-se a árvore desde a raiz até às folhas associadas aos respectivos símbolos, acumulando e considerando como dígitos de menor peso, os dígitos associados aos ramos "percorridos".



Código de Huffman



x_i	$p(x_i)$	$-\log_2 p(x_i)$	$C(x_i)$
a	1/2	1	0
b	1/4	2	10
c	1/8	3	110
d	1/8	3	111

- Entropia da fonte

$$\mathbf{H(X)} = \sum_{x_i \in X} p(x_i) \times -\log_2 p(x_i) = 1,75 \text{ [bit/símbolo]}$$

- Comprimento médio do código (**prefixo** porque verifica $\sum_i 2^{-n(x_i)} \leq 1$)

$$\mathbf{N} = \sum_{x_i \in X} p(x_i) \times n(x_i) = 1,75 \text{ dígitos}$$

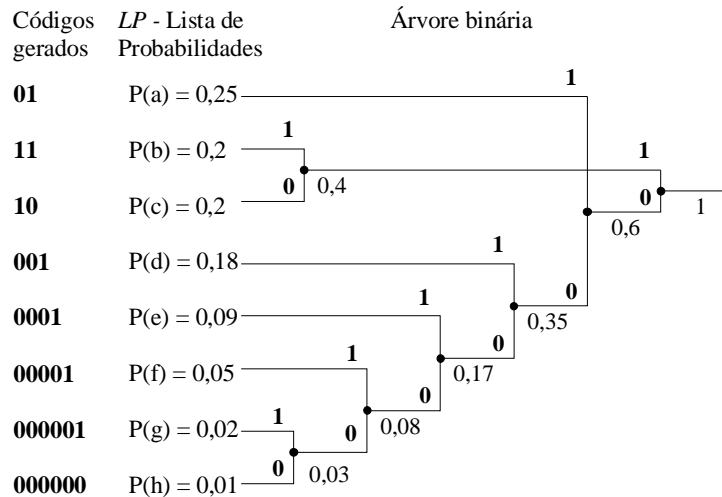
– note-se que $\mathbf{N} = \mathbf{H(X)} + \mathbf{0}$ sendo $\epsilon = \mathbf{0}$ logo **código ótimo!**

- Redundância na representação dos símbolos da fonte

$$\mathbf{R} = 1 - (1,75 / (1,75 \times \log_2 2)) = \mathbf{0}$$



Codificação de Huffman



Extensão de fonte discreta sem memória



- A **extensão da fonte** duma fonte discreta sem memória consiste em agrupar n símbolos da fonte para constituir “super-símbolos”, sendo agora esses os símbolos produzidos pela (“nova”) fonte estendida.
- Seja a fonte \mathbf{X} tal que $x_i \in \{x_1, x_2, x_3\}$ sendo $p(x_1)=1/4$, $p(x_2)=1/4$ e $p(x_3)=1/2$.
 - entropia da fonte \mathbf{X} $H(\mathbf{X}) = \sum_{x_i \in X} p(x_i) \times -\log_2 p(x_i) = 1,5$ [bit/símbolo]
- Seja a fonte $\mathbf{Y}=\mathbf{X}^2$ a extensão de 2ª ordem da fonte \mathbf{X} tal que $y_1 = x_1 x_1$, $y_2 = x_1 x_2$, $y_3 = x_1 x_3$, etc. Então $y_i \in \{y_1, y_2, y_3, y_4, y_5, y_6, y_7, y_8, y_9\}$ com $p(y_1)=1/16$, $p(y_2)=1/16$, $p(y_3)=1/8$, etc.
 - entropia da fonte \mathbf{X}^2 $H(\mathbf{X}^2) = \sum_{x_i \in X} p(y_i) \times -\log_2 p(y_i) = 3$ [bit/símbolo]
- Verifica-se que $H(\mathbf{X}^2) = 2 \times H(\mathbf{X})$
porque os símbolos são independentes; a fonte não tem memória.



Código prefixo estendido



- Seja N_n o comprimento médio das palavras do código prefixo (estendido) estabelecido para a fonte estendida X^n . Então porque o código é prefixo, deduz-se que

$$H(X^n) \leq N_n \leq H(X^n) + 1 \Leftrightarrow$$

$$n \times H(X) \leq N_n \leq (n \times H(X)) + 1 \Leftrightarrow$$

$$H(X) \leq N_n/n \leq H(X) + 1/n \Leftrightarrow$$

$$H(X) \leq N \leq H(X) + 1/n$$

- por isso, considerando blocos de símbolos suficientemente grandes e **codificando em conjunto**, o comprimento médio do código prefixo estendido pode-se tornar tão pequeno quanto a entropia da fonte.

