Teoria da Informação (#4)

Código de Shannon-Fano-Elias e codificação aritmética

Miguel Barão

ESCOLA DE CIÊNCIAS E TECNOLOGIA
DEPARTAMENTO DE INFORMÁTICA

Ideia Em vez de construir uma árvore para obter as palavras de código, estas são obtidas a partir da da expansão binária das probabilidades acumuladas.

Código Shannon-Fano-Elias

Ideia Em vez de construir uma árvore para obter as palavras de código, estas são obtidas a partir da da expansão binária das probabilidades acumuladas.

Algoritmo

1. Calculam-se os comprimentos das palavras de código
$$l(x) = \lceil -log_2 p(x) \rceil + 1.$$
(Note que tem mais um bit do que o código de Shannon!)
2. Dado um alfabeto ordenado $\mathcal{X}$ e probabilidades $p(x)$, constrói-se a função de probabilidade acumulada modificada
$$\bar{F}(x) = \sum_{x_i < x} p(x_i) + \frac{1}{2} p(x).$$
3. Calcula-se a expansão binária de $\bar{F}(x)$ para cada $x \in \mathcal{X}$.
4. As palavras de código são os primeiros $l(x)$ bits à direita da vírgula das expansões binárias obtidas em 3.

Código Shannon-Fano-Elias

Ideia Em vez de construir uma árvore para obter as palavras de código, estas são obtidas a partir da da expansão binária das probabilidades acumuladas.

Algoritmo

1. Calculam-se os comprimentos das palavras de código
$$l(x) = \lceil -log_2 p(x) \rceil + 1.$$
(Note que tem mais um bit do que o código de Shannon!)
2. Dado um alfabeto ordenado $\mathcal{X}$ e probabilidades $p(x)$, constrói-se a função de probabilidade acumulada modificada
$$\bar{F}(x) = \sum_{x_i < x} p(x_i) + \frac{1}{2} p(x).$$
3. Calcula-se a expansão binária de $\bar{F}(x)$ para cada $x \in \mathcal{X}$.
4. As palavras de código são os primeiros $l(x)$ bits à direita da vírgula das expansões binárias obtidas em 3.

Desempenho As palavras de código usam sempre mais um bit do que as do código de Shannon, portanto
$$H(X) + 1 \leq L(C) < H(X) + 2$$

Exemplo

Considere uma fonte com alfabeto $\mathcal{X} = \{A, B, C\}$ e as probabilidades indicadas na tabela. As palavras de código têm comprimentos

$$l(x) = \lceil -\log_2 p(x) \rceil + 1$$

e são obtidas da expansão binária de

$$\bar{F}(x) = \sum_{x_i < x} p(x_i) + \frac{1}{2} p(x).$$

Obtém-se

x	$p(x)$	$l(x)$	$\bar{F}(x)_{\text{dec}}$	$\bar{F}(x)_{\text{bin}}$	$C(x)$
A	0.1	5	0.05	0.000011001100 bin	00001
B	0.3	3	0.25	0.010000000000 bin	010
C	0.6	2	0.70	0.101100110011 bin	10

O comprimento médio é $L(C) = 2.6$ bits.

Exemplo

Considere uma fonte com alfabeto $\mathcal{X} = \{A, B, C\}$ e as probabilidades indicadas na tabela. As palavras de código têm comprimentos

$$l(x) = \lceil -\log_2 p(x) \rceil + 1$$

e são obtidas da expansão binária de

$$\bar{F}(x) = \sum_{x_i < x} p(x_i) + \frac{1}{2} p(x).$$

Obtém-se

x	$p(x)$	$l(x)$	$\bar{F}(x)_{\text{dec}}$	$\bar{F}(x)_{\text{bin}}$	$C(x)$
A	0.1	5	0.05	0.000011001100 bin	00001
B	0.3	3	0.25	0.010000000000 bin	010
C	0.6	2	0.70	0.101100110011 bin	10

O comprimento médio é $L(C) = 2.6$ bits.

- O código Shannon-Fano-Elias parece ter desvantagem sobre o Huffman, pois usa em média mais 1 bit por símbolo.
- A codificação aritmética vai eliminar essa desvantagem e mostrar que se obtém um código com desempenho superior.

Ideia Fazer um único bloco de n símbolos, onde n é o tamanho total da mensagem a comprimir, e aplicar o código S-F-E. Todo o ficheiro comprimido é uma única palavra de código.

Ideia Fazer um único bloco de n símbolos, onde n é o tamanho total da mensagem a comprimir, e aplicar o código S-F-E. Todo o ficheiro comprimido é uma única palavra de código.

Exemplo

Suponhamos que queremos comprimir a string "BANANA". Construindo uma tabela com todas as strings de tamanho 6 formadas pelos símbolos $\{A, B, N\}$ obtemos

$x_1x_2x_3x_4x_5x_6$	$p(x_1x_2x_3x_4x_5x_6)$	$F(x_1x_2x_3x_4x_5x_6)$	
$AAAAAA$	$P(AAAAAA)$	$F(AAAAAA)$	
$AAAAAB$	$P(AAAAAB)$	$F(AAAAAB)$	
...	...	...	
$BABABN$	$p(BANABN)$	$F(BANABN)$	
BANANA	p(BANANA)	F(BANANA)	$\leftarrow$ só queremos esta
$BANANB$	$p(BANANB)$	$F(BANANB)$	
...	...		
$NNNNNN$	$p(NNNNNN)$	$F(NNNNNN)$	

Ideia Fazer um único bloco de n símbolos, onde n é o tamanho total da mensagem a comprimir, e aplicar o código S-F-E. Todo o ficheiro comprimido é uma única palavra de código.

Exemplo

Suponhamos que queremos comprimir a string "BANANA". Construindo uma tabela com todas as strings de tamanho 6 formadas pelos símbolos $\{A, B, N\}$ obtemos

$x_1x_2x_3x_4x_5x_6$	$p(x_1x_2x_3x_4x_5x_6)$	$F(x_1x_2x_3x_4x_5x_6)$
$AAAAAA$	$P(AAAAAA)$	$F(AAAAAA)$
$AAAAAB$	$P(AAAAAB)$	$F(AAAAAB)$
...	...	...
$BABABN$	$p(BANABN)$	$F(BANABN)$
BANANA	p(BANANA)	F(BANANA) $\leftarrow$ só queremos esta
$BANANB$	$p(BANANB)$	$F(BANANB)$
...	...	
$NNNNNN$	$p(NNNNNN)$	$F(NNNNNN)$

A construção da tabela não é viável pois o tamanho da tabela cresce exponencialmente com o tamanho da string.
Felizmente, é possível calcular uma linha sem ter de construir a tabela! :)

Repare que a soma das probabilidades de todas as strings que começam por A é simplesmente $p(A)$

$$p(AAAAAA) + \cdots + p(ANNNNN) = \sum_{x_2 x_3 x_4 x_5 x_6} P(Ax_2x_3x_4x_5x_6) = P(A)$$

Repare que a soma das probabilidades de todas as strings que começam por A é simplesmente $p(A)$

$$p(AAAAAA) + \cdots + p(ANNNNN) = \sum_{x_2 x_3 x_4 x_5 x_6} P(Ax_2 x_3 x_4 x_5 x_6) = P(A)$$

O mesmo raciocínio estende-se aos outros casos, por exemplo

$$p(BANAAA) + \cdots + p(BANANN) = \sum_{x_5 x_6} p(BANAx_5 x_6) = p(BANA)$$

Repare que a soma das probabilidades de todas as strings que começam por A é simplesmente $p(A)$

$$p(AAAAAA) + \cdots + p(ANNNNN) = \sum_{x_2 x_3 x_4 x_5 x_6} P(Ax_2x_3x_4x_5x_6) = P(A)$$

O mesmo raciocínio estende-se aos outros casos, por exemplo

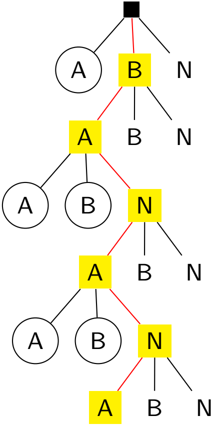
$$p(BANAAA) + \cdots + p(BANANN) = \sum_{x_5 x_6} p(BANAx_5x_6) = p(BANA)$$

Conhecendo as probabilidades

$$p(A) = \frac{1}{2} \qquad p(B) = \frac{1}{6} \qquad p(N) = \frac{1}{3}$$

pode-se facilmente calcular as probabilidades acima, por exemplo:

$$p(BANA) = p(A)^2 p(B) p(N) = \left(\frac{1}{2}\right)^2 \frac{1}{6}\frac{1}{3}$$



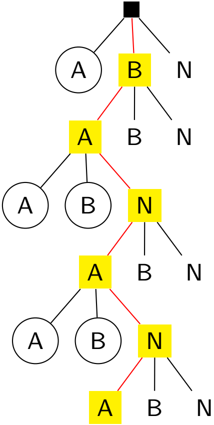
A probabilidade de cada caminho desde a raíz até cada uma das bolas representa a probabilidade de todas as strings com esse prefixo, e é fácil de calcular:

$$\bar{F}(BANANA) = p(A) + p(BAA) + p(BAB) + \\ + p(BANAA) + p(BANAB) + \frac{1}{2}p(BANANA)$$

Obtém-se

$$\bar{F}(x) = \frac{1}{2} + \left(\frac{1}{2}\right)^2 \frac{1}{6} + \frac{1}{2}\left(\frac{1}{6}\right)^2 + \\ + \left(\frac{1}{2}\right)^3 \frac{1}{6}\frac{1}{3} + \left(\frac{1}{2}\right)^2 \left(\frac{1}{6}\right)^2 \frac{1}{3} + \frac{1}{2}\left(\left(\frac{1}{2}\right)^3 \frac{1}{6}\left(\frac{1}{3}\right)^2\right) \\ = 0.565972222\ldots = 0.10010000111000_{(\text{bin})}\ldots$$

Codificação aritmética



A probabilidade de cada caminho desde a raíz até cada uma das bolas representa a probabilidade de todas as strings com esse prefixo, e é fácil de calcular:

$$\bar{F}(BANANA) = p(A) + p(BAA) + p(BAB) + \\ + p(BANAA) + p(BANAB) + \frac{1}{2}p(BANANA)$$

Obtém-se

$$\bar{F}(x) = \frac{1}{2} + \left(\frac{1}{2}\right)^2 \frac{1}{6} + \frac{1}{2}\left(\frac{1}{6}\right)^2 + \\ + \left(\frac{1}{2}\right)^3 \frac{1}{6}\frac{1}{3} + \left(\frac{1}{2}\right)^2 \left(\frac{1}{6}\right)^2 \frac{1}{3} + \frac{1}{2}\left(\left(\frac{1}{2}\right)^3 \frac{1}{6}\left(\frac{1}{3}\right)^2\right) \\ = 0.565972222\ldots = 0.10010000111000_{(\text{bin})}\ldots$$

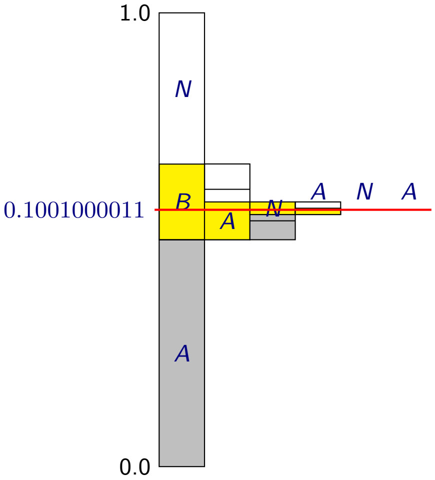
O ficheiro comprimido serão os primeiros n bits de $\bar{F}(X)$, onde

$$n = l(BANANA) = \lceil -log_2 p(BANANA) \rceil + 1 = 10 \text{ bits}$$

Finalmente obtém-se a string comprimida

$$C(BANANA) = 1001000011$$

Outra forma equivalente consiste em subdividir sucessivamente o intervalo $[0, 1[$ proporcionalmente às probabilidades dos símbolos:



As probabilidades acumuladas $F(A) = P(A) + P(BAA) + P(BAB) + \cdots$ que se obtiveram antes, correspondem às regiões a cinzento. A probabilidade $\frac{1}{2}p(BANANA)$ corresponde ao meio da última região amarela.

Aspectos positivos:

- Desempenho superior ao código de Huffman, uma vez que existe um único arredondamento no tamanho, enquanto o Huffman tem geralmente arredondamentos dos tamanhos em todos os símbolos do ficheiro.

Aspectos negativos:

- Presume que os cálculos são feitos com precisão infinita. A probilidade acumulada teria de ser calculada com precisão suficiente (nº de bits) para acomodar todo o ficheiro comprimido.
- Por este motivo, não pode ser implementado tal como descrito anteriormente.