

8.4 Compresión libre de errores

En algunas aplicaciones la compresión libre de errores es la única alternativa aceptable. En esta sección veremos las principales estrategias de compresión libre de errores en uso. Estas están generalmente compuestas de 2 operaciones relativamente independientes: el diseño de una representación por reducción de redundancia entre píxeles y codificación de la representación para eliminar la redundancia de código.

8.4.1 Codificación por longitud variable

El enfoque más simple de compresión sin errores es reducir solamente la redundancia de código. Para hacerlo construiremos un código de longitud variable que asigne las palabras más cortas a los niveles de gris más probables.

Hay varias maneras de construir tal código. Recuérdese que, en la práctica, los símbolos de entrada pueden ser tanto los niveles de gris como el resultado del mapeo para reducir la redundancia entre píxeles.

El código de Huffman

La técnica más popular para remover redundancia de código se le debe a Huffman. Al codificar individualmente los símbolos de una fuente de información, el código de Huffman obtiene el menor número posible de símbolos de código por símbolo de la fuente.

El primer paso es crear una serie de reducciones de fuente ordenando las probabilidades de los símbolos y combinando los símbolos de menor probabilidad en un símbolo único que los reemplaza en la siguiente reducción de fuente. En la figura se muestra el proceso.

Original source		Source reduction			
Symbol	Probability	1	2	3	4
a_2	0.4	0.4	0.4	0.4	0.6 0.4
a_6	0.3	0.3	0.3	0.3	
a_1	0.1	0.1	0.2	0.3	
a_4	0.1	0.1			
a_3	0.06	0.1			
a_5	0.04				

Reducciones de fuente de Huffman

A la izquierda, se muestran los símbolos ordenados de mayor a menor probabilidad. Para la primera reducción de código, se unen 0.06 y 0.04 y se combinan para formar un símbolo compuesto de probabilidad 0.1. Se repite el proceso con la columna resultante hasta que obtenemos una fuente reducida de 2 símbolos.

El segundo paso del procedimiento de Huffman es codificar cada fuente reducida, empezando con la menor hasta llegar a la original.

Original source			Source reduction			
Sym.	Prob.	Code	1	2	3	4
a_2	0.4	1	0.4 1	0.4 1	0.4 1	0.6 0
a_6	0.3	00	0.3 00	0.3 00	0.3 00	0.4 1
a_1	0.1	011	0.1 011	0.2 010	0.3 01	
a_4	0.1	0100	0.1 0100	0.1 011		
a_3	0.06	01010	0.1 0101			
a_5	0.04	01011				

Asignación de códigos

El código binario mínimo para una fuente de 2 símbolos está formado por los símbolos 0 y 1. Como se muestra en la figura, estos símbolos se asignan a los 2 símbolos de la derecha (no importa a cuál). Ya que el símbolo de probabilidad 0.6 se generó combinando 2 símbolos en la fuente reducida a su izquierda, el 0 utilizado para codificarlo se usa ahora para los 2 símbolos que lo formaron, y un 0 y 1 se asignan arbitrariamente después de este para distinguirlos. La operación se repite hasta alcanzar la fuente original. El código final se muestra en la figura.

Podemos ver que la longitud promedio del código es

$$L_{\text{prom}} = (0.4)(1) + (0.3)(2) + (0.1)(3) + (0.1)(4) + (0.06)(5) + (0.04)(5) \\ = 2.2 \text{ bits /símbolo}$$

El procedimiento de Huffman crea el código óptimo para un conjunto de símbolos y probabilidades si los símbolos se codifican de uno por uno.

La codificación y decodificación se hace por medio de una tabla. El código es un código de bloque que puede ser decodificado instantáneamente de manera única.

(ejemplo, decodificar 010100111100)

Otros códigos de longitud variable casi óptimos

Cuando se tiene una gran cantidad de símbolos a codificar, la construcción de un código binario de Huffman no es una tarea trivial. Para el caso general de J símbolos de la fuente, se requieren $J - 2$ reducciones de fuente y $J - 2$ asignaciones.

Dada tal complejidad computacional, a veces se requiere sacrificar eficiencia de código por simplicidad en la construcción de código.

En la tabla se ilustran 4 alternativas. En el último renglón observamos que la longitud promedio del código de Huffman es menor que la de los demás. El código binario natural tiene el promedio más grande.

Source symbol	Probability	Binary Code	Huffman	Truncated Huffman	B ₂ -Code	Binary Shift	Huffman Shift
<i>Block 1</i>							
<i>a</i> ₁	0.2	00000	10	11	C00	000	10
<i>a</i> ₂	0.1	00001	110	011	C01	001	11
<i>a</i> ₃	0.1	00010	111	0000	C10	010	110
<i>a</i> ₄	0.06	00011	0101	0101	C11	011	100
<i>a</i> ₅	0.05	00100	00000	00010	C00C00	100	101
<i>a</i> ₆	0.05	00101	00001	00011	C00C01	101	1110
<i>a</i> ₇	0.05	00110	00010	00100	C00C10	110	1111
<i>Block 2</i>							
<i>a</i> ₈	0.04	00111	00011	00101	C00C11	111 000	00 10
<i>a</i> ₉	0.04	01000	00110	00110	C01C00	111 001	00 11
<i>a</i> ₁₀	0.04	01001	00111	00111	C01C01	111 010	00 110
<i>a</i> ₁₁	0.04	01010	00100	01000	C01C10	111 011	00 100
<i>a</i> ₁₂	0.03	01011	01001	01001	C01C11	111 100	00 101
<i>a</i> ₁₃	0.03	01100	01110	100000	C10C00	111 101	00 1110
<i>a</i> ₁₄	0.03	01101	01111	100001	C10C01	111 110	00 1111
<i>Block 3</i>							
<i>a</i> ₁₅	0.03	01110	01100	100010	C10C10	111 111 000	00 00 10
<i>a</i> ₁₆	0.02	01111	010000	100011	C10C11	111 111 001	00 00 11
<i>a</i> ₁₇	0.02	10000	010001	100100	C11C00	111 111 010	00 00 110
<i>a</i> ₁₈	0.02	10001	001010	100101	C11C01	111 111 011	00 00 100
<i>a</i> ₁₉	0.02	10010	001011	100110	C11C10	111 111 100	00 00 101
<i>a</i> ₂₀	0.02	10011	011010	100111	C11C11	111 111 101	00 00 1110
<i>a</i> ₂₁	0.01	10100	011011	101000	C00C00C00	111 111 110	00 00 1111
<i>Entropy</i>	4.0						
<i>Average length</i>		5.0	4.05	4.24	4.65	4.59	4.13

Huffman truncado

La columna 5 ilustra una modificación simple del código Huffman conocida como *código Huffman truncado*. Un código de Huffman truncado se genera codificando sólo los PSI símbolos más probables de la fuente, siendo PSI algún entero positivo menor a J. Un código de prefijo seguido por un código de longitud variable se usa para representar los demás símbolos de la fuente. En la tabla PSI = 12 y el código de prefijo se genera como la 13ava palabra de código de Huffman (es decir, un símbolo de prefijo cuya probabilidad es la suma de las probabilidades de los símbolos *a*₁₃ a *a*₂₁ se incluye como el 13avo símbolo durante la codificación de Huffman de los 12 símbolos más probables, los 9 restantes se codifican usando el código de prefijo, en este caso 10, y un valor binario de 4 bits igual al subíndice del símbolo menos 13).

Código B

Cada palabra de código se hace a partir de bits de *continuación*, denotados C, y bits de información. El único propósito de los bits de continuación es separar palabras de código individuales, así que estos simplemente alternan entre 0 y 1 para cada palabra de código en una cadena. El código que se muestra es llamado código B₂, ya que se usan 2 bits de información por bit de continuación. La secuencia de códigos B₂ que

corresponde a la cadena de símbolos de la fuente $a_{11}a_2a_7$ es 001 010 101 000 010 o 101 110 100 110, dependiendo de si el primer bit de continuación se toma como 0 o 1.

Shift codes

Un *shift code* se genera

- 1) Ordenando los símbolos de la fuente de manera que sus probabilidades decrezcan (de manera monótona).
- 2) Dividiendo el número total de símbolos en bloques de símbolos del mismo tamaño.
- 3) Codificando los elementos individuales en cada bloque de manera idéntica.
- 4) Añadiendo símbolos especiales de *shift up* y / o *shift down* para identificar cada bloque.

Cuando se reconoce un símbolo *shift up* o *shift down* en el decodificador, se mueve un bloque arriba o abajo respecto al bloque de referencia predefinido.

Para generar el *shift code* de 3 bits de la columna 7, los símbolos de la fuente se ordenan por probabilidad de ocurrencia y se dividen en 3 bloques de 7 símbolos. Los símbolos individuales (a_1 a a_7) del bloque superior (el bloque de referencia) se codifican con los símbolos binarios 000 a 110. El octavo código (111) no se incluye en el bloque de referencia, pero se usa como un control de *shift up* que identifica a los bloques restantes (en este caso no se utiliza un símbolo *shift down*).

Los símbolos en los bloques restantes se codifican con uno o dos símbolos *shiftpup* en combinación con los códigos binarios que representan el bloque de referencia.

Por ejemplo, el símbolo de la fuente a_{19} se codifica como 111 111 100.

El *shift code* de Huffman en la columna 8 se genera de manera similar. La diferencia es en la asignación de la probabilidad al símbolo *shift* antes de codificar con Huffman el bloque de referencia. Normalmente esta asignación se hace sumando la probabilidad de todos los símbolos de fuente fuera del bloque de referencia (igual que en el código de Huffman truncado). Aquí se suman los símbolos a_8 a a_{21} y es 0.39. El símbolo *shift* es entonces el símbolo más probable y se le asigna una de las palabras Huffman más cortas (00).