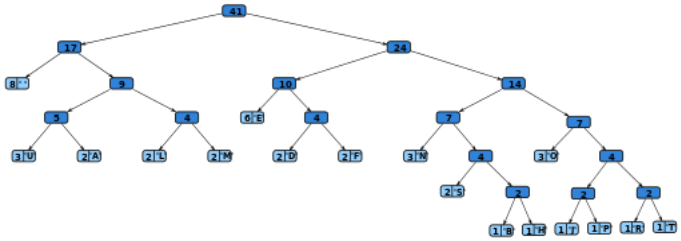
WIKIPEDIA

Codificación Huffman

En ciencias de la computación y teoría de la información, la **codificación Huffman** es un algoritmo usado para compresión de datos. El término se refiere al uso de una tabla de códigos de longitud variable para codificar un determinado símbolo (como puede ser un carácter en un archivo), donde la tabla ha sido rellenada de una manera específica basándose en la probabilidad estimada de aparición de cada posible valor de dicho símbolo. Fue desarrollado por David A. Huffman mientras era estudiante de doctorado en el MIT, y publicado en "A Method for the Construction of Minimum-Redundancy Codes".

La codificación Huffman usa un método específico para elegir la representación de cada símbolo, que da lugar a un código prefijo (es decir, la cadena de bits que representa a un símbolo en particular nunca es prefijo de la cadena de bits de un símbolo distinto) que representa los caracteres más comunes usando las cadenas de bits más cortas, y viceversa. Huffman fue capaz de diseñar el método de compresión más eficiente de este tipo: ninguna representación alternativa de un conjunto de símbolos de entrada produce una salida media más pequeña cuando las frecuencias de los símbolos coinciden con las usadas para crear el código. Posteriormente se encontró un método para llevar esto a cabo en un tiempo lineal si las probabilidades de los símbolos de entrada (también conocidas como "pesos") están ordenadas.

Para un grupo de símbolos con una distribución de probabilidad uniforme y un número de miembros que es potencia de dos, la codificación Huffman es equivalente a una codificación en bloque binaria, por ejemplo, la codificación ASCII. La codificación Huffman es un método para crear códigos prefijo tan extendido que el término "codificación Huffman" es ampliamente usado como sinónimo de "código prefijo", incluso cuando dicho código no se ha producido con el algoritmo de Huffman.



Árbol de Huffman generado para las frecuencias de apariciones exactas del texto "ESTO ES UN EJEMPLO DE UN ARBOL DE HUFFMAN". Las frecuencias y códigos de cada carácter se muestran abajo. Codificar esta frase de 41 caracteres usando este código requiere 156 bits (sin contar con el espacio para el árbol) cuando con bytes de 8 bits requiere 328 bits.

Carácter	Frecuencia	Código
Espacio	8	00
E	6	100
N	3	1100
O	3	1110
U	3	0100
A	2	0101
D	2	1010
F	2	1011
L	2	0110
M	2	0111
S	2	11010
B	1	110110
H	1	110111
J	1	111100
P	1	111101
R	1	111110
T	1	111111

Aunque la codificación de Huffman es óptima para una codificación símbolo a símbolo dada una distribución de probabilidad, su optimalidad a veces puede verse accidentalmente exagerada. Por ejemplo, la codificación aritmética y la codificación LZW normalmente ofrecen mayor capacidad de compresión. Estos dos métodos pueden agrupar un número arbitrario de símbolos para una codificación más eficiente, y en general se adaptan a las estadísticas de entrada reales. Este último es útil cuando las probabilidades no se conocen de forma precisa o varían significativamente dentro del flujo de datos.

Índice

Historia

En 1951, a David Huffman y a sus compañeros de clase de la asignatura "Teoría de la Información" se les permitió optar entre la realización de un examen final o la presentación de un trabajo. El profesor Robert. M. Fano asignó las condiciones del trabajo bajo la premisa de encontrar el código binario más eficiente. Huffman, ante la imposibilidad de demostrar qué código era más eficiente, se rindió y empezó a estudiar para el examen final. Mientras estaba en este proceso vino a su mente la idea de usar árboles binarios de frecuencia ordenada y rápidamente probó que éste era el método más eficiente.

Con este estudio, Huffman superó a su profesor, quien había trabajado con el inventor de la teoría de la

información Claude Shannon con el fin de desarrollar un código similar. Huffman solucionó la mayor parte de los errores en el algoritmo de codificación Shannon-Fano. La solución se basaba en el proceso de construir el árbol de abajo a arriba en vez de al contrario.

Definición del problema

Descripción informal

Dados

Un conjunto de símbolos y sus pesos (normalmente proporcionales a probabilidades).

Encontrar

Un código binario prefijo (un conjunto de elementos del código) con longitud de palabra esperada mínima (de forma equivalente, un árbol con longitud del camino mínima).

Descripción formal

Entradas

El alfabeto $A = \{a_1, a_2, \cdots, a_n\}$, que es el alfabeto de símbolos de tamaño n.
El conjunto $W = \{w_1, w_2, \cdots, w_n\}$, que es el conjunto de pesos (positivos) de los símbolos (normalmente proporcionales a probabilidades), es decir $w_i = \text{peso}(a_i), 1 \leq i \leq n$.

Salida

El código $C(A, W) = \{c_1, c_2, \cdots, c_n\}$, que es el conjunto de elementos del código (binario), donde c_i es la palabra del código para $a_i, 1 \leq i \leq n$.

Objetivo

Sea $L(C) = \sum_{i=1}^{n} w_i \times \text{longitud}(c_i)$ la longitud del camino ponderado del código C. Condición: $L(C) \leq L(T)$ para cualquier código $T(A, W)$.

Ejemplo

Entrada (*A*, *W*)	Símbolo (a_i)	a	b	c	d	e	**Suma**
	Peso (w_i)	0.10	0.15	0.30	0.16	0.29	= 1
Salida *C*	Palabras del código (c_i)	010	011	11	00	10	
	Longitud de la palabra (en bits) (l_i)	3	3	2	2	2	
	Longitud del camino ponderado ($l_i\ w_i$)	0.30	0.45	0.60	0.32	0.58	$L(C)$ = 2.25
Optimalidad	Probabilidad (2^{-l_i})	1/8	1/8	1/4	1/4	1/4	= 1.00
	Cantidad de información (en bits) ($-\mathbf{log}_2\ w_i$) ≈	3.32	2.74	1.74	2.64	1.79	
	Entropía ($-w_i\ \mathbf{log}_2\ w_i$)	0.332	0.411	0.521	0.423	0.518	$H(A)$ = 2.205

Para cualquier código *biunívoco*, aquel código *decodificable de forma única*, la suma de las probabilidades de todos los símbolos es siempre menor o igual que uno. En este ejemplo, es exactamente igual a uno; por lo que decimos que es un código *completo*. Si no es el caso siempre se puede derivar un código equivalente añadiendo símbolos extra (con probabilidades nulas asociadas), para hacer el código completo a la vez que se mantiene *biunívoco*.

Tal como definió Shannon (1948), la cantidad de información h (en bits) de cada símbolo a_i con probabilidad no nula w_i es

$$h(a_i) = \log_2 \frac{1}{w_i}.$$

La entropía H (en bits) es la suma ponderada, de todos los símbolos a_i con probabilidad no nula w_i, de la cantidad de información de cada símbolo:

$$H(A) = \sum_{w_i>0} w_i h(a_i) = \sum_{w_i>0} w_i \log_2 \frac{1}{w_i} = -\sum_{w_i>0} w_i \log_2 w_i.$$

(Nota: un símbolo con probabilidad cero tiene una contribución nula a la entropía. Cuando w = 0, $w \log_2(1/w) = 0 \cdot \infty$ es una indeterminación; aplicando la regla de L'Hôpital :

$$\lim_{w\to 0^+} \frac{\log_2 \frac{1}{w}}{\frac{1}{w}} = \lim_{w\to 0^+} \frac{-\frac{1}{w \ln 2}}{-\frac{1}{w^2}} = \lim_{w\to 0^+} \frac{w}{\ln 2} = 0.$$

Por simplicidad, los símbolos con probabilidad nula han sido dejados fuera de la fórmula anterior).

Como consecuencia del teorema de codificación de fuente de Shannon, la entropía es una medida de la longitud de palabra más pequeña del código que es teóricamente posible para un alfabeto dado con unos pesos asociados. En este ejemplo, la longitud media de la palabra es 2,25 bits por símbolo, ligeramente mayor

que la entropía calculada de 2,205 bits por símbolo. Así que no solo este código es óptimo en el sentido de que ningún otro código posible funciona mejor, sino que además está muy cercano al límite teórico establecido por Shannon.

Nótese que, en general, un código Huffman no necesita ser único, pero si lo es siempre es uno de los códigos que minimiza $L(C)$.

Técnica básica

La técnica utilizada es el propio algoritmo de Huffman. Consiste en la creación de un árbol binario en el que se etiquetan los nodos hoja con los caracteres, junto a sus frecuencias, y de forma consecutiva se van uniendo cada pareja de nodos que menos frecuencia sumen, pasando a crear un nuevo nodo intermedio etiquetado con dicha suma. Se procede a realizar esta acción hasta que no quedan nodos hoja por unir a ningún nodo superior, y se ha formado el árbol binario.

Posteriormente se etiquetan las aristas que unen cada uno de los nodos con ceros y unos (hijo derecho e izquierdo, respectivamente, por ejemplo). El código resultante para cada carácter es la lectura, siguiendo la rama, desde la raíz hacia cada carácter (o viceversa) de cada una de las etiquetas de las aristas.

Construcción del árbol

Para obtener los códigos de Huffman hay que construir un árbol binario de nodos, a partir de una lista de nodos, cuyo tamaño depende del número de símbolos, n. Los nodos contienen dos campos, el **símbolo** y el **peso** (frecuencia de aparición).

Cada nodo del árbol puede ser o bien un nodo hoja o un nodo interno. Inicialmente se considera que todos los nodos de la lista inicial son nodos hoja del árbol. Al ir construyendo el árbol, los nodos internos tendrán un **peso** y dos nodos hijos, y opcionalmente un enlace al nodo padre que puede servir para recorrer el árbol en ambas direcciones. Por convención el bit '0' se asocia a la rama izquierda y el bit '1' a la derecha. Una vez finalizado el árbol contendrá n nodos hijo y $n - 1$ nodos internos.

El proceso de construcción del árbol comienza formando un nodo intermedio que agrupa a los dos nodos hoja que tienen menor **peso** (frecuencia de aparición). El nuevo nodo intermedio tendrá como nodos hijos a estos dos nodos hoja y su campo **peso** será igual a la suma de los **pesos** de los nodos hijos. Los dos nodos hijos se eliminan de la lista de nodos, sustituyéndolos por el nuevo nodo intermedio. El proceso se repite hasta que solo quede un nodo en la lista. Este último nodo se convierte en el nodo raíz del árbol de Huffman.

El algoritmo de construcción del árbol puede resumirse así:

1. Crear un nodo hoja para cada símbolo, asociando un peso según su frecuencia de aparición e insertarlo en la lista ordenada ascendentemente.
2. Mientras haya más de un nodo en la lista:
    1. Eliminar los dos nodos con menor probabilidad de la lista.
    2. Crear un nuevo nodo interno que enlace a los nodos anteriores, asignándole como peso la suma de los pesos de los nodos hijos.
    3. Insertar el nuevo nodo en la lista, (en el lugar que le corresponda según el peso).
3. El nodo que quede es el nodo raíz del árbol.

Aquí hay un ejemplo de construcción del árbol a partir del texto en francés "j'aime aller sur le bord de l'eau les jeudis ou les jours impairs":

Propiedades principales

Las probabilidades usadas pueden ser genéricas para el dominio de la aplicación, que están basadas en el caso promedio, o pueden ser las frecuencias reales encontradas en el texto que se está comprimiendo. (Esta variación requiere que la tabla de frecuencias u otra estructura utilizada para la codificación deben ser almacenadas con el texto comprimido; las implementaciones emplean varios mecanismos para almacenar tablas de manera eficiente).

La codificación Huffman es óptima cuando la probabilidad de cada símbolo de entrada es una potencia negativa de dos. Los códigos prefijos tienden a ser ligeramente ineficientes en alfabetos pequeños, donde las probabilidades normalmente se encuentran entre esos puntos óptimos. El "empaquetado", o expansión del tamaño del alfabeto concatenando múltiples símbolos en "palabras" de tamaño fijo o variable antes de la codificación Huffman, normalmente ayuda, especialmente cuando símbolos adyacentes están correlacionados (como en el caso de un texto en lenguaje natural). El peor caso para una codificación Huffman puede darse

cuando la probabilidad de un símbolo excede $2^{-1} = 0.5$, haciendo el límite superior de ineficiencia ilimitado. Estas situaciones a menudo responden bien a una forma de paquete llamada codificación run-length.

La codificación aritmética produce una ligera ganancia sobre la codificación Huffman, pero en la práctica esta ganancia raramente ha sido lo bastante grande como para utilizar la codificación aritmética que posee una complejidad computacional más elevada y además requiere el pago de royalties. (A julio de 2006, IBM posee patentes de muchos métodos de codificación aritmética en varias jurisdicciones).

Variación

Existen muchas variaciones del código de Huffman, algunos que utilizan Huffman como algoritmo, y otros que encuentra el código prefijo óptimo. Tenga en cuenta que en este último caso el método no es necesariamente similar al de Huffmans y no tiene por qué terminar en tiempo polinómico.

Código Huffman n-ario

El algoritmo n-ario de Huffman usa el alfabeto {0,1,...,n-1} para codificar el mensaje y construir un árbol n-ario. Este enfoque fue considerado por Huffman en su enfoque originario.

Código Huffman adaptable

La variación llamada código de huffman adaptable calcula dinámicamente la probabilidad de la frecuencia de la cadena de origen basada en antiguas apariciones. Está relacionado con la familia de algoritmos LZ.

Algoritmo de Huffman de plantilla

La mayoría de las veces, el tamaño de las implementaciones del código de Huffman están representadas por probabilidades numéricas, pero el algoritmo no lo exige; se requiere solo una manera de ordenar el tamaño y añadirle. El algoritmo de plantilla de Huffman permite utilizar cualquier tipo de tamaño (costos, frecuencias, los pares del tamaño, tamaños no numéricos) y uno de los muchos que combina métodos (no solo la adición). Tales algoritmos pueden resolver problemas de minimización, como la minimización de Max[Wi + C(i)], un problema que se aplicó por primera vez en el diseño de circuitos.

Código de Huffman de tamaño limitado

El Código de Huffman de tamaño de limitado es una variante donde el objetivo es lograr que el camino de coste mínimo con la restricción de que la longitud de cada palabra sea menor que una constante. El algoritmo de package-merge lo soluciona con un algoritmo voraz, muy similar al usado por el algoritmo de Huffman. Su complejidad es del orden de O (nL), siendo L el tamaño de la palabra más larga. No se conoce algoritmo para resolver este problema en tiempo lineal, a diferencia de los problemas convencionales de Huffman.

Codificación Huffman con costes desiguales

En el problema estándar de la codificación Huffman, se asume que cada símbolo del alfabeto con el que se construye cada palabra del código tiene igual costo de transmisión: una palabra del código cuya longitud sea

N dígitos siempre tendrá un costo de *N*, sin importar cuántos de esos dígitos sean ceros, cuántos unos, etc. Cuando se trabaja bajo esta suposición, minimizar el costo total del mensaje y minimizar el número total de dígitos es lo mismo.

En la *codificación Huffman con costes desiguales* la suposición anterior ya no es verdadera: los símbolos del alfabeto pueden tener longitudes no uniformes, debido a características del medio de transmisión. Un ejemplo es el alfabeto del código Morse, donde una 'raya' requiere más tiempo para ser enviada que un 'punto', y por lo tanto el costo del tiempo de transmisión de una raya es mayor. El objetivo sigue siendo minimizar la longitud media de la palabra de código, pero no es suficiente con minimizar el número de símbolos usado en el mensaje. No se conoce un algoritmo para solucionar esto de la misma manera o con la misma eficiencia que la codificación Huffman convencional.

Árboles binarios alfabéticos óptimos (codificación Hu-Tucker)

En una situación de codificación Huffman estándar, se asume que cualquier código puede corresponderse con cualquier símbolo de entrada. En la versión *alfabética*, el orden alfabético de las entradas y salidas debe ser idéntico. Así, por ejemplo, a la entrada $A = \{a, b, c\}$ no se le puede asignar $H(A, C) = \{00, 1, 01\}$, sino que le correspondería $H(A, C) = \{00, 01, 1\}$ o $H(A, C) = \{0, 10, 11\}$. Esto también se conoce como el problema de **Hu-Tucker**, por los autores de la publicación que contiene la primera solución *linearítmica* a este problema de optimalidad binaria alfabética, que es similar al algoritmo de Huffman, pero no es una variación del mismo. Estos árboles binarios alfabéticos óptimos son usados a menudo como áboles binarios de búsqueda.

Código canónico de Huffman

Si los pesos correspondientes a las entradas (ordenadas alfabéticamente) están en orden numérico, los códigos de Huffman tienen la misma longitud que los códigos alfabético óptimos, así que pueden calcularse como estas últimas, haciendo que la codificación Hu-Tucker sea innecesaria. El código resultante de las entradas (re) ordenadas numéricamente se conoce como código canónico de Huffman y es el código que normalmente se usa en la práctica, dada su facilidad para codificar y decodificar. La técnica para encontrar este código se conoce como **codificación de Huffman-Shannon-Fano**, ya que es óptima como la codificación de Huffman, y alfabética según la probabilidad de los pesos, como la codificación de Shannon-Fano.

De esta manera, el método de **codificación de Huffman-Shannon-Fano** ira asignando gradualmente los códigos prefijo más cortos a aquellos símbolos de mayor peso, y los más largos a los de menor peso. Es Decir, ordenará descendentemente los símbolos según su peso, luego codificará según el método de Codificación Shannon-Fano, una vez obtenida la codificación se procede al armado del un Diagrama de flujo que comienza preguntando si el valor del primer bit del prefijo obtenido para el símbolo buscado es 0, del cual salen las 2 ramas, una para la afirmación y otra para la negación; luego seguirá preguntando para el segundo bit y derivará, para cada rama, a otra decisión o a un símbolo, según corresponda.

Aplicaciones

La codificación aritmética puede considerarse como una generalización de la codificación de Huffman, de hecho, en la práctica la codificación Aritmética viene precedida por la codificación de huffman, pues es más fácil encontrar una aritmética para una entrada binaria que para una no binaria. Por otra parte aunque la codificación de compresión ofrece mejor rendimiento que la codificación de Huffman, la codificación de

Huffman se encuentra todavía en uso generalizado debido a su simplicidad, alta velocidad, y falta de problemas de patentes.

La codificación de Huffman se utiliza a menudo en algún otro método de compresión. Como la deflación y códec multimedia como JPEG y MP3 que tienen una cuantificación digital basada en la codificación de Huffman.

Ejemplo

Una sonda espacial ha sido lanzada al espacio para contar cierto tipo de perturbaciones estelares. Ha de contar cuántas se producen en cada minuto, y tiene cada día una ventana de tiempo bastante reducida para enviar los datos a Tierra; por tanto, interesa reducir al máximo el tiempo de transmisión, y para ello se recurre a codificar las muestras mediante un código de Huffman.

En la siguiente tabla se muestran los valores a transmitir, junto con sus frecuencias relativas, su código en una codificación binaria de 3 bits, y su código en un posible código Huffman para estos valores.

Valor	Frecuencia	Código binario	Código Huffman
0	10%	000	010
1	20%	001	10
2	30%	010	00
3	25%	011	11
4	10%	100	0110
5 o más	5%	101	0111

Puede observarse que, en la codificación binaria, todos los posibles valores reciben códigos del mismo número de bits, mientras que en la codificación Huffman, cada valor tiene un número diferente de bits: los códigos más frecuentes poseen dos bits, mientras que los menos frecuentes poseen cuatro bits.

A continuación se observa el código necesario para transmitir la siguiente serie de valores:

```
5,4,2,3,2,2,1,0,1,3,2,4,3,4,3,2,3,4,2,4
```

Utilizando la codificación binaria, sería una serie de 60 bits; es decir, 3 bits por símbolo.

```
10110001001101001000100000101101010001110001101001110001010 0
```

nota: se ha añadido la misma serie separada en bloques con la única razón de facilitar una transcripción manual libre de errores para un estudio por parte del lector interesado.

```
101.100.010.011.010.010.001.000.001.011.010.100.011.100.011.010.011.100.010.100
```

Utilizando, en cambio, la codificación Huffman, se tendría que enviar una secuencia de 53 bits; es decir, 2,65 bits por símbolo.

```
01110110001100001001010110001101101101100110110000110
```

nota: la misma serie dividida en bloques de 4 bits para la misma observación anterior.

```
0111.0110.0011.0000.1001.0101.1000.1101.1011.0110.0110.1100.0011.0
```

En este ejemplo, la media de bits por símbolo que cabría esperar de esta codificación, en cadenas de valores más largas, es de 2,4.

Para su comparación, la entropía del conjunto de símbolos es de 2,366; es decir, el mejor método de compresión sería capaz de codificar estos valores utilizando 2,366 bits por símbolo.

Es posible, también, apreciar cómo se pueden extraer sin ninguna ambigüedad los valores originales a partir de la cadena codificada mediante Huffman.

Hay que añadir que la codificación de Huffman no puede ser aplicada a imágenes en blanco y negro porque es incapaz de producir compresión sobre un alfabeto binario.

Bibliografía

- D.A. Huffman, "A method for the construction of minimum-redundancy codes", Proceedings of the I.R.E., sept 1952, pp 1098-1102

Véase también

- Algoritmo de Huffman
- Código canónico de Huffman
- Codificación aritmética

Enlaces externos

- Generador de árboles de Huffman (http://huffman.ooz.ie)
- Huffman en PHP (http://mmengineer.blogspot.com/2007/11/algoritmo-de-compresion-huffman-php.html)

Obtenido de «https://es.wikipedia.org/w/index.php?title=Codificación_Huffman&oldid=129145176»

Esta página se editó por última vez el 9 sep 2020 a las 03:11.

El texto está disponible bajo la Licencia Creative Commons Atribución Compartir Igual 3.0; pueden aplicarse cláusulas adicionales. Al usar este sitio, usted acepta nuestros términos de uso y nuestra política de privacidad.
Wikipedia® es una marca registrada de la Fundación Wikimedia, Inc., una organización sin ánimo de lucro.