COMPRESIÓN SIN PÉRDIDAS

Federico Miyara

Formato FLAC

FLAC (Free Lossless Audio Codec, codificador-decodificador de audio sin pérdidas libre[1]) es un formato comprimido sin pérdida que es conveniente para el almacenamiento permanente o documental y para la transmisión de grandes cantidades de audio, ya que permite ahorrar hasta alrededor de un 50 % de capacidad de memoria o de ancho de banda del canal de comunicación, por ejemplo una red de área local inalámbrica o Internet (Salomon, 2007; Marengo et al., 2011; Roveri, 2011; Raynaudo et al., 2013). A diferencia del formato WAV, requiere el uso de un *códec* (codificador-decodificador) para comprimir la señal original PCM y posteriormente recuperarla cuando se requiera.

El método de compresión consiste en subdividir la señal en bloques temporales, aproximar en cada uno de ellos la señal con un modelo predictivo que utilice pocos parámetros, que se incluyen en un encabezamiento, y luego codificar el error de predicción, es decir la diferencia entre la señal original y su aproximación, que es más pequeña que la señal y por lo tanto requiere menos bits.

Esta descripción, sin embargo, está muy simplificada, ya que lo que en realidad sucede es que *estadísticamente* hay más valores pequeños que altos. En una codificación PCM, la cantidad de bits está determinada por el máximo valor a representar, por lo cual, con que haya un solo error de máxima amplitud ya será necesario utilizar la máxima cantidad de bits por muestra.

En lugar de esto conviene utilizar un *código de longitud variable* que utilice menos bits para los valores más frecuentes, es decir, los más pequeños. La elección de tal código queda determinada por la distribución estadística de los valores a codificar. Se ha encontrado empíricamente que, para los modelos predictivos típicos, los errores (denominados *residuos*) responden aproximadamente a una distribución de Laplace[2] (ver figura 52) cuya función de densidad de probabilidad es

$$f(x) = \frac{1}{2b} e^{-\frac{|x - \mu|}{b}}, \tag{1}$$

donde μ es la media y b es un parámetro de escala proporcional al desvío estándar:

$$\sigma = \sqrt{2}\, b\,. \tag{2}$$

Código de Golomb-Rice

Para una variable con distribución laplaciana, uno de los mejores códigos es el de Golomb-Rice. Se trata de un código híbrido, en el cual se divide el número a codificar por un divisor D convenientemente elegido (en general una potencia de 2). El cociente entero se representa en código unario y el resto en binario. En principio, el código *unario* representa los números con un solo símbolo repetido tantas veces como el número a representar. Sin embargo, dado que su longitud es variable y a priori

[1] Este formato se inscribe en la corriente del *software libre*, es decir un software de libre licencia, código abierto y distribución gratuita

[2] Estrictamente, la distribución no es laplaciana, porque al estar trabajando con variables discretas, el error es discreto. Lo que se observa es que el histograma tiene una forma similar a la distribución de Laplace.

desconocida, se necesita un símbolo diferente que represente la finalización del número. En versión tipográfica el símbolo podría ser un espacio, pero en un sistema binario el símbolo principal y el de separación deben elegirse entre el 0 y el 1. Si elegimos el 0 como el símbolo a repetir y el 1 como terminación, el 3 se representaría como 0001.

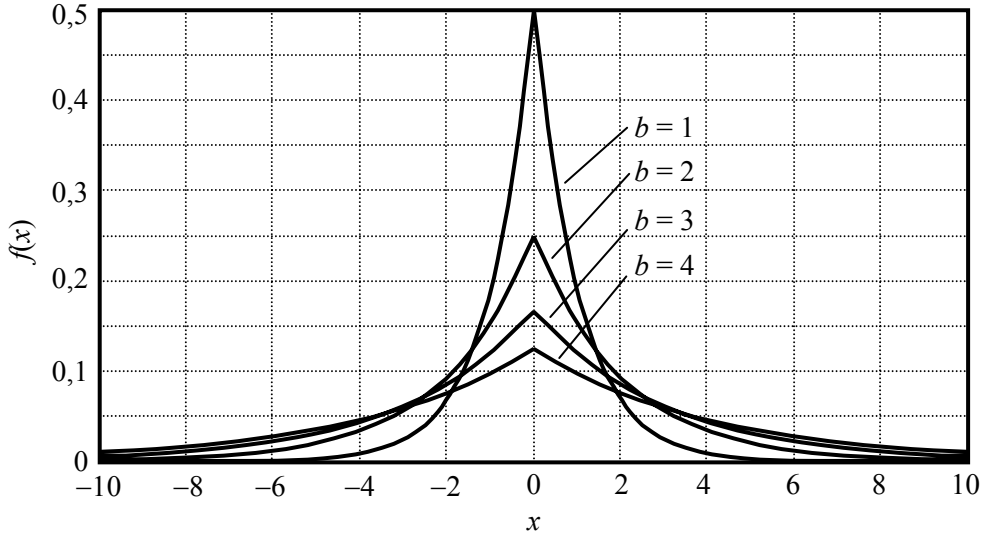

Figura 52. Función de densidad de probabilidad de una distribución de Laplace de media 0 para cuatro valores del parámetro *b*.

En la tabla 3.7 se muestra, como ejemplo, el código de Golomb-Rice con divisor $D = 4$ para números entre 0 y 15. Vemos que si bien los valores más altos requieren 6 bits cuando podrían escribirse con sólo 4, los más bajos requieren apenas 3 bits. La idea es que si los valores pequeños son mucho más frecuentes que los más altos, el código conseguirá reducir la cantidad media de bits requeridos. Esto es precisamente lo que sucede en el caso de una distribución laplaciana.

Nótese que, si se conoce el divisor, una cadena de 0's y 1's que representa una sucesión de números puede decodificarse unívocamente. Dada, por ejemplo, la cadena

00100001010100101101010000111

podemos reescribirla, para mayor claridad, como

00100 | 00101 | 0100 | 101 | 101 | 0100 | 00111

Para ello tuvimos en cuenta que el primer número comienza con una cantidad de 0's seguida por un 1 (símbolo separador), y luego dos bits arbitrarios que representan el resto de la división. En este caso hay dos 0's y un 1, seguido por 00. Esto puede repetirse mientras el código sea coherente, es decir, provenga de un codificador de Golomb-Rice. De acuerdo al significado del código, esto implica que los números son

$$2\times4 + 0,\ 2\times4 + 1,\ 1\times4 + 0,\ 0\times4 + 1,\ 0\times4 + 1,\ 1\times4 + 0,\ 2\times4 + 3,$$

es decir

8, 9, 4, 1, 1, 4, 11

Tabla 3.7. Ejemplo de codificación de Golomb-Rice para números entre 0 y 15 con divisor $D = 4$. El espacio de separación entre el cociente en representación unaria y el resto en representación binaria es al solo efecto de facilitar el análisis visual de la tabla.

N	Binario	Golomb-Rice
0	0000	1 00
1	0001	1 01
2	0010	1 10
3	0011	1 11
4	0100	01 00
5	0101	01 01
6	0110	01 10
7	0111	01 11
8	1000	001 00
9	1001	001 01
10	1010	001 10
11	1011	001 11
12	1100	0001 00
13	1101	0001 01
14	1110	0001 10
15	1111	0001 11

La elección del divisor D depende básicamente del parámetro b de la distribución de Laplace. Cuanto más pequeño sea b, mayor será la proporción de valores muy pequeños, por lo tanto convendrá un divisor pequeño. Dado que D se elige como una potencia de 2, es decir,

$$D = 2^d, \tag{3}$$

basta especificar d, denominado *parámetro de Rice*. FLAC utiliza la siguiente estimación para el valor de d óptimo (Salomon, 2007; Robinson, 1994):

$$d = \log_2\left(\ln 2 \ \mathrm{E}\left(\left|e(k)\right|\right)\right), \tag{4}$$

donde $\mathrm{E}(|e(k)|)$ es la esperanza de los errores absolutos de predicción. Para una distribución laplaciana de los errores se cumple

$$\mathrm{E}\left(\left|e(k)\right|\right) = b = \frac{\sigma}{\sqrt{2}}. \tag{5}$$

Si, por ejemplo, el desvío estándar de los errores fuera 8, resulta $d \cong 2$, de donde $D = 2^2 = 4$, que corresponde al ejemplo anterior.

Modelización predictiva

Con respecto la estrategia de modelización, FLAC utiliza uno de cuatro modelos para predecir la evolución de la señal: 1) la propia señal; 2) un valor constante; 3) una aproximación polinomial; y 4) una aproximación por código de predicción lineal (linear predictive coding, LPC). El modelo elegido es el que comprime más cada bloque.

Modelización mediante la propia señal (*verbatim*)

Este caso es apto para señales completamente aleatorias, por ejemplo, ruido blanco. En este caso, donde la señal tiene máxima entropía,[3] carece de sentido comprimir ya que al no haber redundancia (es decir, cualquier modelo da errores muy grandes) la compresión será mínima y probablemente quede desvirtuada por la necesidad de incluir los parámetros del modelo. A esto se agrega la carga computacional de codificar y decodificar sin que redunde en una compresión apreciable.

Modelización constante

Este caso se aplicaría principalmente a intervalos de silencio, que pueden comprimirse simplemente contando las muestras iguales a 0. Este tipo de compresión se conoce como compresión *run-length* y es la que se utiliza en los compresores tipo .zip para archivos generales o .gif para archivos de imagen.

Modelización polinomial

Este modelo consiste en obtener un polinomio de grado n que pasa por las $n + 1$ muestras anteriores ($n = 0, ..., 3$), extrapolando al instante actual para aproximar el correspondiente valor de la señal (figura 53). Dado que las muestras están separadas entre sí por tiempos uniformes, se puede demostrar que los polinomios adoptan una forma tal que lleva a una expresión muy sencilla del valor de predicción y del error. Llamando t al tiempo discreto, se trata de predecir la muestra $y(k)$ en función de las $n + 1$ muestras anteriores, es decir, $y(k - 1), ..., y(k - n - 1)$ mediante un polinomio de grado n.

Por ejemplo, para $n = 2$, buscamos los coeficientes de un polinomio cuadrático

$$P(t) = a_0 + a_1(t - k) + a_2(t - k)^2 \tag{6}$$

tal que

$$\begin{aligned} P(k-1) &= y(k-1) \\ P(k-2) &= y(k-2) \\ P(k-3) &= y(k-3) \end{aligned} \tag{7}$$

Sustituyendo (7) en (6) resulta el sistema de ecuaciones

$$\begin{aligned} a_0 - a_1 + a_2 &= y(k-1) \\ a_0 - 2a_1 + 4a_2 &= y(k-2), \\ a_0 - 3a_1 + 9a_2 &= y(k-3) \end{aligned} \tag{8}$$

cuya solución es

[3] La entropía de una señal es un indicador de su impredecibilidad y, por lo tanto, de su falta de redundancia. A mayor entropía, menor redundancia, por lo que la compresión será menos efectiva. Para señales discretas se define como $H = -\Sigma\, p_k \log_2(p_k)$, donde p_k es la probabilidad del k-ésimo valor discreto de la señal. Se interpreta como la mínima cantidad media de bits por muestra alcanzable por un código. Los códigos reales en general requieren una mayor tasa de información.

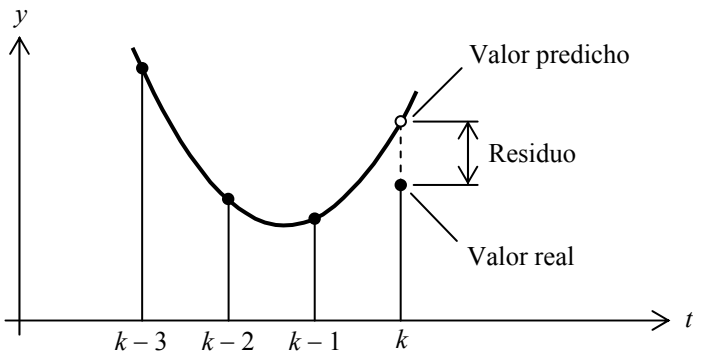

Figura 53. Modelo polinomial de segundo grado.

$$\begin{aligned} a_0 &= 3\,y(k-1) - 3\,y(k-2) + y(k-3) \\ a_1 &= \tfrac{5}{2}\,y(k-1) - 4\,y(k-2) + \tfrac{3}{2}\,y(k-3) \\ a_2 &= \tfrac{1}{2}\,y(k-1) - y(k-2) + \tfrac{3}{2}\,y(k-3) \end{aligned} \tag{9}$$

El valor predicho en $t = k$ es

$$\hat{y}(k) = P(k) = a_0 = 3\,y(k-1) - 3\,y(k-2) + y(k-3) \tag{10}$$

Esta aproximación se denomina de orden 3 ya que usa 3 muestras anteriores. En forma similar se obtienen las aproximaciones polinomiales de los diversos grados, según se indica en la tabla 3.8. Nótese que si bien las aproximaciones son polinomiales, el resultado es, salvo en el caso de orden 0, una *combinación lineal* de las muestras anteriores.

Dado que el concepto de "muestras anteriores" depende del instante específico en el que se quiere predecir una muestra, el modelo se actualiza muestra a muestra. Sólo sirve, por lo tanto, para predecir una muestra al solo efecto de obtener el error de predicción o residuo.

Es natural preguntarse qué grado conviene elegir en cada caso. La respuesta es que el codificador prueba con todos y elige el que produce el menor residuo absoluto acumulado. Aunque puede parecer que es un método de fuerza bruta con alto costo computacional, un hecho interesante es que los residuos para el grado n se pueden obtener a partir de los del grado $n - 1$. Llamando $e_n(k)$ al residuo en la predicción de la muestra $y(k)$ con un polinomio de grado n, es decir

$$e_n(k) = y(k) - \hat{y}_n(k), \tag{11}$$

entonces resulta

$$\begin{aligned} e_0(k) &= y(k) - y(k-1) \\ e_1(k) &= e_0(k) - e_0(k-1) \\ e_2(k) &= e_1(k) - e_1(k-1) \\ e_3(k) &= e_2(k) - e_2(k-1) \end{aligned} \tag{12}$$

Tabla 3.8. Expresiones de las aproximaciones polinomiales hasta el grado 3.

Orden	Grado	$\hat{y}(k)$
0	-	0
1	0	$y(k-1)$
2	1	$2\,y(k-1) - y(k-2)$
3	2	$3\,y(k-1) - 3\,y(k-2) + y(k-3)$
4	3	$4\,y(k-1) - 6\,y(k-2) + 4\,y(k-3) - y(k-4)$

Este proceso iterativo permite, por consiguiente, calcular los residuos en forma computacionalmente eficiente.

Una vez determinado el grado *n* del polinomio que proporciona el mínimo residuo acumulado, el único parámetro que requiere el decodificador es el grado del polinomio.

Modelización LPC

La modelización por predicción lineal es una variante más general de la modelización polinomial. Consiste en expresar el valor predicho como combinación lineal de *p* muestras anteriores:

$$\hat{y}(k) = \sum_{h=1}^{p} a_h \; y(k-h). \tag{13}$$

La diferencia con el modelo polinomial está en que los coeficientes a_h no son fijos sino que se determinan a partir de la señal completa dentro del bloque, eligiéndoselos de de modo de minimizar el error cuadrático de predicción

$$e^2 = \sum_{k=1}^{N} \left(y(k) - \sum_{h=1}^{p} a_h \; y(k-h) \right)^2, \tag{14}$$

donde *N* es la cantidad de muestras del bloque. La determinación se realiza mediante el algoritmo de Levinson-Durbin (Proakis et al., 1998).

El usuario puede optar por un valor máximo de *p* entre 1 y 32. El codificador determina el valor óptimo de *p* a partir del rango dinámico y el tamaño del bloque (Salomon, 2007). Cuanto mayor sea *p* mayor es el tiempo de cómputo. Se observa que para $p \geq 10$ la mejora es despreciable y en general no se justifica ante el mayor tiempo de cómputo.

La minimización del error cuadrático, en el caso de una distribución laplaciana implica también la minimización del error absoluto medio (ecuación (5), por lo cual se aprovecha mejor el código de Golomb-Rice. El precio a pagar es que se requiere una mayor cantidad de información, ya que se deben proporcionar los *p* coeficientes del modelo. De todas maneras, para poder trabajar con números enteros (la única manera de no depender de la implementación aritmética de cada procesador) los coeficientes LPC se cuantizan, por lo que el óptimo no es, en general, alcanzado.

Decorrelación intercanal

Hasta aquí analizamos la técnica de compresión para un único canal de audio. En el caso de dos canales estéreo, en general existe mucha redundancia entre las señales del canal izquierdo, y_I, y derecho, y_D, que puede aprovecharse para reducir aún más la tasa de información requerida. Ello se puede lograr reemplazando la señal mediante la transformación izquierdo-derecho a central-lateral, es decir, la semisuma y la diferencia:

$$\begin{aligned} y_C &= \frac{y_I + y_D}{2} \\ y_L &= y_I - y_D \end{aligned} \tag{15}$$

Si las señales son parecidas, y_L resulta más pequeña, lo cual permite una mayor compresión. Sin embargo, ello no siempre sucede, por lo cual FLAC analiza las dos versiones, (y_I, y_D), (y_C, y_L), codificando finalmente la que logra mejor compresión. Los canales izquierdo y derecho pueden recuperarse mediante las expresiones

$$\begin{aligned} y_I &= \frac{2y_C + y_L}{2} \\ y_D &= \frac{2y_C - y_L}{2} \end{aligned} \tag{16}$$

Dado que FLAC trabaja con enteros, si $y_I + y_D$ es impar la división por 2 de las ecuaciones (15) da fraccionaria y debe truncarse, lo cual llevaría a una recuperación imperfecta. Para evitarla, se puede restar 1 a y_I durante la conversión a (y_C, y_L) y luego, si y_L resulta impar, compensar sumando 1.