

Práctica 4. Codificación predictiva de imágenes

Resumen

En esta práctica implementaremos un codificador sin pérdidas para imágenes basado en el uso de predicción y un código Rice. Comprobaremos que este tipo de codificación puede ser más eficiente que la codificación Huffman.

1. Codificación predictiva sin pérdidas

La figura 1 muestra el diagrama de bloques de un codificador predictivo sin pérdidas para una secuencia unidimensional $x[n]$ ($n = 0, \dots, N - 1$). En este esquema, se codifica sin pérdidas la secuencia *error de predicción*, $e[n]$, que es el resultado de restar a $x[n]$ su *predicción* $\tilde{x}[n]$. El sistema *predictor* genera la predicción de cada muestra de $x[n]$ a partir de muestras anteriores de dicha secuencia. Observa que el codificador transforma $x[n]$ en $e[n]$ mientras que el decodificador realiza la operación inversa (transforma $e[n]$ en $x[n]$).

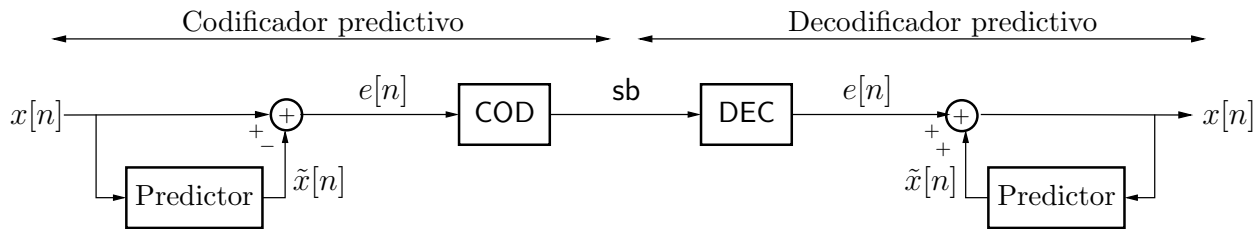


Figura 1. Codificación predictiva sin pérdidas.

El esquema de la figura 2 muestra cómo codificar predictivamente una imagen monocroma $x[n_1, n_2]$. En este esquema, primero se transforma $x[n_1, n_2]$ en una secuencia 1D, $x[n]$, usando exploración raster. Posteriormente se codifica $x[n]$ usando el codificador predictivo de la figura 1. En el otro extremo, primero se decodifica la secuencia de bits generando $x[n]$, y posteriormente se transforma $x[n]$ en la imagen $x[n_1, n_2]$.

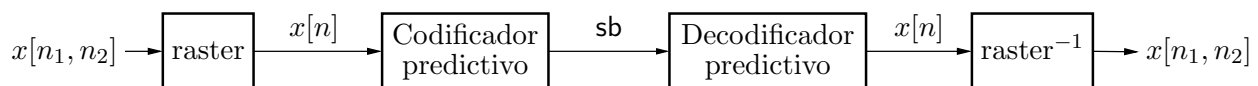


Figura 2. Codificación predictiva sin pérdidas de una secuencia 2D.

El predictor del codificador es el *predictor de primera diferencia*. En este predictor, la predicción de la secuencia en un instante es el valor de la secuencia en el instante anterior:

$$\tilde{x}[n] = x[n-1], \quad n = 1, \dots, N-1. \quad (1)$$

Para predecir la primera muestra $x[0]$, supondremos que $x[-1] = 0$, esto es, $\tilde{x}[0] = 0$.

Veamos con un ejemplo que la codificación predictiva puede ser más eficiente que la codificación directa de $x[n]$. La figura 3(a) muestra el histograma de la luminancia de la imagen `i1.png` ($x \in \{0, 1, \dots, 255\}$). La entropía de dicha imagen es 7.45 bits y, por tanto, la tasa binaria generada al codificar $x[n]$ con un código óptimo sería como mínimo de 7.45 bpp. La figura 3(b) muestra el histograma de la secuencia $e[n]$ que generaría el codificador predictivo (con el predictor de primera diferencia) al codificar la misma imagen. Observe que aunque $e[n]$ puede tomar valores enteros entre -255 y 255, solo unos pocos valores en torno a cero tienen una probabilidad apreciable. De hecho, la entropía de $e[n]$ es 4.6 bits (muy inferior a la de $x[n]$). Por tanto, la tasa binaria de un codificador predictivo puede ser significativamente inferior a la de un codificador que simplemente aplica un código a $x[n]$ (aunque dicho código sea óptimo para $x[n]$).

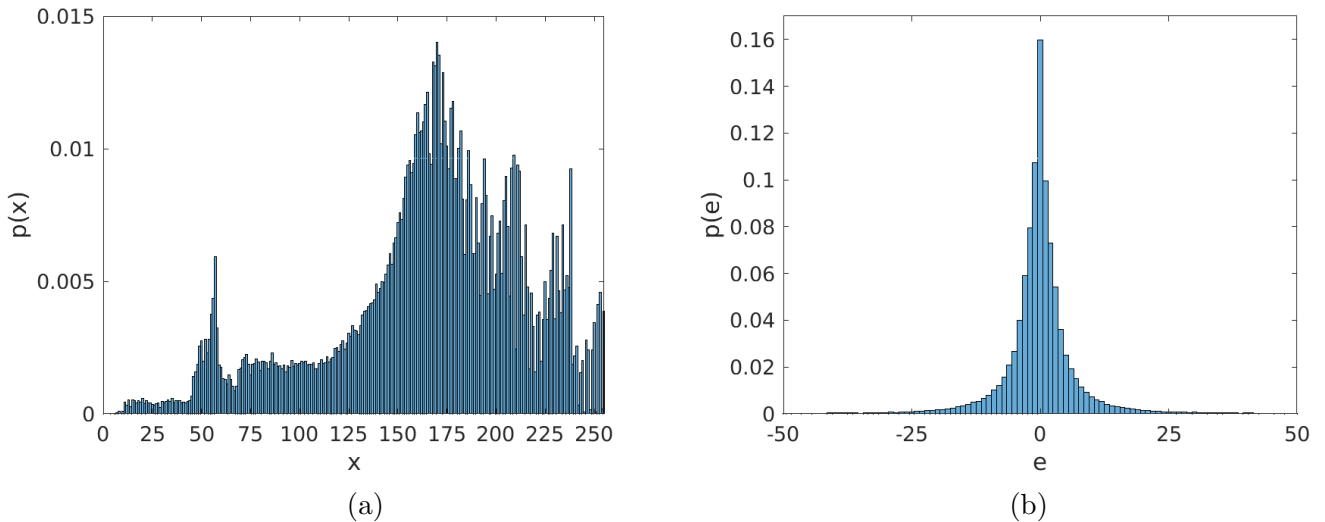


Figura 3. Histogramas de: (a) la luminancia de `i1.png`; (b) su error de predicción cuando el predictor utilizado es el de primera diferencia.

Observa en la figura 3(b) que $p(e)$ decrece conforme $|e|$ aumenta. Por tanto, un código Rice- m puede codificar $e[n]$ eficientemente si el parámetro m se elige de forma adecuada. Dado que e puede ser negativo, utilizaremos el código Rice- m para enteros.

2. Codificación Rice- m del error de predicción

La codificación del entero e con un código Rice- m requiere dar los siguientes pasos:

1. Transforma e en un entero positivo e' mediante:

$$e' = \begin{cases} 2e, & \text{si } e \geq 0 \\ 2|e| - 1 & \text{si } e < 0 \end{cases}$$

2. Transforma e' en los enteros e_q y e_r mediante

$$e_q = \left\lfloor \frac{e'}{2^m} \right\rfloor \quad (\text{cociente})$$

$$e_r = e' - e_q \cdot 2^m \quad (\text{resto})$$

donde e_q y e_r son el cociente y el resto de la división entera entre e' y 2^m .

3. Codifica e_q con un código unario y e_r con un código de longitud fija de m bits.
4. Obtén la palabra código, $c(e)$, concatenando las palabras obtenidas en el paso anterior.

Por ejemplo, para codificar $e = -15$ con un código Rice-3, realizamos los siguientes pasos:

1. $e' = 29$
2. $e_q = 3$ y $e_r = 5$
3. $c(e_q) = 0001$ y $c(e_r) = 101$
4. $c(e) = 0001101$.


La decodificación de una palabra código c generada con un código Rice- m requiere dar los siguientes pasos:

1. Obtén e_q contando el número de ceros que hay antes del primer uno de c
2. Obtén e_r decodificando los m bits que hay después del primer uno de c
3. Obtén e' mediante $e' = e_q \cdot 2^m + e_r$
4. Obtén e mediante

$$e = \begin{cases} e'/2, & \text{si } e' \text{ es par} \\ -(e' + 1)/2, & \text{si } e' \text{ es impar} \end{cases}.$$

Así, para decodificar la palabra código del ejemplo anterior (0001101):

1. obtenemos $e_q = 3$ (puesto que hay tres ceros antes del primer uno);
2. obtenemos $e_r = 5$ (puesto que los tres bits que hay después del primer uno son 101);
3. obtenemos $e' = 3 \cdot 8 + 5 = 29$;
4. como e' es impar, $e = -(29 + 1)/2 = -15$.

 Escribe un script de MATLAB que partiendo de un entero e y un entero positivo m codifique e con un código Rice- m y posteriormente decodifique la palabra código generada. Verifica que el entero decodificado coincide con e .

Para codificar y decodificar enteros con binario natural utiliza las funciones `int2bit` y `bit2int`, respectivamente. Para codificar un entero no negativo x con un código de longitud fija de m bits utiliza la función

$$c = \text{int2bit}(x, m)$$

donde c es un vector **columna** que contiene la palabra que resulta de la codificación (el primer bit de c es el bit más significativo). Similarmente para decodificar la palabra c (vector **columna** binario) de m bits ejecutaremos

$$x = \text{bit2int}(c, m)$$

siendo x el entero decodificado.

3. Implementación de un codificador predictivo con MATLAB

Vamos a implementar una función de MATLAB que codifique sin pérdidas la **luminancia** de imágenes de 8 bits. La codificación será predictiva y estará basada en el predictor de primera diferencia y un código Rice- m . La primera línea de la función debe ser:


```
function [nBits, R] = codPred(nombreI, nombreS, m)
```

donde:

- **nombreI** es el nombre del fichero con la imagen a codificar;
- **nombreS** es el nombre del fichero con la secuencia de bits que se genera;
- **m** es el parámetro del código Rice utilizado;
- **nBits** es el número total de bits que genera el codificador;
- **R** es la tasa binaria expresada en bits/píxel.

El fichero generado debe incluir una cabecera que contenga:

- el número de filas de la imagen (codificado con 16 bits);
- el número de columnas de la imagen (codificado con 16 bits);
- m (codificado con 3 bits).


 ¿Cuál es el mayor número de filas que puede tener la imagen a codificar? ¿Y el número de columnas? ¿Con qué valores de m se puede codificar la imagen?

La función `codPred` debe realizar principalmente los siguiente pasos:

1. Lee la imagen, obtén su luminancia y conviértela a *double* ($\rightarrow \mathbf{x}$)
2. Obtén el número de filas y de columnas de \mathbf{x}
3. Abre el fichero en modo escritura y escribe la cabecera
4. Transforma la matriz \mathbf{x} en un vector usando exploración raster
5. Para cada n :
 - Genera $e[n]$
 - Codifica $e[n]$. Si $n = 0$, usa un CLF de 8 bits; si $n > 0$, usa un código Rice- m
 - Escribe la palabra código obtenida en el fichero
6. Cierra el fichero.

Ten en cuenta que la ejecución de la función puede demorarse varios segundos (debido al gran número de iteraciones del bucle for y la escritura en fichero en cada iteración).

 Ejecuta la función `codPred` con la imagen `i1.png` y $m = 3$, anotando la tasa binaria obtenida.

 Repite la ejecución para $m = 1, 2, 4, 5$ anotando los valores de tasa binaria que obtienes en cada m . ¿Cuál es el valor de m óptimo para `i1.png`? ¿Cuál es la tasa binaria que se obtiene con el valor óptimo de m ?

 Repite lo anterior usando la imagen `i2.png`.

¿El valor óptimo de m encontrado coincide con el de la imagen de `i1.png`?

¿Cuál es la tasa binaria con dicho valor de m ?

Hemos observado que dependiendo de la imagen a codificar, el valor óptimo de m cambia. En nuestro codificador, m es una argumento de entrada por lo que es el usuario quien debe elegir el valor de m . Esto es problemático en la práctica porque el usuario no conoce el valor óptimo de m ; además, si el usuario eligen una valor de m alejado del óptimo, ¡es posible que la compresión aumente la tasa binaria en lugar de reducirla!

En la práctica el valor de m se determina de manera automática de forma que el codificador primero obtiene un valor de m adecuado para la imagen a codificar y posteriormente codifica la imagen con dicho valor.

 Describe de qué manera puede el codificador obtener de forma automática un valor de m adecuado para codificar una imagen.