Guion I

Razón de Compresión y Entropía

Información sobre la entrega de la práctica

Las prácticas se entregarán en un único fichero comprimido Practica01ApellidoNombre.zip. El fichero contendrá:

* Las funciones de Matlab a realizar en ficheros .m con los nombres de las funciones que se indiquen en el guion.
* Los trozos de código a realizar, que se entregarán todos en los pasos correspondientes de un único fichero .m llamado Practica01ApellidoNombre.m . Este fichero lo crearás modificando el fichero .m Practica01MolinaRafael.m en el servidor.
* Las discusiones y respuestas solicitadas en el guion se entregarán en un único fichero pdf. El nombre del fichero será Practica01ApellidoNombre.pdf. Lo construiráseditando Practica01MolinaRafael.doc y salvándolo en formato pdf.

El objetivo de esta práctica es afianzar los conceptos básicos de codificación y compresión de datos: factor de compresión y entropía y estudiar la importancia de la modelización de una fuente para obtener buenos resultados en la compresión de los datos generados por esa fuente. Necesitaremos ficheros de datos para comprimir. Hemos preparado un conjunto de ficheros de diferentes tipos de datos y características que se encuentran en el fichero comprimido *Ficheros de datos para prácticas* dentro de la sección (tema) *Material para las prácticas*.

Trabajaremos con el concepto de entropía de una fuente como medida de la información de la misma y veremos quees importante un correcto modelado de los datos para obtener los mejores resultados de un sistema de compresión.

Aplicaremos los conceptos de entropía y modelado de los datos a imágenes y ficheros de texto y binarios.

*Es importante distinguir entre el número de letras que tiene nuestro alfabeto y la representación de las mismas en el fichero. Suponemos aquí que cada letra se almacena inicialmente usando una codificación que asigna el mismo número de bits a cada una de las letras del alfabeto. Aunque tengamos el alfabeto {0, 1}, nosotros usaremos generalmente un byte para representar cada letra (codificada en binario, usando su código ASCII o cualquier otro sistema de representación) aunque está claro que un sistema más compacto sería usar sólo 1 bit por letra. Esta representación a un byte por letra ha sido escogida simplemente por comodidad puesto que realizar los programas usando bytes es más sencillo y cómodo que tener que usar representaciones a nivel de bits de los mismos.*

**Paso 1**

Limpiamos el espacio de trabajo. Leemos los datos del fichero ‘constitución española.txt’, calculamos el número de veces que aparece cada carácter y dibujamos el correspondiente histograma.

Entiende que hacen las funciones que utilizamos y sus parámetros. No incluyas ningún camino para el fichero, garantízate que has definido bien el path de Matlab. Observa el tipo de dato que leemos del fichero, éste será por defecto el tipo que usaremos.

clear all; close all;

fichero='constitucion española.txt';

fid=fopen(fichero, 'r')

[words count]=fread(fid,inf,'\*uint8');

fclose(fid)

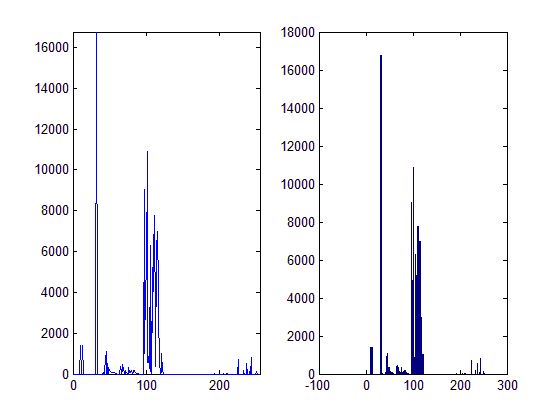
histograma= histc(words,[0:255]);

subplot(1,2,1);

plot([0:255],histograma); axis('tight')

% si prefieres puedes usar la función bar

subplot(1,2,2); bar([0:255],histograma)

**

**Paso 2**

Como ves el histograma está lejano de ser uniforme en el intervalo [0,255] por lo que es seguro que necesitaremos menos de 8 bits por dato. Vamos ahora a calcular la entropía de la fuente.

Crea una función que se llame **entropiaTUSINICIALES**, en mi caso sería **entropiaRMS** de mi nombre **R**afael **M**olina **S**oriano, que acepte un histograma, calcule su distribución de probabilidad asociada y devuelva la entropía. Nota: En Matlab lo podemos hacer de una forma muy sencilla. Primero convertimos el histograma en una distribución de probabilidad que podemos llamarprob. A continuación calculamos la entropía pero tenemos que hacerlo utilizando sólo aquellos términos con prob>0 (lee el manual de la función **find**). Matemáticamente esto no es necesario pero a Matlab no le gusta hacer 0\*log2(0). Luego escribe la fórmula de la entropía.

Es importante que escribas una buena implementación de entropía. Debes evitar los for y debe servir para cualquier tamaño de alfabeto, no solo aquellos que tienen 256 símbolos como máximo.

Incluye aquí el código de tu función entropiaTUSINICIALES.m

function [ entropiaAMC ] = entropiaAMC( histograma )

prob=histograma./sum(histograma)

entropiaAMC=-sum(prob(find(prob)).\*log2(prob(find(prob))))

end

**Paso 3**

Ejecuta la función que calcula la entropía, en mi caso entropiaRMS,

H= entropiaRMS(histograma)

Debe salirte 4.4880.

**Paso 4**

1. ¿Qué significa el valor de la entropía que has obtenido?.
2. ¿Cuál sería el factor de compresión que obtendríamos si usamos un modelo de codificación que alcanzase la entropía?.
3. ¿Podremos, a lo largo del curso, ganar a la entropía?

Escribe tus respuestas aquí

1. El número mínimo de bits para codificar los datos.

2. FC = Bits de la imagen original / Bits de la imagen comprimida. Si alcanzase la entropía el número de bits de la imagen comprimida serían los indicados por la entropía.

FC = 8 / 4.4880 = 1.7825

3. No, no es posible ganar a la entropía en términos de compresión sin pérdidas, porque como representa la cantidad mínima de bits necesarios para codificar los datos, superar a la entropía significaría usar menos bits que esos y no es posible con compresión sin pérdidas. Lo que sí es posible es acercarse a la entropía sin superarla, para optimizar la compresión.

**Paso 5**

Vamos a limpiar de nuevo el espacio de trabajo y las imágenes que hemos mostrado. A continuación leemos la imagen camera.pgm

clearall; closeall;

A=imread('camera.pgm');

% Mostramos la imagencamera.pgm

subplot(1,2,1); imshow(A);

**Paso 6**

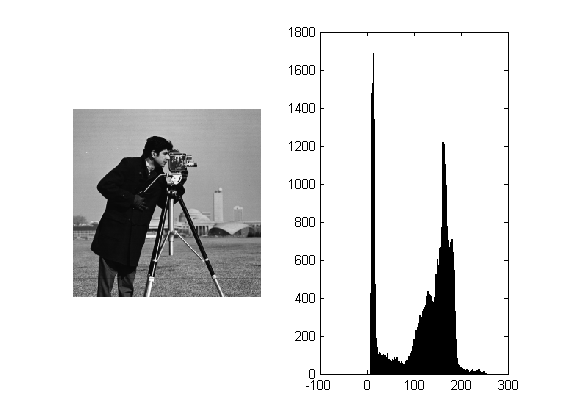
A continuación de la imagen A, no del fichero camera.pgm, calculamos el histograma, lo mostramos y calculamos la entropía de la fuente. Observa cómo introducimos la imagen bidimensional en la función histc

histograma=histc(A(:),[0:255]);

subplot(1,2,2); bar([0:255],histograma)

entropiaRMS(histograma)

Obtenemos gráficamente



**Paso 7**

1. ¿Cuál es el valor de la entropía que has obtenido?.
2. ¿Cuál sería el factor de compresión que obtendríamos si usamos un modelo de codificación que alcanzase la entropía?.

Escribe tus respuestas aquí

1. Entropía = 7.0097
2. FC = 8 / 7.0097 = 1.14127566

**Paso 8**

Limpia el espacio de trabajo y las variables. Creamos una imagen de tamaño 256x256 con niveles de gris 0 (fondo) y un cuadrado con nivel de gris 180. Utiliza el tipo uint8. Calcula ahora el histograma de esta imagen y su entropía.

clear all; close all;

A=uint8(zeros(256));

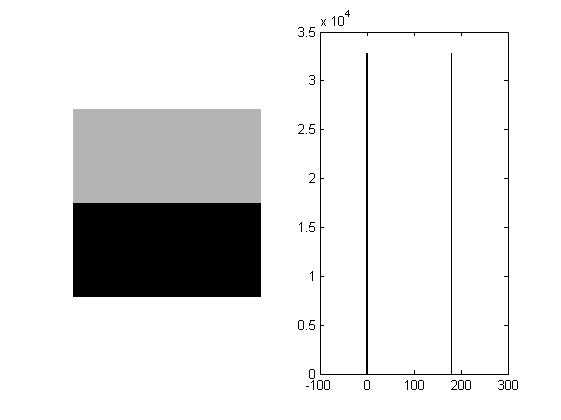
A(1:128,:)=uint8(180);

imshow(A)

histograma= histc(A(:),[0:255]);

figure; bar([0:255],histograma)

H=entropiaRMS(histograma)



**Paso 9**

1. ¿Qué significa el valor de la entropía que has obtenido?.
2. ¿Cuál sería el factor de compresión que obtendríamos si usamos un modelo de codificación que alcanzase la entropía?
3. ¿Podremos, a lo largo del curso, ganar a la entropía?

Escribe tus respuestas aquí

1. El valor de la entropía es 1, y esto significa que solo se necesita un bit (0 / 1) para codificar la imagen. Como la imagen solo tiene dos colores (gris y negro) y la probabilidad de ambos colores es la misma, solo es necesario un bit para representarlo.
2. FC = 8 / 1 = 8
3. En este caso alcanzar a la entropía es muy simple pero ganarla sería usar menos bits de los mínimamente necesarios para representarlo comprimido, por lo que teóricamente no es posible.

**Paso 10**

1. Si hicieras más grande (y luego más chico) el cuadrado blanco, ¿qué le pasaría a la entropía?
2. ¿Cuánto valdría la entropía si toda la imagen fuera blanca o negra?
3. ¿Qué significaría el valor de la entropía obtenido en este caso?.

Escribe tus respuestas aquí

1. La entropía disminuiría en ambos casos. Porque la probabilidad de uno de los dos colores sería mayor, por lo que la variabilidad en los datos se reduce y por ende disminuye la entropía.
2. En este caso la entropía valdría 0.
3. Como no habría variabilidad en los datos (por ser todo de un color), no existiría incertidumbre. Una entropía de 0 en una imagen completamente blanca o negra indica que la imagen es altamente predecible y no contiene información significativa o variabilidad.

**Paso 11**

La entropía de segundo orden de una fuente con un alfabeto de m letras se obtiene considerando la secuencia como formada por parejas de letras, (X1, X2). Supuesto que las parejas de letras (X1, X2) son independientes e idénticamente distribuidas, su entropía se calcula mediante:

H(S) = - Σ=0m-1 Σ j=0m-1 P(X1 = i, X2 = j) log P(X1 = i, X2 = j).

Por tanto, debemos calcular las probabilidades P(X1 = i, X2 = j). Para guardar estas probabilidades podemos construir una matriz de tamaño m x m en que cada celda (i,j) contiene el número de veces que la pareja (X1 = i, X2 = j) aparece en la secuencia dividido entre el número de parejas de letras en la secuencia.

Esta matriz de probabilidades también la podemos representar en forma de un vector, V, con m x m elementos en que cada posición del vector k = i\*m+jcontiene la probabilidad P(X1 = i, X2 = j) y calcular la entropía como

H(S) = - Σ k=0mxm-1 V(k) log V(k).

Observa que estoy empezando los índices en cero. Vamos a leer los caracteres de dos en dos y luego calcularemos la entropía. Lo haremos con el fichero ‘camera.pgm’. Observa la sintaxis de lectura y como calculamos el histograma

clear all; close all;

fichero='camera.pgm'

fid=fopen(fichero, 'r')

words=fread(fid,inf,'\*uint16');

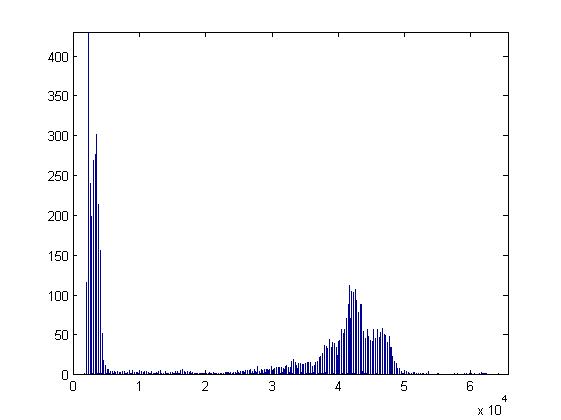
fclose(fid)

histograma= histc(words,[0:256\*256-1]);

bar([0:256\*256-1],histograma), axis('tight')

H=entropiaRMS(histograma)

El histograma es

****

**Paso 12**

¿Cuál es la entropía de esta fuente que codifica los símbolos de ‘camera.pgm’ de dos en dos?.

1. ¿Qué significa el valor de la entropía que has obtenido?.

Escribe tus respuestas aquí

1. El valor que he obtenido de la entropía es 11.1213. Podría pensarse que este valor es incorrecto porque se usarían más bits de los usados originalmente (8 bits) al comprimir, pero es que en este paso no se codifica un símbolo, sino dos. Por lo que los bits usados originalmente serían 16.

**Paso 13**

Leamos ahora el mismo fichero pero de byte en byte y calculamos la entropía

clear all; close all;

fichero='camera.pgm'

fid=fopen(fichero, 'r')

words=fread(fid,inf,'\*uint8');

fclose(fid)

histograma= histc(words,[0:255]);

H=entropiaRMS(histograma)

**Paso 14**

1. Compara los valores de la entropía que has obtenido en los pasos 11 y 13. ¿Qué está pasando?

Escribe tus respuestas aquí

1. La entropía en el paso 13 es 7.0102 porque en este paso sí que se usan 8 bits.

El FC del paso 11 es 1.43868 y el FC del paso 13 es 1.141194 por lo que de esto se deduce que es mejor leer de byte en byte.

**Paso 15**

Una vez que conocemos los programas que vamos a usar para calcular la entropía, vamos a aplicarlos a diferentes tipos de datos y ver su significado.

Para los ficheros bird.pgm, ptt1.pbm, texto10000.txt, Cinco semanas en globo - Julio Verne.txtcompleta la siguiente tabla. ¿Qué significan los valores que obtienes? Escribe el código correspondiente en el Paso 15 del fichero Practica01ApellidoNombre.

Escribe tus respuestas aquí

|  |  |  |
| --- | --- | --- |
| Fichero | Entropía de primer orden | Entropía de segundo orden |
| Bird.pgm | 6.7752 | 10.2581 |
| ptt1.pbm | 0.6979 | 1.1228 |
| texto10000.txt | 0.9999 | 1.9998 |
| Cinco semanas en globo - Julio Verne.txt | 4.4747 | 7.8791 |

Significado:

Lo primero que se puede observar es que la entropía de segundo orden es mayor a la de primer orden en todos los casos, y esto tiene sentido porque también se usan más bits para representarlas sin comprimir.

* Bird.pgm
  + La imagen tiene una cantidad moderada de información en la distribución de intensidades de gris de sus píxeles. Como la entropía de segundo orden (que tiene en cuenta las relaciones entre píxeles vecinos) es más alta que la entropía de primer orden indica que existen patrones o correlaciones entre los píxeles.
* Ptt1.pbm
  + Contiene poca información en cuanto a variabilidad de los datos. Como la entropía de segundo orden es ligeramente mayor que la de primer orden, sugiere cierta relación entre los píxeles vecinos.
* Texto10000.txt
  + Como los valores de la entropía son tan cercanos a 1 se deduce que es casi completamente predecible. Y como la entropía de segundo orden es el doble que la de primer orden indica que existen correlaciones entre caracteres adyacentes en el texto.
* Cinco semanas en globo – Julio Verne.txt
  + Contiene una cantidad moderada de información y variabilidad y que existen correlaciones entre caracteres adyacentes en el texto.

**Paso 16**

Lee la siguiente imagen

A=imread(‘bird.pgm’);

**Paso 17**

En el paso 17 del fichero Practica01ApellidoNombre.m escribe código para

1. Calcular la entropía de la matriz que contiene la imagen.
2. Calcular la diferencia de cada píxel con el anterior por filas. Es decir, vamos a calcular A(i,j)-A(i,j-1). No debes usar bucles y además tienes que tener mucho cuidado con las diferencias ya que la diferencia de dos caracteres sin signo da un carácter sin signo y esto no es lo que queremos hacer. Para calcular la diferencia de la primera columna considera que la columna anterior es cero.
3. Calcula también las diferencias módulo 256, es decir, (diferencias+256) módulo 256
4. Dibujar en una misma ventana la imagen de diferencias y su histograma.
5. Dibujar también la imagen de (diferencias +256) módulo 256 y su histograma
6. Calcular la entropía de primer orden sobre la imagen de diferencias y sobre la imagen (diferencias +256) módulo 256

Nota 1: Ten cuidado con los tipos de datos cuando hagas diferencias

Nota 2: ¡Cuidado al calcular las diferencias! Para una imagen en escala de grises, las diferencias pueden estar en el intervalo [-255, 255] y por tanto necesitamos al menos 9 bits para representarlas. Esto no es necesario en la imagen (diferencias +256) módulo 256.

Incluye aquí el código del paso 17 que has incluido en fichero Practica01ApellidoNombre.m

%1.

histograma=histc(A(:),[0:255]);

entropiaAMC(histograma)

%2.

Dif=zeros(size(A));

Dif(1:end,1:1)=int16(A(1:end,1:1));

Dif(1:end,2:end)=int16(A(1:end,2:end))-int16(A(1:end,1:end-1));

figure

subplot(1,2,1); imshow(Dif, [-255,255])

histogramaDif=histc(Dif(:),(-255:255));

subplot(1,2,2); bar([-255:255],histogramaDif);

entDif=entropiaAMC(histogramaDif)

%3.

Modulo=zeros(size(A));

Modulo=mod((Dif+256),256);

figure

subplot(1,2,1); imshow(Modulo, [0,255])

histogramaMod=histc(Modulo(:),(0:255));

subplot(1,2,2); bar([0:255],histogramaMod);

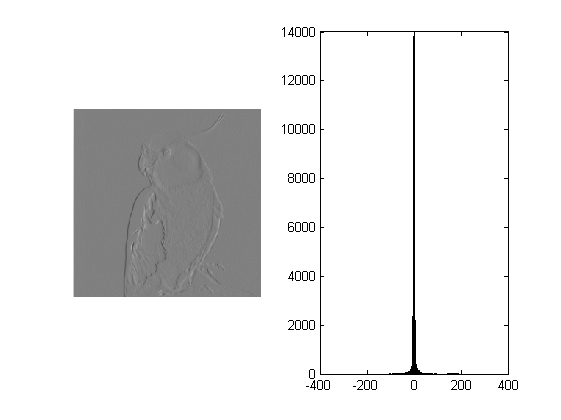
entMod=entropiaAMC(histogramaMod)

**Paso 18**

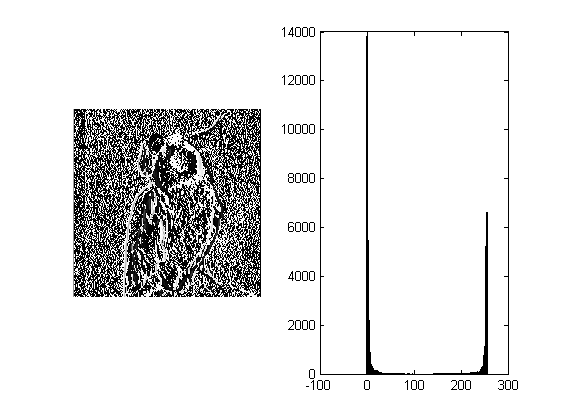
1. Incluye los gráficos del paso 17aquí
2. ¿Cuál son las entropías de la imagen original, de la imagen de diferencias y de la imagen (diferencias+256) módulo 256 ?
3. Compáralas y explica el resultado
4. Si hubiese codificado las diferencias usando (diferencias+256) módulo 256, ¿podrías con esta codificación de las diferencias reconstruir la señal original?

Escribe tus respuestas aquí

1. Diferencias



Módulo



1. Entropía imagen original = 6.7744

Entropía diferencias = 4.2333

Entropía diferencia de módulos = 4.2308

1. Como la entropía de la imagen original es la más alta significa que tiene la mayor cantidad de información y variabilidad. Esto es esperado porque al ser la imagen original contiene detalles y variaciones en intensidad de gris.

La entropía de las diferencias es más baja que la de la imagen original por lo que las diferencias entre píxeles adyacentes reducen la variabilidad en los datos, aunque aún hay información en estas diferencias.

La entropía del módulo de las diferencias no cambia apenas de la entropía de las diferencias, porque al aplicar el módulo 256 no cambia la cantidad de información contenida en las diferencias.

4. Sí, se podría reconstruir la señal original. El módulo 256 es una operación reversible, lo que significa que puedes deshacerla para obtener las diferencias originales. Por tanto, esta codificación de las diferencias preserva la información suficiente para reconstruir la señal original sin pérdida de datos.

**Paso 19**

Supongamos que tenemos una fuente que obtiene palabras de cuatro letras. Supongamos además que las letras son generadas aleatoriamente suponiendo una distribución uniforme sobre las 27 letras del abecedario. ¿Cuántos bits necesitaríamos en media para representar cada palabra de cuatro letras?

Escribe tus respuestas aquí

Si se tienen 27 letras se necesitan 5 bits (como mínimo) para representar cada letra (25 = 32). Las palabras son de 4 letras (4 \* 5 = 20 bits). Se usarán 20 bits para codificar una palabra.

La probabilidad de aparición de cada letra es 1/27 (la misma para cada letra).

Como la probabilidad de ocurrencia de cada letra es 1/27 (para todas las letras la misma probabilidad) y se busca la cantidad promedio de bits necesarios para representar cada palabra de cuatro letras, entonces se usa la fórmula de la entropía de Shannon:

Como hay 274 palabras posibles de cuatro letras, se puede calcular la entropía como:

Por lo tanto se necesitan aproximadamente 19.0196 bits de media para representar cada palabra de cuatro letras.