MATLAB y la teoría de la información

Tomás Robles Valladares

Ramón Alcarria Garrido

Borja Bordel Sánchez

Diego Martín de Andrés

MATLAB aplicado a la ingeniería telemática

Departamento de Ingeniería de Sistemas Telemáticos (UPM)

PROGRAMA

- Introducción
- Cálculo de entropía e información mutua en MATLAB
- Codificación de fuente con MATLAB
- Cálculo de la capacidad de un canal con MATLAB
- Técnicas FEC: codificación de canal en MATLAB

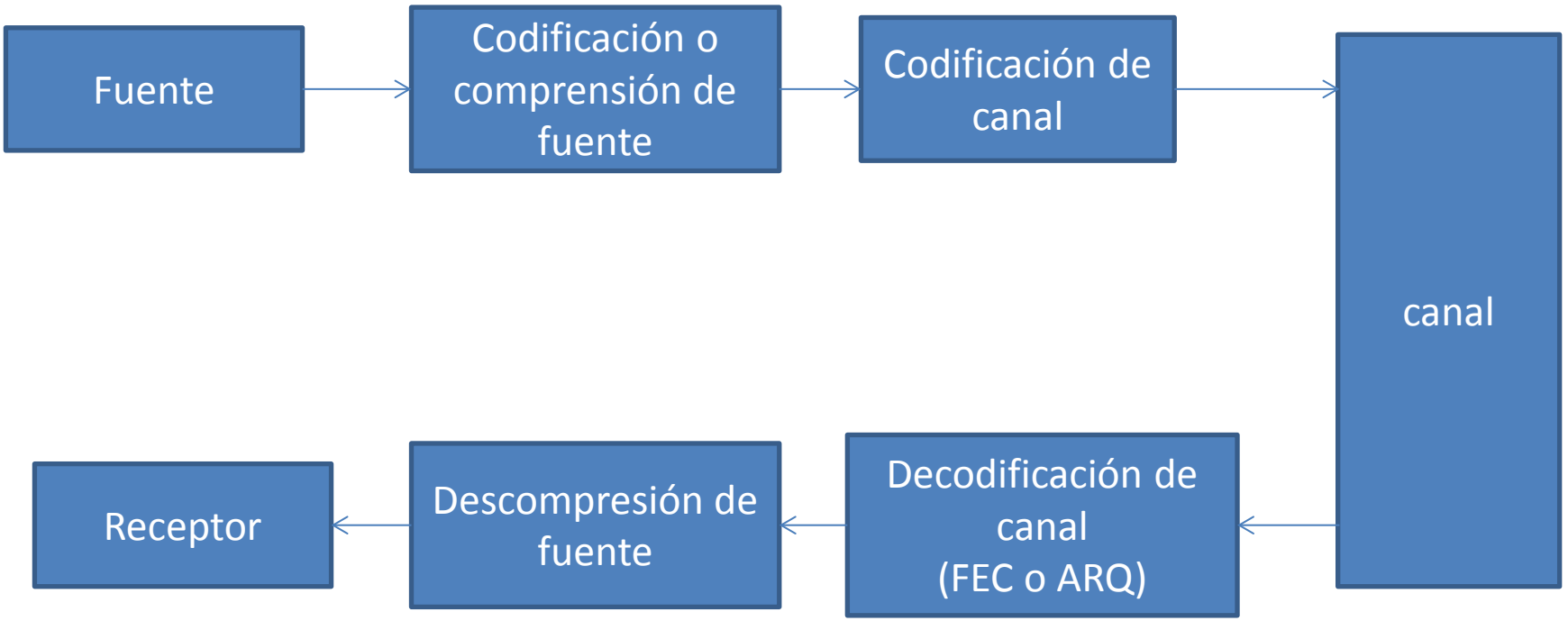
INTRODUCCIÓN

- La teoría de la información, también conocida como teoría matemática de la comunicación es una propuesta teórica presentada por Claude E. Shannon y Warren Weaver a finales de la década de 1940.
- Esta teoría agrupa las leyes matemáticas que rigen la transmisión de la información

INTRODUCCIÓN

- La teoría de la información se ocupa de la representación de la misma, así como de la capacidad de los sistemas de comunicación para transmitirla.
- Dada una fuente, que genera una serie de símbolos a una determinada tasa, el esquema básico de un sistema de comunicaciones en la teoría de la información es

INTRODUCCIÓN



INTRODUCCIÓN

- La codificación de fuente trata de reducir el tamaño medio de los mensajes, expresando los símbolos nativos de la fuente en función de un segundo conjunto de símbolos
  - Normalmente, en función de grupos de bits
- Se demuestra que existe un límite inferior para ese tamaño, relacionado con la entropía de la fuente generadora

INTRODUCCIÓN

- La codificación de canal trata de preparar los mensajes para su transmisión por un canal
- Se trata de incluir mecanismos que permitan detectar y/o corregir los errores que pudieran introducirse en el mensaje durante su transmisión por el canal
- Se habla de técnicas FEC si se pretende corregir errores en la decodificación, y de ARQ si se pretende detectar errores y pedir retransmisión de los mensajes defectuosos

INTRODUCCIÓN

- En la teoría de la información, los canales se modelan a través del conjunto de símbolos de entrada, el conjunto de salida y las funciones de probabilidad que los relacionan
- Existe un límite a la cantidad de información que puede transmitirse por un canal de forma fiable
  - La capacidad de canal
- Su cálculo es el objetivo primordial en el estudio de canales

CÁLCULO DE ENTROPÍA E INFORMACIÓN MUTUA EN MATLAB

- En el ámbito de la teoría de la información la entropía mide la incertidumbre de una fuente de información
- Si conocemos la probabilidad de ocurrencia de todos los símbolos de la fuente, su entropía puede obtenerse de forma teórica

$$H(X) = -\sum p(x) \cdot log_b[p(x)]$$

CÁLCULO DE ENTROPÍA E INFORMACIÓN MUTUA EN MATLAB

- La unidad en que se mide la entropía depende de la base del logaritmo escogida
  - $b = 2$ son BITS
  - $b = e$ son NATS
  - etc.
- El problema es que en casi ningún caso práctico se conoce a priori la probabilidad de ocurrencia de los símbolos de la fuente

CÁLCULO DE ENTROPÍA E INFORMACIÓN MUTUA EN MATLAB

- En la práctica, debe obtenerse la entropía de una fuente a partir de los mensajes generados por ésta
- Interesan mensajes largos, donde aparezcan todos los símbolos, de tal manera que la probabilidad de ocurrencia en el mensaje aproxime bien la probabilidad global para la fuente

CÁLCULO DE ENTROPÍA E INFORMACIÓN MUTUA EN MATLAB

- MATLAB no dispone de ninguna función que calcule automáticamente la entropía de la fuente que generó un mensaje
- El problema radica en identificar cuales son los símbolos sobre los que se trabaja
- Por ello, el cálculo debe hacerse mediante la suma de dos procesos

CÁLCULO DE ENTROPÍA E INFORMACIÓN MUTUA EN MATLAB

- Primero, con la función *hist()* calculamos la probabilidad de cada símbolo
- La forma de hacerlo es sencilla

```
distribución = hist(mensaje, [simbolo1,…,simbolon])/length(mensaje);
```

- En todo momento, suponemos que **mensaje** ha sido transformado en un array numérico

CÁLCULO DE ENTROPÍA E INFORMACIÓN MUTUA EN MATLAB

- Después, con ayuda del *Toolbox* gratuito *Information Theory Toolbox* podemos calcular la entropía desde la distribución obtenida
  - http://www.mathworks.com/matlabcentral/fileexchange/35625-information-theory-toolbox

```
entropia = entropy(distribucion);
```

CÁLCULO DE ENTROPÍA E INFORMACIÓN MUTUA EN MATLAB

- En MATLAB, el *Wavelet Toolbox* incluye una función para calcular la entropía de una matriz o vector, según varias definiciones
  - *wentropy()*
- Aunque incluye una variante llamada "Shannon", en realidad hace referencia a la llamada "entropía de Shannon no-normalizada" que nosotros no consideramos

CÁLCULO DE ENTROPÍA E INFORMACIÓN MUTUA EN MATLAB

- Su uso, por si fuera de interés, es muy simple

```
entropia = wentropy(mensaje, 'shannon', 0);
```

- **mensaje** puede ser cualquier matriz o vector
- El último argumento puede tomar cualquier valor, ya que no afecta al resultado

CÁLCULO DE ENTROPÍA E INFORMACIÓN MUTUA EN MATLAB

- Sustituyendo el parámetro *shannon* por otro (por ejemplo *log energy*) puede obtenerse la entropía según se ha definido en otros campos
- Puede ser útil en procesado de imágenes como etapa previa a realizar alguna transformada
- El resultado sólo puede obtenerse en NANTS
  - Puede transformarse a bits haciendo uso de los cambios de base en logaritmos

CÁLCULO DE ENTROPÍA E INFORMACIÓN MUTUA EN MATLAB

- En teoría de la información, la información mutua de dos fuentes mide la reducción de la incertidumbre (entropía) en una de ellas al conocer el símbolo que se ha generado en la otra
- De nuevo, si conocemos la probabilidad de ocurrencia de todos los símbolos de ambas fuentes puede obtenerse de forma teórica

CÁLCULO DE ENTROPÍA E INFORMACIÓN MUTUA EN MATLAB

$$I(X;Y) = \sum_{X}\sum_{Y} p(x,y) \cdot log_b\left(\frac{p(x,y)}{p(x) \cdot p(y)}\right)$$

- Por ahora MathWorks no ha incluido de forma nativa ninguna función para calcular la información mutua a partir de los mensajes generados por las fuentes

CÁLCULO DE ENTROPÍA E INFORMACIÓN MUTUA EN MATLAB

- La comunidad de usuarios ha propuestos muchas soluciones. Algunas muy extendidas como
  - http://www.mathworks.com/matlabcentral/fileexchange/13289-fast-mutual-information-of-two-images-or-signals
- Sin embargo, la forma más “elegante” de realizar el cálculo es mediante el uso del *Information Theory Toolbox*

CÁLCULO DE ENTROPÍA E INFORMACIÓN MUTUA EN MATLAB

- Una vez obtenidas las distribuciones de probabilidad de cada fuente con la función *hist(),* basta hacer

```
Imutua = mutualInformation(distribucion1, distribucion2);
```

- Existe otro procedimiento que, dependiendo del caso, puede ser más preciso

CÁLCULO DE ENTROPÍA E INFORMACIÓN MUTUA EN MATLAB

- Sabemos que

$$I(X;Y) = H(X) + H(Y) - H(X,Y)$$

- Con lo que el problema reside ahora en calcular la entropía conjunta $H(X,Y)$
- Para ello calcularemos el histograma conjunto de los mensajes de ambas fuentes

CÁLCULO DE ENTROPÍA E INFORMACIÓN MUTUA EN MATLAB

- La forma es sencilla

```
distribucionConjunta = accumarray([mensaje1 mensaje2], 1)/length (mensaje1);
```

- Es preciso que ambos mensajes tengan la misma dimensión para que este proceso puede ejecutarse

CÁLCULO DE ENTROPÍA E INFORMACIÓN MUTUA EN MATLAB

- Después, con ayuda del *Toolbox* gratuito *Information Theory Toolbox* podemos calcular la entropía conjunta desde la distribución obtenida

```
entropiaConjunta = jointEntropy(distribucion);
```

CODIFICACIÓN DE FUENTE CON MATLAB

- En codificación de fuente trataremos de transformar el conjunto de símbolos con los que se representa un mensaje, para acortar su tamaño
- Se puede demostrar que, en media y para una fuente dada, no se pueden emplear menos BITS para codificar un símbolo que los dados por la entropía de la fuente

CODIFICACIÓN DE FUENTE CON MATLAB

- Si nuestro código emplea, en media, tantos BITS como indica la entropía diremos que es un código óptimo
  - Es el más corto posible
- En caso contrario hablaremos de códigos sub-óptimos
  - Son más o memos largos, pero mayores que uno óptimo

CODIFICACIÓN DE FUENTE CON MATLAB

- Existen dos códigos de fuente destacados
  - Shannon-Elias-Fano, código subóptimo
  - Huffman, código óptimo
- Desde su origen, MATLAB ha soportado codificación de Huffman
- Los codificadores de Shannon-Elias-Fano, aunque antiguos, pertenecen a librearías externas gratuitas

CODIFICACIÓN DE FUENTE CON MATLAB

- Para todos los casos, un paso previo muy importante es conocer la probabilidad de ocurrencia de los símbolos de la fuente
- Será preciso, por tanto, emplear la función *hist()* para poder calcular las distribuciones tal y como vimos anteriormente

CODIFICACIÓN DE FUENTE CON MATLAB

- La codificación de Shannon-Elias-Fano no viene soportada directamente en MATLAB
- Puede emplearse mediante funciones distribuidas por la comunidad de usuarios como
  - http://www.mathworks.com/matlabcentral/fileexchange/41727-shannon-fano-encoder

CODIFICACIÓN DE FUENTE CON MATLAB

- El uso de la función es relativamente sencillo

```
[diccionario, logitudCodigo] = sfencoderkasan(distribucion);
```

- La función recibe una distribución de probabilidad, y devuelve un diccionario, de tal forma que el símbolo en la primera posición sustituye a aquel cuya probabilidad ocupaba la primera posición en la distribución

CODIFICACIÓN DE FUENTE CON MATLAB

- Por su parte, la codificación de Huffman está ampliamente soportada y documentada en MATLAB
- En este caso, no sólo podremos obtener un diccionario, sino también codificar mensajes empleando el diccionario calculado

CODIFICACIÓN DE FUENTE CON MATLAB

- Para realizar codificación Huffman lo primero es obtener un diccionario

```
[diccionario, logitudCodigo] = huffmandict (simbolos, distribucion);
```

- En este caso, además de la distribución, la función precisa del listado de símbolos generados por la fuente

CODIFICACIÓN DE FUENTE CON MATLAB

- El orden de listado de los símbolos debe coincidir con el de la distribución
- Una vez obtenido el diccionario se puede codificar cualquier mensaje como

```
mensajeCodificado = huffmanenco (mensaje, diccionario);
```

- También puede luego descodificarse

```
mensaje = huffmandeco (mensajeCodificado, diccionario);
```

CÁLCULO DE LA CAPACIDAD DE UN CANAL CON MATLAB

- La capacidad de un canal indica la mayor cantidad de información que puede transmitirse de forma fiable por cada símbolo de canal
  - Fiable implica con probabilidad de error tan pequeña como se quiera
- En general esta medida viene dada en bits/símbolo

CÁLCULO DE LA CAPACIDAD DE UN CANAL CON MATLAB

- Matemáticamente

$$C = \max_{p(x)} I(X;Y)$$

- Es decir, se trata de localizar la fuente para la que la información mutua entre los mensajes a la entrada y a la salida del canal es máxima

CÁLCULO DE LA CAPACIDAD DE UN CANAL CON MATLAB

- Si en el caso de la entropía y la información mutua la definición matemática era directamente aplicable, en este caso es impracticable salvo en casos sencillos
  - Canal sin ruido
  - Canal binario simétrico
  - Etc.

CÁLCULO DE LA CAPACIDAD DE UN CANAL CON MATLAB

- Para el resto de casos, es preciso aplicar un algoritmo numérico que aproxime la definición
- Para el caso de la capacidad de canal el más común es el algoritmo de Blahut–Arimoto
- Se trata de un algoritmo iterativo, que pretende solucionar un problema de optimización, por lo que hay ocasiones en la que no alcanza la convergencia

CÁLCULO DE LA CAPACIDAD DE UN CANAL CON MATLAB

- Una vez más este algoritmo no está soportado de forma nativa en MATLAB
- El *Information Theory Toolbox* tampoco incluye esta funcionalidad
- Es preciso instalar una nueva librería externa
- Una de las más sencillas de usar se encuentra distribuida como código libre
  - https://gist.github.com/Piyush3dB/01df75af9889414de1b6

CÁLCULO DE LA CAPACIDAD DE UN CANAL CON MATLAB

- Basta invocar la función como

```
capacidad = BlahutArimoto(matrizTransmision);
```

- **matrizTransmision** se construye de tal manera que la posición (i,j) indica la probabilidad de que a la salida del canal aparezca el símbolo j-ésimo cuando a la entrada se sitúa el i-ésimo

CÁLCULO DE LA CAPACIDAD DE UN CANAL CON MATLAB

- La matriz de transmisión puede deducirse de forma experimental
- Puede construirse un algoritmo MATLAB que construya la matriz a partir de los mensaje de entrada y salida del canal
- Codificar dicha rutina es un problema algorítmico que tiene múltiples soluciones

TÉCNICAS FEC: CODIFICACIÓN DE CANAL EN MATLAB

- Las técnicas FEC, o de corrección de errores hacia adelante, son un tipo de mecanismos de corrección de errores que permite su corrección en el receptor sin retransmisión de la información original
- La posibilidad de corregir errores se consigue añadiendo al mensaje original unos bits de redundancia

TÉCNICAS FEC: CODIFICACIÓN DE CANAL EN MATLAB

- Se llaman códigos de canal a los esquemas que permiten calcular los bits de redundancia para un mensaje dado, y detectar y/o corregir los errores en el receptor si los hubiera
- En concreto, aquí nos centraremos en los códigos de bloque

TÉCNICAS FEC: CODIFICACIÓN DE CANAL EN MATLAB

- Los códigos de bloque transforman un mensaje binario de N bits en otro de K bits con K > N.
- El número de bits de redundancia será, por tanto, M=K-N
- En este curso hablaremos de códigos lineales
  - Y especialmente de los códigos cíclicos (subconjunto de los lineales)

TÉCNICAS FEC: CODIFICACIÓN DE CANAL EN MATLAB

- De forma genérica, se puede aplicar codificación de canal a un mensaje con la función

```
codificado = encode(mensaje, n, k, 'tipo/formato', generador);
```

- **tipo** puede tomar tres valores: *linear, cyclic* y *hamming* en función del código que queramos aplicar

TÉCNICAS FEC: CODIFICACIÓN DE CANAL EN MATLAB

- **formato** puede tomar dos valores: *binary* y *decimal* en función de si queremos aplicar el código a BITS o a números enteros
- **generador** es el elemento generador del código
  - Una matriz en el caso lineal, polinomio para los cíclicos, etc.
- **n** y **k** son los tamaños del mensaje antes y después de la codificación respectivamente

TÉCNICAS FEC: CODIFICACIÓN DE CANAL EN MATLAB

- La decodificación puede hacerse de igual manera

```
mensaje = decode(mensaje, n, k, ‘tipo/formato’, generador);
```

- Los parámetros son idénticos a los empleados para el caso del codificador

TÉCNICAS FEC: CODIFICACIÓN DE CANAL EN MATLAB

- Para el caso de los códigos cíclicos y tipo *hamming*, los elementos generadores pueden obtenerse con las funciones

```
generadorCiclico = cyclpoly(n, k);
```

```
generadorHamming = gfprimdf(n-k);
```

TÉCNICAS FEC: CODIFICACIÓN DE CANAL EN MATLAB

- Un caso particular de gran interés por su uso extensivo en la actualidad son los códigos Reed Solomon.
- Se trata de un subconjunto de códigos cíclicos, actualmente empleado, por ejemplo, en el estándar DVB

TÉCNICAS FEC: CODIFICACIÓN DE CANAL EN MATLAB

- Se puede codificar un mensaje con un código Reed Solomon haciendo

```
codificado = rsenc(mensaje,n,k);
```

- Si se desea explicitar un polinomio generador concreto también puede aplicarse

```
codificado = rsenc(memsaje,n,k, generador);
```

TÉCNICAS FEC: CODIFICACIÓN DE CANAL EN MATLAB

- Existe, también, una función para calcular polinomios generadores de códigos Reed Solomon

```
generadorReed = rsgenpoly(n,k);
```

TÉCNICAS FEC: CODIFICACIÓN DE CANAL EN MATLAB

- La decodificación se realiza de la misma forma

```
mensaje = rsdec (codificado,n,k);
```

```
mensaje = rsdec (codificado,n,k, generador);
```