Transmisión de Datos
Codificación de Fuente

José M. Martínez
Grupo de Tratamiento de Imágenes
Escuela Politécnica Superior
Universidad Autónoma de Madrid, SPAIN

JoseM.Martinez@uam.es
tel:+34.91.497.22.58

2011-2012

Escuela Politécnica Superior
Universidad Autónoma de Madrid

Índice

Índice

- Introducción
- Fundamentos
- Algoritmos de codificación de fuente sin pérdidas
- Teoría Tasa-Distorsión
- Cuantificación
- Codificadores

Introducción

Información: nuevo conocimiento que se obtiene acerca de algo

- La información se obtiene oyendo, viendo, oliendo, tocando, …

Fuente de información: algo de interés para el receptor que no sabe la salida por adelantado

En Sistemas de (tele)comunicación la información se obtiene observando una fuente aleatoria transmitida por un canal, esto es, una variable aleatoria

Objetivo: diseñar sistema de comunicación para transmitir fiablemente (o con la mínima distorsión) un proceso aleatorio (la información)

- Reducir (sin/con pérdidas) la cantidad de información [***codificación de fuente***]
- Diseñar sistemas transmisión con la menor probabilidad de error de bit [teoría de la comunicación]
- Diseñar códigos/sistemas de protección (detección/corrección) frente a errores [codificación de canal]

Introducción

Código fuente: Mapeo del rango de una fuente a un conjunto finito de símbolos de un alfabeto D-ario

- Binario: {0,1}, {punto, raya}, …
- Ternario: {A, B, C} …
- Ejemplo: fuente:{Red, Blue}, alfabeto={0,1}, código={00,11}. No es óptimo pero es código

Códigos:

- Palabras código de longitud fija (asignación “ciega”)
- Palabras código de longitud variable (basados en estadísticos)
- Palabras código de longitud fija con longitud de entrada variable (no requieren conocer estadísticos)
- …

Compresión (límite H): Codificación sin pérdidas vs con pérdidas

Ejemplos de codificación de fuente (I)

Código Braille:

- 3x2 puntos llano/marcado (on/off) => binario
- 6 puntos binarios => 6 bits => 64 símbolos (2^6)
- Letras, dígitos y símbolos de puntuación son menos de 64
- El resto se utiliza para palabras, prefijos y sufijos comunes (como en taquigrafía)
- No comprime mucho, pero es fácilmente (y rápidamente) decodificable

Compresión de texto irreversible

- En general texto se basa en código ASCII (8bits/símbolo): caracteres imprimibles 32-255
  - Código Unicode (16bits/símbolo): caracteres imprimibles 0x0020-0xFFFC (32-65.532)
- Quitar caracteres de formato, acentos, puntuación, pasar todo a mayúsculas/minúsculas (26 + ñ = 27) => 5 bits (2^5=32) y quedan 5 para blanco, puntuación, palabras comunes, …
- Ciertas pérdidas, pero …

Ejemplos de codificación de fuente (II)

Compresión ad-hoc (reversible) de texto

- Quitar blancos y sustituir por cadena binaria de 100000100001…
  - 1: indica blanco, 0: carácter
- Quitar blancos de formato y sustituir por digito de número de blancos (***extensión de fuente-codificación por carreras…***)
  - Hace falta código de escape …
- Usar 7 bits para los caracteres textuales ASCII (compresión 7/8)
  - Más el bit anterior, lo que hace es reducir los blancos de 8 a 1.
- 40^3=64.000 < 2^{16}=65.536
  - 40 = 26 letras, 10 dígitos, 4 símbolos de puntuación (. , ! ?) –esto es para inglés ☹ : ni ñ, ni ¿, ni ¡, …-
  - Si agrupo caracteres de 3 en 3 (***extensión de fuente-codificación vectorial***) tengo una nueva fuente que tiene 40^3 símbolos y que se puede codificar con 16 bits (teniendo incluso adicionalmente símbolos adicionales): compresión 2/3

Este ejemplo da algunas pistas de trucos para codificar, pero lo suyo es acercarse “rigurosamente” al problema…

Índice

Modelado de Fuentes (I)

Las fuentes de información se pueden modelar como procesos aleatorios, dependiendo sus propiedades (e.g., entropía, ancho de banda, energía, …) de la naturaleza de la fuente

- Señales de voz: la densidad espectral de potencia se concentra mayoritariamente en la banda de 300 a 3400 Hz
- Señales de audio: su ancho de banda está limitado entre los 20 y 20.000 kHz (en audio CD se muestrea a 44,1 kHz –filtros no ideales-)
- Señales de video: su ancho de banda está también limitado, si bien depende del sistema empleado: 0-4,5 MHz (PAL), 0-6,5 MHz (NTSC)
- Señales de telemetría: su ancho de banda depende de la velocidad de cambio de la señal que se mide (temperatura, electrocardiograma, …)

Todas tienen en común que son procesos limitados en banda, por lo que si se muestrean por encima de la frecuencia de Nyquist se puede reconstruir perfectamente (si no se cuantificasen).

- Muestreo con $f_s \geq 2\ B_{max}$ (Nyquist): $x(t) \rightarrow x[n]$
- Cuantificación $x[n] \rightarrow x_q[n] \Rightarrow$ distorsión (error de cuantificación)
- El muestreo es reversible pero la cuantificación no

Modelado de fuentes (II)

La codificación de fuente se definirá (y aplicará) sobre vv.aa. discretas, pues las fuentes digitales son procesos aleatorios discretos y las analógicas hemos visto que se pueden modelar como tales (recordando siempre que el paso de cuantificación introduce ya unas pérdidas)

Modelo matemático de Fuente: $\{x_i\}_{i=-\infty}^{+\infty}$, con f.d.p p(x)

- pudiendo ser los valores de x_i discretos (fuente digital o analógica muestreada cuantificada) o continuos (fuente analógica muestreada sin cuantificar)
  - Si bien las propiedades estadísticas de la fuente no depende de sus valores, si pueden hacerlo del número "infinito" de posibles valores (en cualquier caso en todos los casos prácticos tendremos valores discretos al final)
- Solamente veremos modelos simples de fuente (DMS)

Fuente Discreta Sin Memoria (DMS)

El modelo de fuente más sencillo es la Fuente Discreta Sin Memoria –DMS- (*Discrete Memoryless Source*)

Proceso aleatorio discreto en tiempo y en amplitud, y donde todos los x_i se generan independientemente y con la misma f.d.p

DMS genera una secuencia de vv.aa. i.i.d con valores pertenecientes a un conjunto discreto

Se describen mediante

- Alfabeto $A=\{a_i\}_{i=1...N}$
- f.d.p. (discreta) $p(x)=\{p_i\}_{i=1...N}$

Ejemplo más sencillo: fuente binaria DMS

- $A=\{0,1\}$
- p(i)={p, 1-p}
- Si p=1/2 => Fuente Binaria Simétrica –BSS- (*Binary Simetric Source*)

Ejercicios de clase 4: DMS

Sea una señal de audio B=4.000 Hz que se muestrea a la frecuencia de Nyquist. Asumiendo que la secuencia de salida se puede modelar como una DMS:

- $A=\{-2,-1,0,1,2\}$
- p(i)={1/2,1/4,1/8, 1/16,1/16}

Hallar el régimen binario (mínimo) de la fuente R en [bits/seg] para codificación sin pérdidas.

Comentar los resultados.

Ejercicios de clase 4: DMS - solución

Índice

Teorema de Codificación de Fuente

Uno de los 3 teoremas fundamentales enunciados por Shannon en 1948

- Si $Rf_{\text{sin pérdidas}} \geq H \Rightarrow \exists$ código con $P_e \rightarrow 0$
- Shannon no daba algoritmos, aunque si pistas (si bien dan lugar a algoritmos "poco prácticos")

Sea una DMS y observamos secuencias de n símbolos

Si n $\rightarrow \infty$, por la ley de los grandes números y la PEA, habrá:

- Aproximadamente np_1 de símbolos a_1
- Aproximadamente np_2 de símbolos a_2
- Aproximadamente np_3 de símbolos a_3
- …

Cada secuencia de n símbolos tendrá con P→1:

- $np_1 \rightarrow a_1$, $np_2 \rightarrow a_2$, $np_3 \rightarrow a_3$, … (misma composición con distintos órdenes de aparición)

¿Cual es la probabilidad de que una secuencia sea típica (X=x)?

- $P(X=x) \approx \prod_{i=1..N} p_i^{npi} = \prod_{i=1..N} (2^{\log\ pi})^{npi} = 2^{n(\sum pi.\log\ pi)} = 2^{-nH(x)}$

#secuencias típicas$\approx 2^{nH(x)}$ versus #total de secuencias N^n siendo N el cardinal del alfabeto

$P(X \neq x) \rightarrow 0$ (más cierto cuanto mayor sean n)

Fundamento de la compresión de fuente

Por tanto:

- Se codifica óptimamente las secuencias típicas: 2^{nH} con nH bits/símbolo (símbolo==secuencia típica)
- Se codifica subóptimamente las otras secuencias (codificación sin pérdidas) o no se codifican (codificación con pérdidas)

Para que haya compresión tiene que cumplirse que:

- $2^{nH} \leq N^n$ lo que equivale a que $H \leq \log N$ (v.a. no uniforme)
  - N=#símbolos de la fuente
  - N^n=#palabras de la fuente
  - 2^{nH}=#secuencias típicas
- Si H=log N podría haber compresión si N <> 2^n

Si no DMS => fuentes con memoria (tasa de entropía): cuántas más salidas se conocen más predecible es la siguiente

- Por ejemplo, texto en inglés: tras 'q' suele ir 'u', 'l' y 'a' suelen ir entre espacios, … Calculando la tasa de entropía con muchos textos, se demuestra que con n=10 ésta ya converge (Datos para 26 letras más espacio solamente: H(x)[n=1]=4,03 bit/letra, mientras que H=1,3 bits/letra)

Enunciado del Teorema de Codificación de Fuente

Una fuente con entropía o tasa de entropía H se puede codificar con error nulo si la tasa de la fuente R ≥ H.

Si R < H el error no estará acotado independientemente de la complejidad del codec (**co**dificador-**dec**odificador)

[Shannon 1948]

El teorema da la condición necesaria y suficiente, pero no los algoritmos.

El teorema se enuncia con R (tasa) pero hay que recordar que H [bits/símbolo] y R [bits/segundo] se relacionan mediante $R=\bar{l}*f_s$.

Índice

Índice

- *Introducción*
- *Fundamentos*
- Algoritmos de codificación de fuente sin pérdidas
  - **Introducción**
  - Codificación Huffman
  - Modificación de fuente (extensión de fuente)
  - Codificación aritmética
  - Codificación Lempel-Ziv
- Teoría Tasa-Distorsión
- Cuantificación
- Codificadores

Algoritmos de codificación de fuente sin pérdidas

Según el teorema de codificación de fuente se pueden diseñar algoritmos de codificación (y decodificación) sin pérdidas con $\bar{l} \leq \log N$ siempre que $H \leq \log N$, siendo H el límite de $\bar{l}$.

Los mecanismos son variados:

- Codificación de longitud variable versus fija
  - Cadenas de símbolos de X (longitud fija) se mapean a cadenas de tamaño variable de símbolos del alfabeto D-ario (cuanto más frecuentes menos símbolos de D). Problemas de sincronización para saber cuando termina la cadena de tamaño variable
  - Codificación Shannon: $l_i=\text{ceil}(-\log p_i)$. Solamente funcionan bien bajo ciertas condiciones
    Analizar {a1,a2} con p1=0.0001 y p2=0.9999 [14, 1] !!!!!
  - Codificación Huffman (ver más adelante)
  - Codificación Shannon-Fano-Elias: se basa en $F(x)=\Sigma_{a \leq x} p(a)$ [Función de probabilidad acumulada] y logra códigos con $l_i=\text{ceil}(-\log p_i)+1$
  - Codificación aritmética: se basa en la anterior y codifica bloques de n símbolos en lugar de símbolos aislados. Más eficiente pero con cierta complejidad de implementación: tamaño de buffer, precisión de cálculo, … Patente de IBM (ver más adelante)
- Extensión de fuente (ver más adelante)
  - Previamente a la codificación se modifica (adapta) la fuente para aprovechar el codificador posterior (modificar estadísticos, orden, …)
- Códigos universales
  - No es necesario conocer la fdp a priori
  - Codificación aritmética adaptativa
  - Codificación Lempel-Ziv y variantes (ver más adelante)

Codificación de longitud fija a variable

Códigos no-unívocamente decodificables: requieren marcar la longitud de la palabra

Códigos autosincronizados: indican la longitud en el código y por tanto son unívocamente decodificables

Códigos instantáneos: detectan el final de palabra código con la misma palabra

Códigos prefijo: ninguna palabra es prefijo de otra (=> código instantáneo y unívocamente decodificables)

- Si $\Sigma D^{-li} \leq 1$ => existe un código prefijo

Ejemplo: p(x)={1/2, ¼, 1/8, 1/16, 1/16}

Codificación de longitud fija a variable: ejemplo

símbolo	Cod 0	Cod 1	p_i	Cod 2	Cod 3	-log p_i	Cod 4
a_1	000	00	½	1	1	1	0
a_2	001	01	¼	10	01	2	10
a_3	010	10	1/8	100	001	3	110
a_4	011	11	1/16	1000	0001	4	1110
a_5	100	110	1/16	10000	00001	4	1111
	$\bar{l}$=48/16 =3	$\bar{l}$=33/16 =2,063	H=15/8 =1,875	$\bar{l}$=31/16 =1,938	$\bar{l}$=31/16 =1,938	ΣD^{-li} =1 ≤1	$\bar{l}$=30/16 =1,875

Código 1: no es unívocamente decodificable (a_5a_5 == $a_4a_2a_3$)

Código 2: autosincronizado, no instantáneo

Código 3: autosincronizado, instantáneo = prefijo

Código 4: autosincronizado, instantáneo = prefijo y $\bar{l}$ mínima

Índice

Codificación Huffman

La idea es intentar dar a cada símbolo una l_i lo más cercana posible a $-\log p_i$ (de forma que $\bar{l}$ tenderá a H) y que además se genere un código prefijo.

Los códigos Huffman son los códigos prefijo (unívocamente decodificables e instantáneos) de $\bar{l}$ mínima.

Se trata de un algoritmo sencillo para alfabeto binario, aunque se puede generalizar fácilmente a alfabetos D-arios.

Requiere conocer la fdp de la fuente $\{p_i\}$ i=1..N

Se puede demostrar que $H(X) \leq \bar{l} \leq H(X)+1$

Pueden existir varios códigos Huffman equivalentes

- Cambiando el orden de asignación de 0's y 1's
- Si varios elementos tienen la misma probabilidad se pueden ordenar de forma distinta
- …

Algoritmo de codificación Huffman binaria

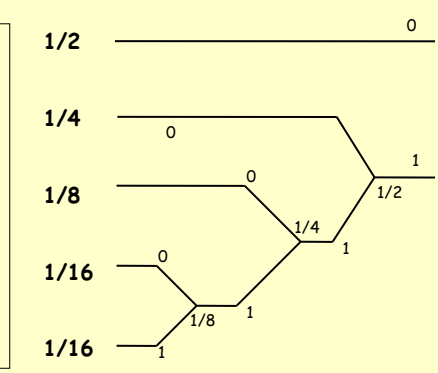
1. Ordenar los símbolos en orden descendiente de p_i.
2. Juntar los dos con probabilidad menor en un nuevo grupo.
3. Repetir 1 y 2 hasta que queden don grupos.
4. Asignar 0 y 1 a cada uno de los dos grupos resultantes.
5. Si el grupo es el resultado de juntar dos, añadirle 0 y 1 a cada uno de sus predecesores; y repetir hasta el final

- Para lograr códigos prefijo hay que asignarle 0/1 al mayor/menor o viceversa de manera consistente.
  - Mayor/menor no arriba/abajo!!!!!

Ejemplo: {1/2, ¼, 1/8, 1/16, 1/16} = {0, 10, 110, 1110, 1111} = {1, 01, 001, 0001, 0000} = {0, 11, 101, 1000, 1001}

Algoritmo de codificación Huffman binaria: ejemplo

A1 = 0
A2 = 10
A3 = 110
A4 = 1110
A5 = 1111



A1 0	½ 0	½ 0	½ 0	½ 0
A2 10	¼ 10	¼ 10	¼ 10	½ 1
A3 110	1/8 110	1/8 110	¼ 11	
A4 1110	1/16 1110	1/8 111		
A5 1111	1/16 1111			

Algoritmo de codificación Huffman binaria: ejemplo 2

p_i={0.3, 0.25, 0.2, 0.2, 0.05}
H= 2,16

C1={00, 01, 11, 100, 101} $\bar{l}$ = 2,25
C2 = {00, 10, 010, 11, 011} $\bar{l}$ = 2,25

Ejercicios de clase 5: Algoritmo codificación Huffman binaria

Calcule un código Huffman binario para la siguiente secuencia de símbolos: ABRACADABRA. H(X)=2,0404

Ejercicios de clase 5 - Algoritmo codificación Huffman binaria: solución

Calcule un código Huffman binario para la siguiente secuencia de símbolos: ABRACADABRA. H(X)=2,0404

Codificación Huffman con extensión de fuente

No se codifican los símbolos de la fuente X, sino bloques de n símbolos, creando una nueva fuente "virtual" X^n: fuente extendida

Para dicha fuente extendida X^n se cumple que

$H(X^n) \leq \bar{l}_n \leq H(X^n)+1$

Como $\bar{l}_n$ es la longitud media de palabras de n símbolos se cumple que

$\bar{l}_{efectiva} = \bar{l} = \bar{l}_n/n$

Si la fuente es sin memoria $H(X^n) = n\,H(X)$ [si con memoria igual pero con la tasa de entropía]

=> $nH(X) \leq \bar{l}_n \leq nH(X)+1$ => $H(X) \leq \bar{l} \leq H(X)+1/n$ => según n aumenta $\bar{l}$ tiene a H(X)

Codificación Huffman con extensión de fuente: ejemplo

Sea una fuente con p_i={1/2,1/4,1/4}

Calcular su entropía.

Calcular un código Huffman y su longitud media.

Calcular un código Huffman de su extensión de fuente de orden 2 y su longitud media (efectiva).

¿Cuál sería la extensión necesaria para asegurar una longitud media efectiva de 1,6 (a 2 décimas de la entropía)?

Codificación Huffman con extensión de fuente: ejemplo - solución

Sea una fuente con p_i={1/2,1/4,1/4}

Calcular su entropía.

- H(X)=1,5

Calcular un código Huffman y su longitud media.

- {a,b,c}={0, 10, 11}
- $\bar{l}$ =H(X)=1,5

Calcular un código Huffman de su extensión de fuente de orden 2 y su longitud media (efectiva).

- {aa,ab,ac,ba,bb,bc,ca,cb,cc}={00,010,011,100,1010,1011,110,1110, 1111}
- $\bar{l}_2$=H(X^2)=3
- $\bar{l}_e$ =H(X^2)/2=1,5

Codificación Huffman con extensión de fuente: ejercicio propuesto 2

Sea una fuente con p_i={1/3,1/3,1/3}

Calcular su entropía.

Calcular un código Huffman y su longitud media.

Calcular un código Huffman de su extensión de fuente de orden 2 y su longitud media (efectiva).

Codificación y decodificación de códigos Huffman (I)

Hasta ahora se ha visto el algoritmo Huffman que permite crear un código prefijo de $\bar{l}$ mínima ($H(X) \leq \bar{l} \leq H(X)+1/n$ – siendo n el orden de la extensión de fuente).

Sin embargo no hay que olvidar la etapa (más o menos) trivial de codificación y decodificación (previa y posterior respectivamente a la codificación de canal, modulación digital, transmisión/recepción, demodulación digital, y decodificación de canal).

Codificación y decodificación de códigos Huffman (II)

Para codificar simplemente es necesario mantener una estructura de datos que permita manipular (en esta caso asignar) dos datos: el símbolo y el código Huffman correspondiente. El codificador irá leyendo símbolos (o grupos de n) e irá generando una serie de palabras códigos.

La decodificación es el proceso inverso, ya que la tabla permitirá ir asignando símbolos a cada palabra código de entrada (recordar aquí que al ser el código Huffman un código prefijo, la decodificación en instantánea y no requiere mecanismos adicionales de sincronización).

Ejercicio de clase 6: Codificación y decodificación Huffman

Calcule el código resultante de codificar: ABRACADABRA, con el siguiente código Huffman:

A: 1, B: 01, R: 000, C: 0010, D: 0011

- Comente el resultado

Calcule la palabra emitida (sin error de transmisión) siendo la secuencia recibida 000001101010111001100 10

- Comente el resultado

Ejercicio de clase 6: Codificación y decodificación Huffman - solución

Calcule el código resultante de codificar: ABRACADABRA, con el siguiente código Huffman:

A: 1, B: 01, R: 000, C: 0010, D: 0011

-
-
-

Calcule la palabra emitida (sin error de transmisión) siendo la secuencia recibida 00000110101011100110010

-
-
-

Codificación y decodificación de códigos Huffman (III)

Hay que tener en cuenta que la tabla debe de estar presente en el codificador y decodificador, de forma que siempre hay dos opciones:

- Se calcula para cada realización concreta de la fuente la tabla (el código) y se transmite antes de empezar a transmitir la secuencia codificada.
  - - mayor tiempo de proceso (dos pasadas, generación de código)
  - + mayor compresión
  - - mayor sobrecarga de transmisión (cabeceras)
  - - requiere protocolo de sincronización (e.g., fin de secuencia, inicio de tabla, fin de tabla, …)
- La tabla es fija, de forma que no hay que transmitirla (en este caso es posible que cada realización de la fuente no tenga las estadísticas exactas –más exactas cuanto más símbolos se emitan-, pero se reduce la sobrecarga de transmisión de las tablas).
  - + menor tiempo de proceso
  - - menor compresión
  - + menor sobrecarga de transmisión (cabeceras)
  - + no requiere código de sincronización (solamente transmite/recibe bits de palabras código)
- Una opción “intermedia” es tener varias tablas (códigos) en ambos extremos (codificador y decodificador) y se usa una u otra en función de los estadísticos concretos de cada realización. En este caso, antes de transmitir cada secuencia codificada solamente es necesario identificar la tabla usada en codificación
  - + menor tiempo de proceso (requiere calcular estadísticos para seleccionar la tabla, pero no generar el código)
  - + mayor compresión (aunque menos que en el primer caso)
  - + menor sobrecarga de transmisión (cabeceras) (aunque mínimamente mayor que el segundo caso)
  - - requiere protocolo de sincronización (e.g., fin de secuencia, código de escape para el valor del índice de tabla, …)

Ejercicio de clase 7: Codificación y decodificación Huffman

Siendo el siguiente código Huffman: A: 1, B: 01, R: 000, C: 0010, D: 0011; correspondiente a una fuente con los siguientes estadísticos {p(A) : 5/11, p(B) : 2/11, p(R) : 2/11, p(C) : 1/11, p(D) : 1/11}

Evalúe las posibilidades de transmisión de las siguientes secuencias

- ABRACADABRA
- ACABADCABRA
- DABARARARAR

Comente los resultados

Ejercicio de clase 7: Codificación y decodificación Huffman - solución

Codificación Huffman – ejercicio propuesto 3

Escribir el pseudo-código de:

- el algoritmo de codificación Huffman (para crear el código)
- Un codificador Huffman (con código –tabla- único por secuencia)
- Un decodificador Huffman (con código –tabla- único por secuencia)

Índice

Índice

Modificación de fuente

El ejemplo de extensión de fuente para codificación Huffman mostraba una de las posibles modificaciones de fuente, la extensión mediante bloques de n símbolos.

Sin embargo existen más algoritmos de modificación de fuente, siendo dos de los más populares la codificación mediante carreras de símbolos (Run-Lenght enConding - RLC) y mediante llevar al frente (Move To Front - M2F).

No se debe olvidar que la modificación de fuente puede dar lugar a expansión de la misma, y que siempre irá seguida de un algoritmo de codificación de fuente (al cual se adapta la fuente).

Modificación de fuente: RLC

El principio común de las modificaciones de fuente por carreras de símbolos es sustituir cada grupo de símbolos por el número de ocurrencias y el símbolo. Existen numerosas variantes en función de cómo se aplique el principio común.

- Se aplica a todos los símbolos
  - ABCDDDDDABC => 1A1B1C5D1A1B1C (expansión)
- Se aplica a un único símbolo
  - ABCDDDDDABC => ABC5DABC (compresión)
- Se aplica a un subconjunto de símbolos
- Se aplica solamente cuando el tamaño de la carrera de símbolos es mayor que un límite

Las diversas variantes están pensadas para adaptarse a las diversas posibles características de las fuentes y los codificadores de fuentes que se usen.

RLC: consideraciones a tener en cuenta

Si las carreras de símbolos (o de la mayoría) no son frecuentes, la fuente resulta expandida.

Hay que prestar especial atención al sincronismo o diferenciación entre el símbolo de la fuente original y el código para indicar la longitud de la carrera

- Si la fuente no tiene caracteres numéricos, se pueden usar los dígitos sin más
  - AAABBAABCD (10 símbolos) => 3A2B2A1B1C1D (12 símbolos)
  - AAABBBBBCD (10 símbolos) => 3A5B1C1D (8 símbolos)
  - AAA2BCD1AA (10 símbolos) => 3A121B1C1D112A (15 símbolos) y no unívocamente decodificable => AAAB……BCDA…..A ó AAA1…error! … pero fácilmente solucionable …
- Si se codifica cada símbolo con longitud fija (parece sensato al tener posteriormente la codificación de fuente) se puede asignar el primer símbolo al tamaño y el segundo al símbolo de la fuente original (se aplica a todos los símbolos y para cualquier tamaño de carrera)
  - AAA2BCD1AA (10 símbolos) => (3,A)(1,2)(1,B)(1,C)(1,D)(1,1)(2,A) (14 símbolos)
  - El carácter de control/escape queda determinado por la representación binaria de los símbolos: el 12 no serán 2 símbolos (caracteres de 1 byte) sino '00001100'
  - Las carreras (de 1 a 256 no 0..255 –aunque se verá que en ciertos casos se indican carreras de 0 –en ese caso la carrera irá de 1..255) no pueden ser mayores de 256 (si 8 bits/símbolo): $2^{nbits/símbolo}$ en general
- Si no se aplica a todos los símbolos, puede valer con un código de escape (¿qué ocurre si el código de escape forma parte del alfabeto de símbolos de la fuente original?) para saber a cuales se aplica y a cuales no.
- Si a cada símbolo se le añade un bit de escape, se puede saber si lo que sigue es símbolo o tamaño de la carrera (entero de 2^{nbits}).

RLC: ejemplos

2. all is too well
- 121.1 1a2l1 1i1s1t2o1 1w1e2l
  - Expande y requiere trabajar a nivel “byte” no carácter.
- 2. a@2l is t@2o we@2l
  - En este ejemplo deja todo igual, pues solamente hay carreras de 2 y el carácter de escape no ayuda a comprimir (ni a mejorar los estadísticos)
  - Queda el problema de la posible aparición de la secuencia de escape en el texto original

hola@@@@@hola
- hola@5@hola
  - hola@@@@@hola: correcto

josem.martinez@uam.es
- josem.martinez@uam.es
  - Como debe trabajar con el símbolo binario (el ASCII de ‘u’ es 117) tras el de escape decodificaría: josem.martineza.....am.es
- josem.martinez@@uam.es
  - Al leer @ (ASCII 64) como carrera tiene que detectar que no es una carrera permitida (no la permito –mínima pérdida de eficiencia- para quitar el problema de no poder tener las ventajas del código de escape
  - Si aparece una carrera de 64 se codifica como carrera de 63 y luego un símbolo sin carrera
  - Si no todos los símbolos de entrada se usa se puede usar uno de los no usados como escape (en texto generalmente no se usan los no imprimibles, pero cuidado si estamos trabajando con ficheros binarios).

RLC: Zero-RLC

Se trata de un RLC donde solamente se codifican las carreras de un único símbolo de la fuente (el cero, aunque no tiene porqué tener ese valor). Está muy relacionada con otra técnica muy usada que, si bien se podría considerar una modificación de fuente, generalmente se agrupa dentro de las técnicas de transformación de dominio: la predicción. Esta técnica lo que hace es codificar la diferencia entre el símbolo actual y símbolos anteriores. Si los símbolos vecinos tienden a ser similares (e.g., ciertas zonas de señales de voz e imagen), un gran número de símbolos serán nulos (ceros), de forma que existirán con alta probabilidad carreras de ceros. Adicionalmente la predicción hace que existan mayoritariamente símbolos de pequeña magnitud.

RLC: Zero-RLC

Esta técnica se usa frecuentemente en codificación de audio e imagen. En codificación de imagen, tiene también aplicación en codificación en dominio transformado (ver más adelante en este mismo tema).

Típicamente esta modificación de fuente suele generar símbolos de la fuente extendida de la siguiente forma:

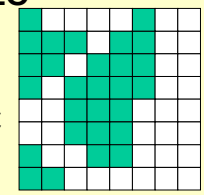
- (números de ceros antes del símbolo no nulo, símbolo no nulo)
- Ejemplo: 121200000300100002
  - 121200000300100002 (18 bytes –si el rango de símbolos es 0..255)
  - (0,1)(0,2)(0,1)(0,2)(5,3)(2,1)(4,2) -> 14 bytes (si carreras <256)
- Los codificadores posteriores suelen codificar los símbolos (x,y), pudiendo estar las posibles carreras limitadas para poder usar códigos fijos en ciertos casos (subóptimos desde el punto de vista de codificación de fuente).

Ejercicios de clase 8: RLC y ZRLC

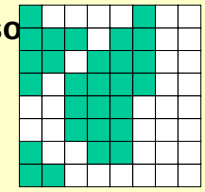
Sea la imagen de 8x8 de la figura.

Comparar la fuente original con su RLC y ZRLC

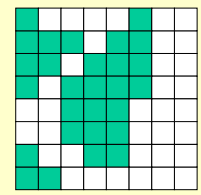
Comente los resultados



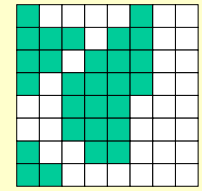
Ejercicios de clase 8: RLC y ZRLC – so



Ejercicios de clase 8: RLC y ZRLC – solución (II)



Ejercicios de clase 8: RLC y ZRLC – solución (III)



Sea la imagen de 8x8 de la figura.
Comparar con su ZRLC
Comente los resultados

RLC (por columnas y vectorizado)

- (4,V)(2,B)(2,V)(1,B)(2,V)(4,B)(1,V)(1,B)(1,V)(1,B)(3,V)**(4,B)** (5,V)**(2,B)** (6,V)(1,B)(4,V) **(20,B)**

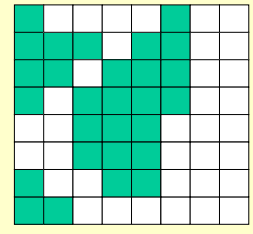
ZRLC (B se toma como símbolo nulo)

- (0,V) (0,V) (0,V) (0,V) (2,V) (0,V) (0,V) (1,V) (0,V) (0,V) (4,V) (0,V) (1,V) (0,V) (1,V) (0,V) (0,V) (0,V) (4,V) (0,V) (0, V) (0,V) (0,V) (0,V) (0,V) (1,B) (0,V) (0,V) (0,V) (0,V) (20, ???)
- (0,V) (0,V) (0,V) (0,V) (2,V) (0,V) (0,V) (1,V) (0,V) (0,V) (4,V) (0,V) (1,V) (0,V) (1,V) (0,V) (0,V) (0,V) (4,V) (0,V) (0, V) (0,V) (0,V) (0,V) (0,V) (1,B) (0,V) (0,V) (0,V) (0,V) (EOB)
- H(X) = …

RLC y ZRLC – ejercicio propuesto 4

Ejercicio de examen Septiembre 2007

- Sea la imagen de 64 píxeles binarios de la figura la cual se quiere codificar sin pérdidas mediante el encadenamiento de una extensión de fuente tipo RLC (Run-Length Coding) y una codificación Huffman. La extensión RLC considerará solamente carreras de hasta 4 símbolos e incluirá un símbolo EOF (End Of File). El código Huffman estará perfectamente adaptado a esa fuente (en un caso real a cada realización de la fuente).



Índice

Modificación de fuente: M2F (Move 2 Front)

La idea es la de generar una nueva fuente basada en la posición dinámica de los símbolos de la fuente original.

- Los símbolos de la fuente A se sustituyen por la posición del símbolo en un alfabeto dinámico A^p, esto es, que va reordenando sus símbolos.
  - Cada símbolo se codifica con el número de símbolos que lo preceden en la lista del alfabeto de símbolos A^p, lo que da lugar a una lista de enteros con igual rango que la fuente original [N=card(A)=card(A^p)] que se puede codificar con un algoritmo Huffman, artimético, … posterior.
- Es un algoritmo localmente adaptativo: funciona bien si hay símbolos iguales cercanos.
  - M2F intenta mejorar las propiedades de la fuente para su codificación posterior, partiendo de la hipótesis de que símbolos cercanos sean iguales. Si se cumple, los números se concentrarán en valores bajos (0,1,2,3, …) y por lo tanto la fuente se hace más asimétrica a nivel probabilístico y se pueden mejorar la compresión siguiente (-bits a +frecuentes)
- Se codifica por posición en el alfabeto dinámico
  - N=card(A)=card(A^p) => N símbolos como la fuente, de forma que en longitud fija son equivalentes (pero solamente usaremos longitud fija en fuentes uniformes y M2F difícilmente dará lugar a una fuente uniforme).
- En el peor de los casos es un poco peor que Huffman sin M2F, mientras que en el mejor de los casos muchísimo mejor.

M2F: Algoritmo de "codificación"

1. Se iniciliaza $A^p = A$
2. Se codifica el símbolo x perteneciente a A por su posición (número de símbolos que le preceden) en A^p : 0..N-1
3. El símbolo x se mueve al frente: a la primera posición de A^p.
4. Se repiten 2-3 hasta que no hay más símbolos
5. Se pasa la secuencia resultante al codificador de fuente

[Bentley 1986]

Según se calcula la secuencia de posiciones se puede ir calculando la estadística de la fuente extendida

Existen variantes al algoritmo original: mover k posiciones hacia delante, esperar c repeticiones antes de mover, trabajar por cadenas (palabras) en lugar de símbolos (letras), …

M2F: Algoritmo de decodificación

Se trata del proceso inverso, posterior a la decodificación de fuente. Los símbolos de entrada son enteros (y) y la salida símbolos del alfabeto A.

1. Se iniciliaza $A^p = A$
2. Se decodifica el símbolo x perteneciente a A que se encuentre en la posición y (número de símbolos que le preceden) en A^p
3. El símbolo x se mueve al frente: a la primera posición de A^p.
4. Se repiten 2-3 hasta que no hay más símbolos

Algoritmo M2F: ejemplo

A={a,b,c} y X=aababcccb

Fuente original:

- uniforme => 3 símbolos => 2 bits/símbolos si codificación fija.
- H(X)=log 3 = 1,585

Fuente M2F:

- 001112001 => pi={4/9, 4/9,1/9}
- H(X)= 1,391 (pequeña mejora por ejemplo corto)

símbolo A	Orden en A^p anterior	A^p={abc}
A	0	abc
A	0	abc
B	1	bac
A	1	abc
B	1	bac
C	2	cba
C	0	cba
C	0	cba
B	1	bca

Ejercicios de clase 9: M2F

Sea la fuente con alfabeto A={a,b,c,d,m,n,o,p} y la siguiente secuencia resultante de la realización de la misma

- Abcddcbamnopponm

"Codificarla" mediante M2F y comparar sus estadísticos con los de la fuente original

"Decodificarla".

Comente los resultados.

Ejercicios de clase 9: – solución (I)

"Codificarla" mediante M2F y comparar sus estadísticos con los de la fuente original

Fuente original uniforme

-
-

Fuente M2F: {0,1,2,3,4,5,6,7}

-
-

símbolo A	Orden en A^p anterior	A^p={abcdmnop}
A		
B		
C		
D		
D		
C		
B		
A		
M		
N		
O		
P		
P		
O		
N		
m		

Ejercicios de clase 9: – solución (II)

"Decodificarla".

Orden en A^p anterior	símbolo A	A^p={abcdmnop}

M2F - ejercicio propuesto 5

Ejercicio de examen Febrero 2006

Sea la secuencia **abaacdcdcd**.

- Codifíquela mediante el algoritmo Huffman
- Codifíquela mediante la aplicación en cadena del algoritmo Move To Front (M2F) y del algoritmo Huffman. Compare la ganancia obtenida por la inserción del algoritmo M2F.

Índice

Codificación aritmética

Los códigos Huffman son los mejores para codificación símbolos a símbolo, conociendo a priori (o calculándola) la distribución de probabilidades de los símbolos (p_i).

Los códigos Huffman solamente logran $\bar{l} = H(X)$ si $p_i=2^{-li}$, sino logran $H(X) \leq \bar{l} \leq H(X)+1/n$ (siendo n el orden de la extensión de fuente).

La codificación aritmética solventa el problema de asignar códigos con li entera (lo cual es posible en caso muy concretos) mediante la asignación de un único código (largo) a toda la realización de la fuente (secuencia) o en el caso realista a una larga secuencia de símbolos. Al igual que Huffman requiere el conocimiento a priori de la distribución de probabilidades de los símbolos (aunque existen versiones adaptativas)

Codificación aritmética: principios

La codificación se basa en codificar un intervalo.

Se parte de un intervalo y se va reduciendo el intervalo en función de la pi de los símbolos que van siendo codificados. Cuando se terminan los símbolos (o cuando se hace necesario por temas de implementación práctica) se codifica el intervalo resultante (realmente uno de sus extremos).

Cuanto más se reduce el intervalo serán necesarios más símbolos para representarlo

- Intervalo [a,b) se puede representar con sus extremos, o un extremo y su tamaño (b-a)
  - [0;1) => 0 y 1 (1 bit para cada uno)
  - [0.1;0.512) => 0,1 y 0.412 => 1 y 412 (necesito al menos 9 bits para el segundo)
  - [0.12575; 0.1257586) => 12575 y 0000086 (no 86!!! Ya que puede darse el rango 0..9999999)

Para aplicar el principio general de codificación de fuente (dar menos bits a los más frecuentes):

- Los símbolos de p_i alta reducen poco el intervalo
- Los símbolos de p_i baja reducen mucho el intervalo

Codificación aritmética: algoritmo de codificación

1. Se asocia a cada símbolo del alfabeto un intervalo proporcional a su probabilidad
   - $I_1 = [0;p_1) - I_2 = [p_1;p_1+p_2) - I_3 = [p_1+p_2; p_1+p_2+p_3)$ …
   - $I_i = [a_i;b_i) = [\Sigma_{j=1}^{i-1} p_j; \Sigma_{j=1}^{i} p_j)$
2. Se empieza con el intervalo [a;b) = [0;1)
3. Se codifica cada símbolo x_i (con intervalo $[a_i;b_i)$) modificando $[a_{actual};b_{actual})$ mediante la siguiente fórmula
   - $[a_{nuevo};b_{nuevo}) = [a_{actual} + (b_{actual}-a_{actual})\, a_i \; ; \; a_{actual}+(b_{actual}-a_{actual})\, b_i)$
4. Cuando se llega al intervalo final, se codifica el mismo: cualquiera de sus extremos identifica (para codificación aritmética) unívocamente el código obtenido. Generalmente se utiliza el límite inferior del intervalo, codificándose como un entero (0,xxxxx).

Codificación aritmética: ejemplo codificación

Sea una fuente $X=\{a_1,a_2,a_3\}$ con $p_i=\{0,4 ; 0,5 ; 0,1\}$. Codificar la secuencia $a_2a_2a_2a_3$.

Se asocian los siguientes intervalos a cada símbolo

- $I_1 = [0;0.4) - I_2 = [0.4;0.9) - I_3 = [0.9; 1)$
- (cualquier otra asociación valdría, siempre que la secuencia nunca terminase con el símbolo asociado al intervalo que empieza en 0).

0. [a;b) = [0;1)
1. a2: [0.4;0.9) => [a;b) = [0+1*0.4; 0+1*0.9) = [0.4;0.9)
2. a2: [0.4;0.9) => [a;b) = [0.4+0.5*0.4; 0.4+0.5*0.9) = [0.6;0.85)
3. a2: [0.4;0.9) => [a;b) = [0.6+0.25*0.4; 0.6+0.25*0.9) = [0.7;0.825)
4. a3: [0.9; 1) => [a;b) = [0.7+0.125*0.9; 0.7+0.125*1) = [0.8125;0.8250)
5. 0.8125 => **8125**

Ejercicios de clase 10: Codificación aritmética - codificación

Codificar la secuencia: ABRAR

- $[a_{nuevo};b_{nuevo}) = [a_{actual} + (b_{actual}-a_{actual})\ a_i\ ;\ a_{actual}+(b_{actual}-a_{actual})\ b_i)$

Ejercicios de clase 10: Codificación aritmética- codificación - solución

ABRAR => A=0.4; R=0.4; B=0.2

Codificación aritmética: algoritmo de decodificación

1. Se recibe el entero y se asocia a '$a_{i\ actual}$'
2. Se identifica en que intervalo se encuentra $a_{i\ actual}$ => $[a_{actual};b_{actual})$ y se decodifica el símbolo al que corresponde
3. Se deshace el paso que llevó a ese intervalo ($a_{nuevo}= a_{actual} + (b_{actual}-a_{actual})\ a_i$) =>
   1. $a_i = (a_{i\ actual} – a_{actual})/(b_{actual}-a_{actual})$
   2. $a_{i\ actual} = a_i$
4. Se repiten 2 y 3 hasta que $a_{i\ actual} = 0$

Codificación aritmética: ejemplo de decodificación

0. inicialización
   - $I_1 = [0;0.4) – I_2 = [0.4;0.9) – I_3 = [0.9; 1)$
   - 8125 => 0.8125
1. $a_{i\ actual} = 0.8125$ pertenece a I_2 => $[a_{actual};b_{actual})=[0.4;0.9)$
   - ***Decodifico a_2***
   - $a_i = (0.8125 – 0.4)/0.5 = 0.825$
2. $a_{i\ actual} = 0.825$ pertenece a I_2 => $[a_{actual};b_{actual})=[0.4;0.9)$
   - ***Decodifico a_2***
   - $a_i = (0.825 – 0.4)/0.5 = 0.85$
3. $a_{i\ actual} = 0.85$ pertenece a I_2 => $[a_{actual};b_{actual})=[0.4;0.9)$
   - ***Decodifico a_2***
   - $a_i = (0.85 – 0.4)/0.5 = 0.9$
4. $a_{i\ actual} = 0.9$ pertenece a I_3 => $[a_{actual};b_{actual})=[0.9;1)$
   - ***Decodifico a_3***
   - $a_i = (0.9 – 0.9)/0.1 = 0$

Ejercicios de clase 11: Codificación aritmética - decodificación

Decodificar la secuencia: 21568

- $I_A = [0;0.4) – I_B = [0.4;0.6) – I_R = [0.6; 1)$
- $a_i = (a_{i\ actual} – a_{actual})/(b_{actual}-a_{actual})$

Ejercicios de clase 11: Codificación aritmética- decodificación - solución

Codificación aritmética: consideraciones

Importante: el símbolo asociado al intervalo $I_1 = [0;p)$ no puede ser el último de la secuencia a codificar.

- ABRAR => A=0.4; R=0.4; B=0.2
  - Se asocian los siguientes intervalos a cada símbolo
    $I_R = [0;0.4) - l_B = [0.4;0.6) - l_A = [0.6; 1)$
  - Codificar y decodificar
    Codifica 7792 ([0,7792;0,78432))
    Decodifica solamente ABRA

Si no se puede controlar el último símbolo, es necesario añadir en el sistema de codificación un símbolo especial "eof" con probabilidad muy pequeña (para que distorsione poco el sistema –cambia las probabilidades de la fuente-)

- Este EOF se le asocia un p=0,0001 (por ejemplo) y el intervalo [0,9999; 1). El epsilón (p) se le quita al primer o al último intervalo real (o a todos proporcionalmente). Distorsiona la fuente, pero permite ser generalista.

Codificación aritmética: codificación y decodificación

Ambos codificadores tienen que estar sincronizados, esto es, conocer las p_i (l_i).

En la práctica las secuencias pueden ser muy largas, de forma que no se podría representar en el ordenador los números con los que se trabaja. Por ello en las implementaciones prácticas se tiene que poner un límite al número de símbolos y reiniciar la codificación cada cierto tiempo.

- Límite fijo de símbolos
- Límite guiado por la precisión de cálculo (codificador y decodificador deben tener la misma)
- …

El código que representa el intervalo se suele representar en binario (como un entero de cierto número de bits – precisión), pero se podría codificar como una cadena de dígitos (4 bits por dígito – desaprovecho -) y aplicar otro algoritmo de codificación sin pérdidas para comprimir esa cadena.

Índice

Codificación Lempel-Ziv

Para los códigos estadísticos (e.g., Huffman, arítmético) es necesario conocer la distribución de probabilidades de la fuente, lo que implica la necesidad de dar dos pasadas o utilizar estadísticas medias.

Para fuentes con memoria es mejor trabajar con bloques de símbolos, …

- Si se codifica símbolo-a-símbolo no se hace uso de la memoria (dependencia) de la fuente y por lo tanto el código es sub-óptimo.

… pero la complejidad de la extensión (vectorial) de una fuente aumenta con n.

Los códigos universales son códigos de longitud variable a fija que no hacen uso de estadísticas previas

- Número de símbolos de entrada variable
- Número de símbolos de salida fija (para número de entrada variable)

Los códigos Lempel-Ziv (existen numerosas variantes) son los códigos universales más conocidos y utilizados.

Codificación LZ: algoritmo de codificación

Se basa en registros y diccionarios.

0. Se reserva el índice 0 del diccionario para indicar que es símbolo nuevo (no tiene palabra previa)
1. Se observa la entrada (símbolo) y se añade al registro
2. Si la palabra del registro está en el diccionario se pasa a 1.
3. Si la palabra del registro no está en el diccionario
   1. Se codifica la palabra anterior con su índice del diccionario (0 si no hay frase anterior: la entrada observada es un símbolo único que no está en el diccionario) y el último símbolo (entrada observada)
   2. Se introduce en el diccionario la nueva palabra
   3. Se limpia el registro
4. Se vuelve a 1 hasta que se acaban los símbolos de entrada

Codificación LZ: ejemplo codificación

Siendo la fuente X={A,B,C,D,R}, codificar ABRACADABRA

Entrada	Registro	Diccionario	Salida
A	A	|A|…	(0,A)
B	B	|A|B|…	(0,B)
R	R	|A|B|R|…	(0,R)
A	A		
C	AC	|A|B|R|AC|…	(1,C)
A	A		
D	AD	|A|B|R|AC|AD|…	(1,D)
A	A		
B	AB	|A|B|R|AC|AD|AB|…	(1,B)
R	R		
A	RA	|A|B|R|AC|AD|AB|RA|…	(3,A)

A primera vista no comprime nada, pero según aumenta el tamaño de la secuencia se comprime ($\bar{l}_n$ tiene a nH(x) => $\bar{l} = \bar{l}_n/n$ tiende a H(X))

Ejercicios de clase 12: Codificación LZ

Codificar LZ la secuencia: 010000110000…

Ejercicios de clase 12: Codificación LZ – solución

Codificar LZ la secuencia: 010000110000 …

Codificación LZ: algoritmo de decodificación

1. Se observa la entrada (código: índice+símbolo) y se añade al registro
2. Se extrae la palabra del diccionario indicada por el índice, se le añade el símbolo y se pone en la salida
3. Se añade la palabra decodificada al diccionario
4. Se repite 2 y 3 hasta que no quedan símbolos

Codificación LZ: ejemplo decodificación

No requiere el registro, pero si el diccionario

Secuencia codificada: (0,A) (0,B) (0,R) (1,C) (1,D) (1,B) (3,A)

Entrada	Salida	Diccionario
(0,A)	(null)+A	|A|…
(0,B)	(null)+B	|A|B|…
(0,R)	(null)+R	|A|B|R|…
(1,C)	A+C	|A|B|R|AC|…
(1,D)	A+D	|A|B|R|AC|AD|…
(1,B)	A+B	|A|B|R|AC|AD|AB|…
(3,A)	R+A	|A|B|R|AC|AD|AB|RA|…

Salida (secuencia decodificada): A.B.R.AC.AD.AB.RA….

Ejercicios de clase 12: Codificación LZ - decodificación

Decodificar LZ la secuencia: (0,0) (0,1) (1,0) (3,1) (2,0) (3,0)

Ejercicios de clase 12: Codificación LZ: decodificación – solución

Codificación LZ: consideraciones

En el ejemplo anterior 11 símbolos se codifican con 7 (índice+símbolo): además de la codificación del símbolo (idéntica a la de la fuente –longitud fija-) hay que añadir la codificación del índice. Para codificar el índice se tiene que tener en cuenta el tamaño del diccionario (tamaño diccionario ≤ 2^{nbits})

Tamaño del diccionario: el tamaño del diccionario no puede ser infinito (es necesario conocer nbits –si no adaptativo- y además no podemos tener memoria infinita). Por lo tanto cuando el diccionario se llena hay que utilizar políticas de refresco (e.g., FIFO, menos usado, …)

Símbolo EOF: es posible que el último símbolo no vacíe el registro. Por lo tanto hay que incluir un símbolo EOF en cada codificación de una secuencia "finita".

Codificación LZ: codificación del código LZ

Tras tener el código LZ (índice+código) hay que binarizar (generalmente mediante código de longitud fija) ambos elementos

- Bits para el índice: función del tamaño del diccionario
- Bits para el código: función del tamaño del alfabeto

Y viceversa en el decodificador.

Codificación LZW: algoritmo de codificación

Lempel-Ziv-Welch es una de las variantes más populares de LZ: cada cadena de entrada se codifica solamente con el índice, no siendo necesario el símbolo; pero es (algo) más complejo.

0. Se inicializa el diccionario con los símbolos del alfabeto
1. Se observa la entrada (símbolo) y se añade al registro
2. Si la palabra del registro está en el diccionario se pasa a 1.
3. Si la palabra del registro no está en el diccionario
   1. Se codifica la palabra anterior con su índice del diccionario
   2. Se introduce en el diccionario la nueva palabra
   3. Se limpia el registro dejando solamente el símbolo nuevo
4. Se vuelve a 1 hasta que se acaban los símbolos de entrada

Codificación LZW: ejemplo codificación

Siendo la fuente X={A,B,C,D,R}, codificar ABRACADABRA

Entrada	Registro	Diccionario |A|B|C|D|R|…	Salida
A	A		
B	AB	|A|B|C|D|R|AB|…	(0)
R	BR	|A|B|C|D|R|AB|BR|…	(1)
A	RA	|A|B|C|D|R|AB|BR|RA|…	(4)
C	AC	|A|B|C|D|R|AB|BR|RA|AC|…	(0)
A	CA	|A|B|C|D|R|AB|BR|RA|AC|CA|…	(2)
D	AD	|A|B|C|D|R|AB|BR|RA|AC|CA|AD|…	(0)
A	DA	|A|B|C|D|R|AB|BR|RA|AC|CA|AD|DA|…	(3)
B	AB		
R	ABR	|A|B|C|D|R|AB|BR|RA|AC|CA|AD|DA|ABR|	(5)
A	RA		
???			
EOF	RA+EOF	|A|B|C|D|R|AB|BR|RA|AC|CA|AD|DA|ABR|	(7)

Codificación LZW: algoritmo de decodificación

1. Se observa la entrada (código: índice), se extrae la palabra del diccionario indicada por el índice, se pone en la salida y se añade al registro
2. Si la palabra del registro está en el diccionario se pasa a 1
3. Si la palabra del registro no está en el diccionario, se añade al diccionario la palabra formada por la palabra antigua y el primer símbolo de la nueva, y se limpia el registro dejando únicamente la palabra nueva
4. Se repite 1, 2 y 3 hasta que no quedan símbolos

Codificación LZW: ejemplo decodificación

Siendo la fuente X={A,B,C,D,R}, codificar ABRACADABRA

Entrada	Salida	Registro	Diccionario
			|A|B|C|D|R|…
(0)	A	A	
(1)	B	(A)B	|A|B|C|D|R|AB|…
(4)	R	(B)R	|A|B|C|D|R|AB|BR|…
(0)	A	(R)A	|A|B|C|D|R|AB|BR|RA|…
(2)	C	(A)C	|A|B|C|D|R|AB|BR|RA|AC|…
(0)	A	(C)A	|A|B|C|D|R|AB|BR|RA|AC|CA|…
(3)	D	(A)D	|A|B|C|D|R|AB|BR|RA|AC|CA|AD|…
(5)	AB	(D)AB	|A|B|C|D|R|AB|BR|RA|AC|CA|AD|DA|…
(7)	RA	(AB)RA	|A|B|C|D|R|AB|BR|RA|AC|CA|AD|DA|ABR|…
(??)	??	(RA)	

Codificación de fuente sin pérdidas: ejercicio propuesto 6

Codificar y decodificar la secuencia BABACABABA mediante los algoritmos:

- Huffman
- Huffman con extensión de orden 2
- M2F seguido de Huffman
- Aritmética
- Lempel-Ziv
- Lempel-Ziv-Welch

Índice

Índice

- *Introducción*
- *Fundamentos*
- *Algoritmos de codificación de fuente sin pérdidas*
- Teoría Tasa-Distorsión
  - **Introducción**
  - Entropía diferencial
  - Función Tasa-Distorsión

    Introducción, Distorsión, Teorema Tasa-Distorsión
- Cuantificación
- Codificadores

Teoría Tasa-Distorsión: Introducción

H límite codificación sin pérdidas, pero si:

$R_{sp} = \bar{I} * f_s > R_{txcanal} \leq C$ (o $R_{sp} = \bar{I} *$ #símbolos >capacidad disco) no podré transmitir sin errores (pérdidas)

Por lo tanto se hace necesario pasar a técnicas de codificación de fuente con pérdidas ($\Rightarrow$ **distorsión de X**) que permitan no desbordar la capacidad del canal

- Para controlar dentro de lo posible la distorsión, y no dejarla sujeta a los efectos de los errores del canal (por desbordamiento de la capacidad del mismos) o a los "cortes" en el sistema de almacenamiento.

Independientemente de la etapa posterior de codificación de fuente (con o sin pérdidas) que siga a la adquisición de la fuente, se pueden identificar dos posibles casos de distorsión en adquisición:

- Caso fuentes analógicas:
  - Si $f_s \geq f_{Nyquist}$ existe la posibilidad de reconstrucción perfecta
  - Pero si las muestras son números reales como mínimo se introducirá el ***error de cuantificación***, por lo que en **casos reales siempre hay distorsión**.
- Caso fuente digital:
  - Según el número de niveles de la fuente y los que soporte el sistema ($2^M \rightarrow 2^N$ $N \leq M$) puede ser necesario cuantificar a un número menor de niveles.

Índice

Entropía diferencial

La Entropía diferencial se define para funciones (vv.aa.) continuas con rango de valores (muestras) reales, usándose como equivalente de entropía, si bien no se puede interpretar de la misma forma

- Entropía diferencial
  - $h(x)=-\int f_x(x).\log f_x(x)\, dx$
  - (con 0 log 0 = 0)

De la misma definición se derivan las demás funciones relacionadas con la Teoría de la Información

- Entropía diferencial conjunta:
  - $h(x,y)=-\iint f(x,y).\log f(x,y)\, dxdy$
- Entropía diferencial condicional:
  - $h(x|y)=h(x,y)-h(y)$
- Información mútua (al contratio que las h(x), la Información Mútua se interpreta exactamente igual que en el caso de vvaa discretas)
  - $I(x;y)=h(y)-h(y|x)=h(x)-h(x|y)=I(y;x)$

Entropía diferencial: ejemplo

Sea X una v.a. continua con función densidad de probabilidad uniforme en [0,a]. Calcular su entropía diferencial.

Entropía diferencial: ejemplo - solución

Sea X una v.a. continua con función densidad de probabilidad uniforme en [0,a]. Calcular su entropía diferencial.

$h(x)=-\int f_x(x).\log f_x(x)\, dx = -\int_0^a (1/a).\log (1/a)\, dx = \log a$

Comentarios

- Si a<1 => h(x) < 0
  - versus H(X)≥0 siempre
- Si a=1 => h(x) = 0
  - Versus H(X)=0 ⇔X variable determinista (complemente predecible)

Índice

Función Tasa-Distorsión: Introducción

Si paso una v.a. continua (rango real) a un entero finito (generalmente 2^n niveles) o una v.a. discreta (rango: $0..2^m-1$) a un rango menor ($0..2^n-1$, siendo $n<m$)

- ¿cuánto se parece la versión comprimida a la original?

Dado un número de bits por símbolo o bits por segundo (bps)

- ¿cuál es la cota mínima de error (distorsión)?
- ¿cómo se alcanza?

Dado un cierto nivel de distorsión (error)

- ¿cuál es el mínimo número de bits por símbolo o bits por segundo para comprimir la fuente con ese nivel de distorsión?

Nota: El término compresión suele implicar pérdidas, mientras codificación no. En cualquier caso son términos que se utilizan indistintamente y siempre hay que leerlos en contexto.

Función Tasa-Distorsión: Distorsión (I)

La **Distorsión** se define como una **medida de fidelidad** entre la versión reconstruida de la señal comprimida y la señal original

- Medida de cuanto de separadas están dos señales (y, ŷ)
- Existen numerosas ("infinitas") medidas de distorsión
  - $\max_t |y(t)- \hat{y}(t)|$
  - $\lim (1/T) \int_{-T/2}^{T/2} |y(t)- \hat{y}(t)| \, dt$
  - $\lim (1/T) \int_{-T/2}^{T/2} (y(t)- \hat{y}(t))^2 \, dt$
  - …
- Generalmente las funciones (medidas) de distorsión son llamadas también **distancias**.

Una buena función (medida) de distorsión debe cumplir (al menos) las siguientes dos propiedades (relativamente contradictorias)

- Debe ser una buena aproximación al proceso (humano) de percepción
  - ponderación sofométrica en audio, direccionalidad y altas frecuencias en video, fase poco importante en voz pero muy importante en vídeo, …
- Debe ser suficientemente sencilla como para ser matemáticamente tratable

Función Tasa-Distorsión: Distorsión (II)

Normalmente, por simplificar, se asumen distorsiones símbolo a símbolo, esto es, la distorsión de una secuencia se calcula como la media de la distorsión de sus componentes

- $d(y^n, \hat{y}^n) = (1/n)\Sigma_{i=1}^{n} d(y_i, \hat{y}_i)$
- Se asume, además de la independencia, que la posición del error no importa en reproducción y, por lo tanto, la distorsión es independiente del contexto.
  - Esto no es cierto y por ello casi todos los estándares de compresión se basan en características de enmascaramiento de los errores por los sistemas visuales y auditivos humanos, por lo que las distancias ponderan realmente los errores contextualmente (dando más peso -importancia- a los errores en zonas "sensibles")

Como la fuente es una v.a., $d(Y^n, \hat{Y}^n)$ es una v.a. y se define la distorsión de una fuente (D) mediante la esperanza estadística de la medida de distorsión o distancia:

- $D=E[d(Y^n, \hat{Y}^n)]=(1/n) \Sigma_{i=1}^{n} E[d(Y_i, \hat{Y}_i)]$
- $D=E[d(Y^n, \hat{Y}^n)]=(1/n) \Sigma_{i=1}^{n} E[d(Y_i, \hat{Y}_i)]=E[d(Y, \hat{Y})]$
  - Si fuente estacionaria

Función Tasa-Distorsión: distancias más comunes

Aunque existen infinitas distancias, generalmente adaptadas para cada aplicación concreta, se pueden citar dos como las más comunes en discreto y continuo, respectivamente

- Distancia Hamming
  - $d_H(y, \hat{y})= \{1,0\}$ si $\{y <> \hat{y}, y= \hat{y}\}$
- Distancia error cuadrático medio
  - $d_{mse}(y, \hat{y}) = (y-\hat{y})^2$

Aunque sean comúnmente usadas no tienen porqué ser las óptimas

- Dos señales de audio retardas entre sí muy poco darían una medida de distorsión muy elevada, mientras que ese retardo puede ser inapreciable por cualquier usuario.

Función Tasa-Distorsión: Teorema R-D (I)

Sea una Fuente Sin Memoria con:

- un alfabeto A de N símbolos (N puede ser infinito) y función densidad de probabilidad (fdp) p(y)
- un alfabeto de reconstrucción A'
- una medida de distorsión (distancia) d(y, ŷ)

¿Cuál es la mínima tasa R [bits/símbolo ***–¡atención!-***] necesaria para garantizar que la distorsión media no exceda un determinado valor D?

- Obviamente: Si aumento R disminuye D y viceversa
- La relación R-D se expresa mediante la función Tasa-Distorsión R(D) o la función Distorsión-Tasa D(R) (el teorema se enuncia con R(D))

Función Tasa-Distorsión: Teorema R-D (II)

Representación esquemática

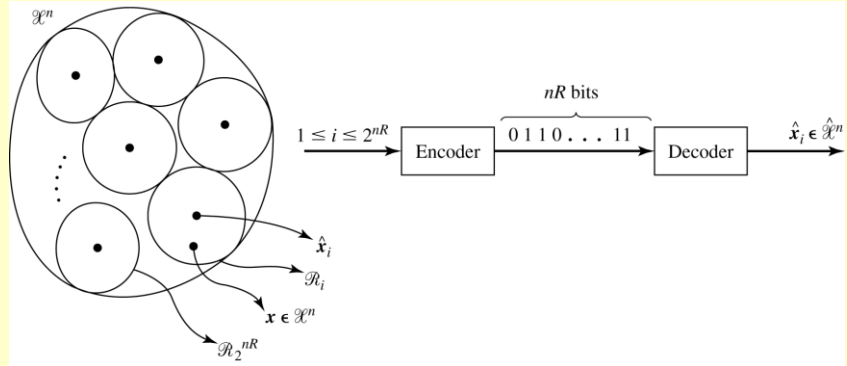


Figure 6.7

Schematic representation of the rate-distortion theorem.

Función Tasa-Distorsión: Teorema R-D (III)

Representación esquemática:

- Se divide A en 2^R regiones P_i.
  - $P_i \cap P_j = 0 \ \forall i \neq j$
  - $\cup P_i = P$
- Si y cae en P_i se transmite el símbolo i (asociado al valor $\hat{y}_i$)
- Como $1 \le i \le 2^R$ => R bits para representar N símbolos: se comprime si $R \le \log N$
  - Si $R \ge H(Y)$ posibilidad de codificación sin pérdidas
    Card(A) = N (finito)
    R = ceil(log N)
- El receptor (descompresor) asocia a cada símbolo i un valor de "reconstrucción" (no decodificador, pues reconstruye => pérdidas) predeterminado $\hat{y}_i$, de forma que la distorsión sea mínima para todo $y_i \in P_i$.
- Esto aplica también a vectores de símbolos
  - A^n en 2^{nR} regiones P_i.
  - y^n cae en P_i
  - Como $1 \le i \le 2^{nR}$ => nR bits para representar nN símbolos se comprime si $R \le \log N$
  - Cada símbolo i a un $\hat{y}_i^n$

Función Tasa-Distorsión: Teorema R-D (IV)

Teorema R-D: La mínima tasa R para una fuente sin memoria que se quiera reconstruir con una distorsión menor que un umbral D (R_{min} tal que $E[d(Y^n, \hat{Y}^n)] \leq D$) viene dada por la función Tasa-Distorsión (*Rate-Distorsion*)

- **$R(D)=\min I(Y; \hat{Y})$, para todo $p(y|\hat{y}):E[d(y, \hat{y})]\leq D$**
  [Demostración en Cover]
  - $I(Y; \hat{Y})=H(Y)-H(Y|\hat{Y})$ ó $I(Y; \hat{Y})=h(Y)-h(Y|\hat{Y})$
  - $E[d(y, \hat{y})]=\Sigma \Sigma\ p(y,\hat{y})\ d(y, \hat{y}) = \Sigma \Sigma\ p(y)\ p(\hat{y}|y)\ d(y, \hat{y})$

- Igualmente despejando la distorsión: D(R), se puede ver el teorema como dada un R, cual es la mínima distorsión esperable.

Función Tasa-Distorsión: Teorema R-D (V)

Cada y^n en P_i se reconstruye con $\hat{y}_i^n$, de forma que la distorsión sea mínima para todo $y_i^n \in P_i$.

Si aumento R (más dinero, ancho de banda, …) o aumento n (más retardo) nR aumenta y por lo tanto 2^{nR}: número de regiones P_i => regiones más pequeñas, de forma que $d(y_i^n, \hat{y}_i^n)$ disminuye

R=0 => D máxima: existe un único valor de reconstrucción (e.g., el centroide de y^n)

D=0 => R máxima = log N (cada símbolo tiene un único valor de reconstrucción)

R(D) es para una fuente (fdp) y distancia (d(x,y))

- R(D) da el valor mínimo asintótico con la complejidad del codec
- Otra cosa es como obtenerlo…

Función Tasa-Distorsión: Ejemplo 1 (I)

Sea una v.a de una fuente DMS con distancia Hamming como medida de distorsión.

Para esa v.a. y distancia, se puede demostrar que

$$R(D)=\begin{cases} H_b(p)- H_b(D) & , 0\leq D\leq \min(p,1-p) \\ 0 & , \text{resto} \end{cases}$$

a) Si la fuente es binaria simétrica, calcular R para Pe≤0.25

b) Si la fuente es binaria simétrica y R=0.75, calcular Pe

Función Tasa-Distorsión: Ejemplo 1 (II)

$$R(D)=\begin{cases} H_b(p)- H_b(D) & , 0\leq D\leq \min(p,1-p) \\ 0 & , \text{resto} \end{cases}$$

$D=D_H=E[d_H(Y, \hat{Y})]=p(y\neq\hat{y}).1+p(y=\hat{y}).0 = p(y\neq\hat{y}) =Pe$

a) Si la fuente es binaria simétrica, calcular R para Pe≤0.25

- p=0,5, D=Pe=0.25
- D=0.25(<0.5)
- $R(0.25)=H_b(0.5)-H_b(0.25) \approx 0.189$ [bits/símbolo]
- $\Rightarrow$ Como R(D) da R mínima para distorsión menor que D, entonces Pe será igual a D y por tanto menor de 0.25

b) Si la fuente es binaria simétrica y R=0.75, calcular Pe

- $R(D)=0.75=H_b(0.5)-H_b(D)$
- $H_b(D)=0.25 \Rightarrow D\approx 0.042$(por tablas) (cumple <0.5) $\Rightarrow$ Pe≈0.042

Función Tasa-Distorsión: Ejemplo 1 (III)

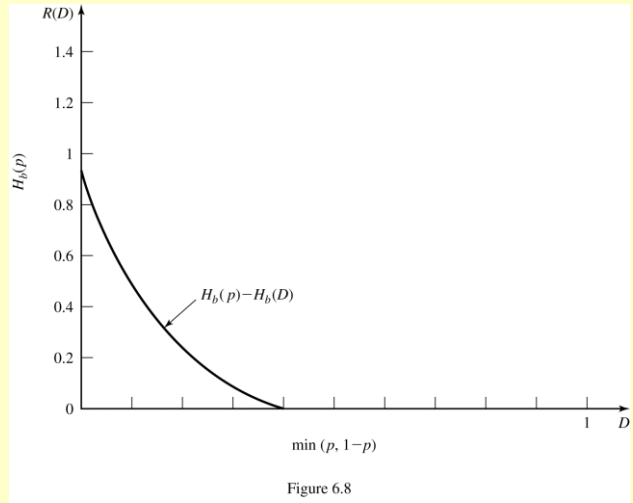


Figure 6.8
Rate-distortion function for a binary source with Hamming distortion.

Función Tasa-Distorsión: Ejemplo 2 (I)

Sea una v.a de una fuente gaussiana con distancia error cuadrático.

Para esa v.a. y distancia, se puede demostrar que

$$R(D)=\begin{cases} \frac{1}{2}\log(\sigma^2/D) & \text{si } 0\le D\le\sigma^2 \\ 0 & \text{resto} \end{cases}$$

Si me encuentro en $0\le D\le\sigma^2$ la distorsión es: $D(R)=\sigma^2 .2^{-2R}$

¿Qué ocurre cuando aumento en un bit R?

- $D'=D.2^{-2}=D/4$
- $D'=D-6dB \Rightarrow$ la distorsión disminuye en 6dB
- Sea $\sigma^2=1$ y 8 bits/símbolo:
  - $D=1.2^{-2.8}=1{,}52.10^{-5}$
- 16 bits/símbolo:
  - $D'=1.2^{-2.16}=1{,}52.10^{-5}/4^8=D-48$ dB

Función Tasa-Distorsión: Ejemplo 2 (II)

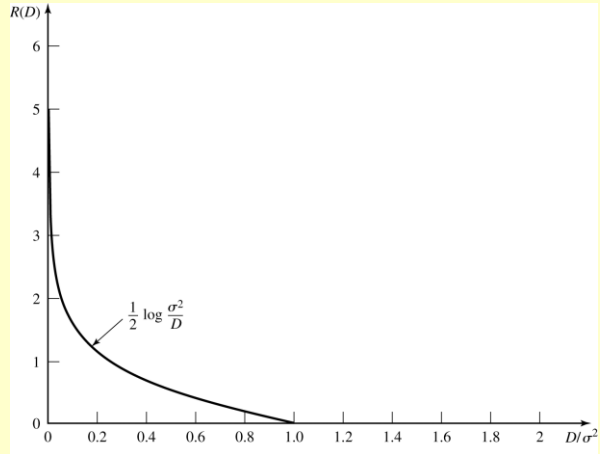


Figure 6.9
Rate-distortion function for a Gaussian source with squared-error distortion.

Índice

Índice

Cuantificación: Introducción

Como se ha comentado, con fuentes analógicas (muestreadas) es imposible la codificación sin pérdidas, pues las muestreas reales tiene que ser “discretizadas” en amplitud para poder ser procesadas digitalmente.

- Por lo tanto siempre hay distorsión

La teoría de cuantificadores permite diseñar esquemas para representar fuentes analógicas sin introducir una distorsión excesiva (para una fuente con determinada fdp y una distancia dada).

El límite de la función R-D sólo se alcanza asintóticamente

- Nunca se alcanza
- Según nos acercamos a él, la complejidad se dispara

Cuantificación: Tipos

Cuantificación Escalar: a cada muestra le asigno un valor (palabra/símbolo) de cuantificación
- regiones unidimensionales→lineales

Cuantificación Vectorial: Las muestras se agrupan en vectores de n muestras y a cada uno de estos vectores se le asigna un valor (palabra/símbolo) de cuantificación
- regiones n-dimensionales
- +eficiente,+complejo

Índice

Cuantificación escalar

Cada muestra (valor) de entrada se cuantifica (asocia) a un valor (símbolo/palabra/nivel) de cuantificación que pertenece al rango (finito) de los especificados por el cuantificador. Posteriormente se codifica el valor de cuantificación mediante una secuencia binaria (mediante una codificación sin pérdidas –generalmente estadística-)

Objetivo: minimizar D en el proceso de asignación (mapeo) del rango de los reales ($y \in P$) al de los valores discretos del cuantificador ($\hat{y}_k$)

La región del rango de entrada P se divide en regiones $\{P_k\}$ k=1..N tales que:

- $Ri \cap Rj = 0 \ \forall i \neq j$
- $\cup Ri = R$

Cuantificación: para cada $yi \in Ri$ se le asignada el $\hat{y}i$ asociado a Ri (que suele "descuantificarse" a un real)

Con N regiones de cuantificación son necesarios ceil(log (N)) *bits/símbolo* para representar los valores cuantificados

- $R_Q = \log(N)$ vs $R_F = \infty \Rightarrow$ el cuantificador comprime
  - $\text{Log}(N) \leq 2^{nbits}$ (lo ideal es =)
- $D = E[(Y-Q(Y))^2]$: Ruido de cuantificación cuadrático medio (se pueden usar otras distancias)
- D es poco relevante, generalmente se suele utilizar una medida mucho más relevante, la Relación Señal a Ruido: SNR: $E[Y^2]/E[(Y-Q(Y))^2]$
  - Relación Señal a Ruido de Cuantificación: SQNR: $E[Y^2]/E[Y-(Q(Y))^2]$

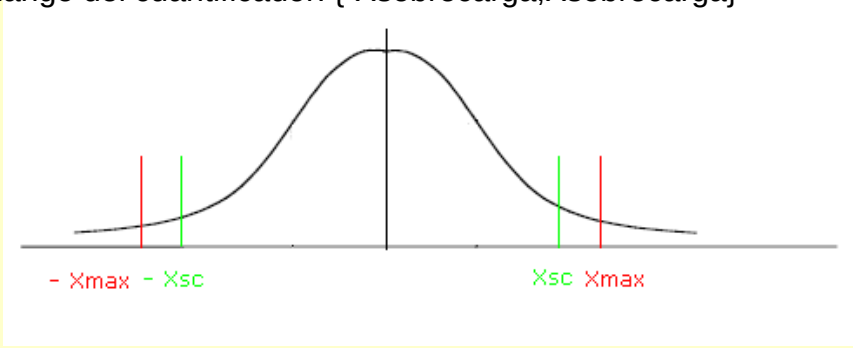
Cuantificación escalar: caracterización entrada

Dominio de entrada: continuo (números reales), discreto

Características de la señal de entrada: función densidad de probabilidad

Rango señal entrada: {-Xmax,Xmax}

Rango del cuantificador: {-Xsobrecarga,Xsobrecarga}



Cuantificación escalar: función de cuantificación

Función de cuantificación: Q(y)=ŷ

- No lineal => no invertible => pérdida de información

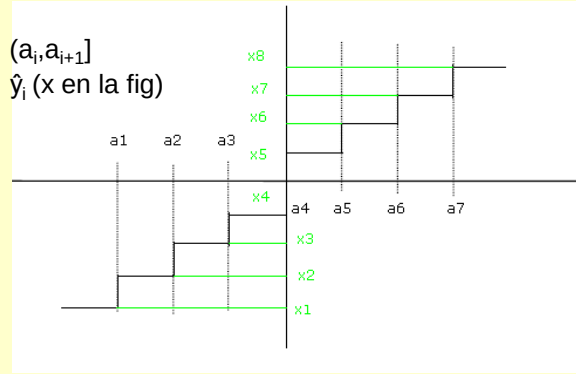
Niveles de decisión: a_i

Intervalo de cuantificación: $(a_i,a_{i+1}]$

Valores de reconstrucción: $\hat{y}_i$ (x en la fig)

Tipos:

- simétricos o no
- uniformes o no
- con corte central o no



El de la figura: cuantificador uniforme simétrico con corte central (N par)

- $\hat{y}_i$: i=1..N [pares]
- a_i: i=1..N-1 (a_i, i=0..N) [impares]

Cuantificación escalar: cuantificadores con/sin corte central

Cuantificador simétrico con corte central (N par)

- $\hat{y}_i$: i=1..N [pares]
- a_i: i=1..N-1 (a_i: i=0..N) [impares]

- $a_{(n/2)}=0$
- $\hat{y}_i = a_{i-1} + (a_i-a_{i-1})/2 = a_i - (a_i-a_{i-1})/2$
  - Suele ser lo común, pero en principio $\hat{y}_i \in (a_{i-1},a_i)$

Cuantificador simétrico sin corte central (N impar)

- $\hat{y}_i$: i=1..N [impares] (no uso al menos un símbolo : 2^n)
- a_i : i=1..N-1 (a_i: i=0..N) [pares]

- $\hat{y}_{ceil(N/2)} = 0$ ó $\hat{y}_{(N+1)/2} = 0$
- $\hat{y}_i = a_{i-1} + (a_i-a_{i-1})/2 = a_i - (a_i-a_{i-1})/2$
  - Suele ser lo común, pero en principio $\hat{y}_i \in (a_{i-1},a_i)$

Ejercicio de clase 13: Cuantificación escalar

Dibujar la función de cuantificación de un cuantificador simétrico con corte central (N=8)

Dibujar la función de cuantificación de un cuantificador simétrico sin corte central (N=7)

Comentar los resultados

Ejercicio de clase 13: Cuantificación escalar - solución

Cuantificación escalar: Distorsión – Ruido de cuantificación

$D = E[(Y-Q(Y))^2]$

$D=\int_{-\infty}^{a1} (y- \hat{y}_1)^2 f_y dy + \sum_{i=2}^{N-1} \int_{ai-1}^{ai} (y- \hat{y}_i)^2 f_y dy + \int_{a(N-1)}^{\infty} (y- \hat{y}_N)^2 f_y dy$

$D=\int_{-\infty}^{a0} (y- \hat{y}_1)^2 f_y dy + \sum_{i=1}^{N} \int_{ai-1}^{ai} (y- \hat{y}_i)^2 f_y dy + \int_{a(N)}^{\infty} (y- \hat{y}_N)^2 f_y dy$

$D=$ ruido sobrecarga (inf.)$+ \sum_{i=1}^{N}$ ruido granular + ruido sobrecarga (sup)

Generalmente

- $a_0=-V_{sc}$
- $a_N=V_{sc}$

SQNR: $S/D_Q = E[Y^2]/E[(Y-Q(Y))^2]$

Cuantificación escalar: ejemplo

Sea una fuente gaussiana estacionaria x(t) de media cero y densidad espectral
$S_x(f)$= {2,0} si {|f|<100 Hz, resto}

La fuente es muestreada a la tasa de Nyquist y cada muestra se cuantifica utilizando un cuantificador uniforme de 8 niveles con los siguientes valores de decisión y reconstrucción

- a_i={-60, -40, -20, 0, 20, 40, 60} , i=1..N-1
- x_i={-70, -50, -30, -10, 10, 30, 50, 70} , i=1..N

Hallar R y D (ruido de cuantificación).

Cuantificación escalar: ejemplo – solución (I)

Para calcular R:

- R=fs[muestras/seg].l[bits/muestra]
- Como tengo 8 niveles => 3 bits/símbolo-muestra
- Como B=100 HZ, fs (según Nyquist) ≥ 2B == fs = 200 muestras/seg
  - fs mínima (aunque cara)
- R = 200[muestras/seg].3[bits/muestra] = 600 bits/seg

Para calcular D:

- $D = E[(x-Q(x))^2] = \sum_i \int (x-Q(x))^2 f_x dx =$
  $= \int_{-\infty}^{a1} (x- x_1)^2 f_x dx + \sum_{i=2}^{N-1} \int_{ai-1}^{ai} (x- x_i)^2 f_x dx + \int_{a(N-1)}^{\infty} (x- x_N)^2 f_x dx$
- Cada muestra es una v.a gaussiana de media cero y varianza:

$$f_x(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-x^2/2\sigma^2}$$

$$\sigma^2 = E(X_i^2) = R_x(\tau)|_{\tau=0} = \int_{-\infty}^{\infty} S_x(x)df = \int_{-100}^{100} 2df = 400$$

Sustituyendo se obtiene D≈33.38

Si no transmitiese nada (R=0) y reconstruyese todo a 0 (valor medio), la distorsión es $D= \sigma^2 =400$

La distorsión mínima se obtiene substituyendo R=3 y σ=20 en
$R= ½ \log (\sigma^2 /D)$ (función R-D para fuente gaussiana) y es $D_{min}=6.25$

Cuantificación escalar: ejemplo – solución (II)

Por lo tanto se está lejos del límite R-D para R=600 bps (en R(D) R=3!!!! D≈33.38) que es D=6,25

- La cuantificación uniforme con 3 bits es algo muy simple
- Para acercarnos al límite son necesarias técnicas más complicadas
  - Cuantificación vectorial, compresión mediante transformación, …
- Además cada símbolo de salida no es equiprobable, ya que la fdp es gaussiana: se podría usar un código de longitud variable mediante codificación estadística
  - $P(\hat{y}_i) = \int_{ai-1}^{ai} f_x dx \quad 2 \le i \le N-1$
  - $P(\hat{y}_1) = \int_{-\infty}^{a1} f_x dx$
  - $P(\hat{y}_N) = \int_{aN-1}^{\infty} f_x dx$
  - $P(\hat{y}_1)= P(\hat{y}_N)=0.0014$, $P(\hat{y}_2)= P(\hat{y}_7)=0.0214$, $P(\hat{y}_3)= P(\hat{y}_6)=0.1359$, $P(\hat{y}_4)= P(\hat{y}_5)=0.3414$
  - $H(\hat{Y})=2.105$

Cuantificación escalar: ejemplo – solución (III)

Por lo tanto es más justo comparar D=33,38 con la correspondiente a la mínima que se puede lograr para esa fuente sin pérdidas D(R=H=2.105)=21,61

- Ahora ya hay menos diferencia

Pero se puede mejorar … intuitivamente

- La fdp es gaussiana, por lo que el uso de un cuantificador simétrico es correcto
- El símbolo más probable es el cero, y no es valor de reconstrucción por lo que se añade mucho ruido granular:
  - usar cuantificador sin corte central (valor de reconstrucción en cero)
- La fdp no es uniforme, hay más valores alrededor del origen:
  - Hacer intervalos de cuantificación más pequeños (menor error granular) alrededor del origen => cuantificador no uniforme

Índice

Cuantificación uniforme (I)

Los cuantificadores uniformes son los más sencillos

- todas las regiones (menos los extremos si existe sobrecarga) tienen el mismo tamaño Δ (intervalo de cuantificación)
- $\Delta = a_{i+1} - a_i \quad 1 \le i \le N-1 \ (0 \le i \le N)$
- $D = \int_{-\infty}^{a1} (y - \hat{y}_1)^2 f_y dy + \sum_{i=2}^{N-1} \int_{ai-1}^{ai} (y - \hat{y}_i)^2 f_y dy + \int_{a(N-1)}^{\infty} (y - \hat{y}_N)^2 f_y dy$
- $D = \int_{-\infty}^{a0} (y - \hat{y}_1)^2 f_y dy + \sum_{i=1}^{N} \int_{ai-1}^{ai} (y - \hat{y}_i)^2 f_y dy + \int_{a(N)}^{\infty} (y - \hat{y}_N)^2 f_y dy$

implica (cambio de variable por homogeneizar)

- $D = \int_{-\infty}^{a0} (x - x_1)^2 f_x dx + \sum_{i=1}^{N} \int_{a0+(i-1)\Delta}^{a0+i\Delta} (x - x_i)^2 f_x dx + \int_{a0+N\Delta}^{\infty} (x - x_N)^2 f_x dx$
- $D = \int_{-\infty}^{a1} (x - x_1)^2 f_x dx + \sum_{i=2}^{N-1} \int_{a1+(i-2)\Delta}^{a1+(i-1)\Delta} (x - x_i)^2 f_x dx + \int_{a1+(N-2)\Delta}^{\infty} (x - x_N)^2 f_x dx$

Por lo tanto existen N+2 variables a calcular que minimicen D:

- Δ
- a_0 (o cualquier a_i ya que se relacionan mediante el valor de Δ)
- N valores de x_i

Cuantificación uniforme (II)

Para calcular las N+2 variables se hace uso generalmente de técnicas de análisis numérico para resolver el sistema de ecuaciones

- $\partial D/\partial x_i' = 0$ (i=1..N); $\partial D/\partial \Delta = 0$; $\partial D/\partial a_1 = 0$

Antes de resolver este sistema podemos simplificarlo si la fdp es par (simétrica respecto al origen)

- Si N es par: 2+(N/2) variables
  - $a_i = -a_{N-i} = -(N/2 - i)\,\Delta$ $(i \le i \le N/2)$, $a_{N/2} = 0$,
  - $x_i = -x_{N+1-i}$ $(i \le i \le N/2)$
- Si N es impar: 2+(N+1)/2 variables
  - $a_i = -a_{N-i} = (-N/2 + i)\,\Delta$ $(i \le i \le (N-1)/2)$
  - $x_i = -x_{N+1-i}$ $(i \le i \le (N+1)/2)$, $x_{(N+1)/2} = 0$

Incluso con las simplificaciones la minimización es tediosa y se hace mediante análisis numérico

- Generalmente para las fdp comunes existen tablas para los cuantificadores uniformes
  - ver Proakis para algunos ejemplos de tablas (e.g Tabla 6.2)

Cuantificación uniforme: ejercicio propuesto 7

Calcular la fórmula de la Distorsión de un cuantificador uniforme asumiendo fdp par (simétrica respecto al origen) para N par e impar.

Índice

Cuantificación no uniforme

Si se relaja la condición de que los intervalos de cuantificación sean de igual tamaño, la minimización de la distorsión tiene menos restricciones y por lo tanto el cuantificador será mejor (menor D para el mismo número de niveles – símbolos).

- $D=\int_{-\infty}^{a1} (y- \hat{y}_1)^2 f_y dy + \sum_{i=2}^{N-1} \int_{ai-1}^{ai} (y- \hat{y}_i)^2 f_y dy + \int_{a(N-1)}^{\infty} (y- \hat{y}_N)^2 f_y dy$
- $D=\int_{-\infty}^{a0} (y- \hat{y}_1)^2 f_y dy + \sum_{i=1}^{N} \int_{ai-1}^{ai} (y- \hat{y}_i)^2 f_y dy + \int_{a(N)}^{\infty} (y- \hat{y}_N)^2 f_y dy$

El número de variables a calcular mediante la minimización de D será 2N-1:

- N valores de reconstrucción (x_i) y N-1 valores de decisión (a_i)
- La mitad si fdp par (simétrica)

Cuantificador no uniforme óptimo: Max-Lloyd

Para calcular el cuantificador escalar uniforme óptimo hay que resolver el sistema de 2N – 1 variables

- $\partial D/\partial a_i = 0$
- $\partial D/\partial x_i = 0$

Este sistema de ecuaciones da lugar a las dos **condiciones de Max-Lloyd** para obtener cuantificadores óptimos para una fdp

1. Las fronteras de los intervalos de cuantificación (los valores de decisión) son los puntos medios de los valores de reconstrucción (ley del vecino más próximo)
2. Los valores de reconstrucción son los centroides de los intervalos de cuantificación

$$\hat{x}_i = E\left[x \mid x \in [a_{i-1}, a_i)\right] = \frac{\int_{a_{i-1}}^{a_i} x \; fdp(x)\, dx}{\int_{a_{i-1}}^{a_i} fdp(x)\, dx}$$

Cuantificador no uniforme óptimo: Max-Lloyd

Aunque son reglas sencillas, no son reglas para soluciones analíticas. Se suele empezar con unos intervalos de cuantificación fijos, a continuación se usa la segunda condición para encontrar valores de reconstrucción y luego la primera para obtener los nuevos intervalos de cuantificación. Se sigue iterativamente hasta que $|D_j - D_{j+1}|$ es menor que una cota prefijada.

- Generalmente para las fdp comunes existen tablas para los cuantificadores no uniformes
  - ver Proakis para algunos ejemplos de tablas (e.g Tabla 6.3)

Índice

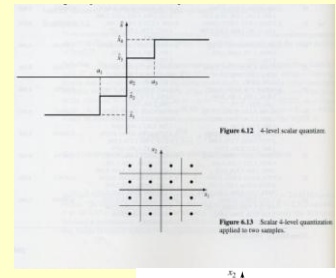
Cuantificación vectorial

Se basa en cuantificar grupos de muestras en lugar de muestras "sueltas".

Si agrupo muestras sueltas, puedo aplicar a cada muestra un cuantificador escalar lo que da lugar a regiones rectangulares (pero no mejora generalmente).

La idea de la cuantificación vectorial es tomar bloques de n muestras y diseñar el cuantificador en el espacio n-dimensional. En función del número de regiones K (no limitado a NxM) la tasa será log(K)/n

Para calcular las regiones de cuantificación y los valores de reconstrucción se usa el mismo procedimiento descrito para cuantificadores escalares óptimos (Max-Llloyd) extendido a regiones en el espacio n-dimensional.



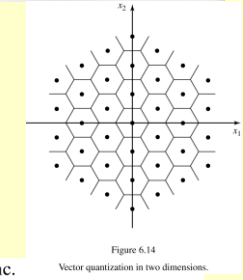


Figure 6.14
Vector quantization in two dimensions.

Índice

Índice

- *Introducción*
- *Fundamentos*
- *Algoritmos de codificación de fuente sin pérdidas*
- *Teoría Tasa-Distorsión*
- *Cuantificación*
- Codificadores
  - **Introducción**
  - Codificación por forma de onda
  - Codificación análisis-síntesis
  - Codificación transformacional

Codificadores: Introducción

Los codificadores están compuestos generalmente por tres etapas:

- Cuantificación (A/D o M->N)
- Compresión (Codificación con pérdidas): suele incluir una cuantificación adicional a la de entrada
- Codificación (Codificación sin pérdidas)

siendo la etapa de compresión la que puede faltar (aplicaciones en las que no se toleran pérdidas de información).

Adicionalmente a su clasificación según la etapa de cuantificación (e.g., escalares, vectoriales, uniformes, no uniformes) y codificación (e.g., estadísticos, universales), se suelen agrupar por el método utilizado para comprimir:

- Codificadores de Forma de Onda
- Codificadores de Análisis-Síntesis
- Codificadores Transformacionales

Codificadores: Tipos

Codificadores de Forma de Onda

- Se basan en las propiedades (características) de la onda: ancho de banda, densidad espectral de potencia, función densidad de probabilidad, … Son genéricos y robustos para cualquier tipo de onda de características similares.

Codificadores de Análisis-Síntesis

- Se basan en un modelo de síntesis de la señal en el receptor, a partir de de una serie de parámetros que se transmiten. Estos parámetros del modelo se obtienen mediante análisis en el emisor. Se usan generalmente para voz, cuyo modelo de generación está muy estudiado (el tracto vocal es un filtro que modula una señal ruidosa o tonal emitida por las cuerdas vocales).

Codificadores Transformacionales

- Se basan en trabajar en un dominio transformado de la fuente. Generalmente se incluyen aquí los codificadores predictivos (codificar la diferencia entre muestras en lugar de las muestras), aunque los más utilizados son los que (además) trabajan en el dominio “frecuencial” (e.g., DCT, FFT)

Índice

Codificación por forma de onda

El diseño de los codificadores por forma de onda se plantea como objetivo reproducir la forma de onda de la forma más fielmente posible.

Son robustos y generalistas, siempre que las fuentes tengan características similares

- B, fdp, energía, …

La técnica establecida (más antigua y sencilla) se denomina Modulación por Impulsos Codificados –MIC- (Pulse Code Modulation –PCM-) y se basa en:

- Muestrear la señal analógica por encima de la frecuencia de Nyquist.
  - Con un filtro de premuestreo previo (W ≈ 2*Bx(t))
    Es peor la distorsión armónica que eliminar las altas frecuencias
- Cuantificar con un número $N=2^{nbits}$ de niveles.
- Codificar mediante un código de longitud fija de n bits.

existiendo versión uniforme y no uniforme.

La codificación por forma de onda es básicamente un cuantificador.

PCM uniforme (I)

Se diseña para fuente uniforme: $x_n \in [-x_{max}, x_{max}]$

- Por lo tanto solamente son óptimos para ese tipo de fuente
- Se asume $V_{sc} = x_{max}$ (Si $|x_{max}|>|V_{sc}|$, habrá que codificar al extremo, si $|x_{max}|<|V_{sc}|$ se desaprovecha parte del Q)

Intervalo de cuantificación:

- $\Delta = 2V_{sc}/N = 2V_{sc}/2^{nbits} = V_{sc}/2^{nbits-1}$

Valores de decisión:

- $a_i = -V_{sc} + i\,\Delta$ i=0..N (i=1..N-1)

Valores de reconstrucción:

- $x_i' = -V_{sc} + (i-½)\Delta$ i=1..N
- $x_i' \Rightarrow x \in [a_{i-1}, a_i)$ {o (..]}

Índice de cuantificación (IC: i=0..N vs. i=1..N-1) :

- $x \Rightarrow IC \Leftrightarrow x \in [a_i, a_{i+1}) \Rightarrow -V_{sc} + i\Delta \le x < -V_{sc} + (i+1)\Delta \Rightarrow IC=floor[(x+ V_{sc})/\Delta]$
  - ¡¡¡ IC=0..N-1!!! Es correcto, pues IC=1..N son N valores, que para codificar con longitud fija requiere nbits si el rango es 0..N-1
  - $x==a_N \Rightarrow x'=x_N$

Error de cuantificación = x-Q(x) = v.a uniforme en [- Δ/2, Δ/2) {o (..]}

- Si N aumenta => Δ disminuye
- Si V_{sc} disminuye => Δ disminuye

PCM uniforme (II)

$$\tilde{X} = X - Q(X)$$

Distorsión

$$D = E[\tilde{X}^2] = \int_{\frac{-\Delta}{2}}^{\frac{\Delta}{2}} \frac{1}{\Delta}\tilde{x}^2 d\tilde{x} = \frac{\Delta^2}{12} = \frac{V_{sc}^2}{3N^2}$$

Relación Señal a Ruido

$$SQNR = \frac{E\left[X^2\right]}{E\left[\tilde{X}^2\right]} = \frac{\bar{X}^2}{V_{sc}^2} \cdot 3N^2 = \bar{X}_n^2 \cdot 3N^2 = \bar{X}_n^2 \cdot 3 \cdot 4^{nbits}$$

- $3N^2$: límite superior de SQNR

Si el valor medio normalizado disminuye => disminuye SQNR

SQNR(dB)=Px_n (dB)+6 nbits (dB)+4.8 (dB)

- Si aumento 1 bit, mejora de 6 dB (como ya se vio en el caso general)

Ejercicio de clase 14: PCM uniforme

Sea un PCM uniforme con 8 bits y V_{sc}=1,5 voltios. Codificar y decodificar una muestra V=0,7505 voltios.

Ejercicio de clase 14: PCM uniforme – solución

Sea un PCM uniforme con 8 bits y Vsc=1,5 voltios. Codificar y decodificar una muestra V=0,7505 voltios.

PCM no uniforme (I)

Si la fuente es aproximadamente uniforme se puede hacer uso de PCM uniforme, pues la desviación es aceptable.

Las señales naturales tienden a ser poco uniformes

- E.g., en voz lo normal es que existan más valores de pequeña amplitud y pocos de gran amplitud
  - Parece interesante reducir el error en los más frecuentes => Δ pequeños alrededor del origen
  - Cuantificación no uniforme
- Generalmente en voz se hace uso de PCM no uniforme con el siguiente procedimiento
  - Se lleva a cabo una compresión del rango
  - Se cuantifica uniformemente
  - Se lleva a cabo una expansión del rango
- Que equivale a una cuantificación no uniforme, ya que se reduce (o se amplia) el rango de los valores grandes (o pequeños)

PCM no uniforme (II)

Compansión: COMPResión-expANSIÓN
$\{x_n\}$ -> compresión [g(x)] -> Q-PCM uniforme (nbits) -> …
… R-PCM uniforme (nbits) -> expansión [g^{-1}(x)] -> $\{x'_n\}$

Existen dos tipos de "compansores" muy utilizados en señales vocales

- Ley μ (USA, Canada; μ=255, nbits=7)

$$g(x) = \frac{\log(1+\mu|x|)}{\log(1+\mu)}\operatorname{sgn}(x)$$

- Ley A (Europa; A=87.56, nbits=8)

$$g(x) = \frac{1+\log(A|x|)}{1+\log(A)}\operatorname{sgn}(x)$$

No son sistemas de D_{min}, sino sistemas de S/D constante (cuantificación uniforme robusta)

PCM no uniforme: Rec. UIT-T Rec. G.711 (I)

La Recomendación de la UIT-T G.711 define el estándar de codificación PCM para frecuencias vocales (telefonía)

- f_s = 8 kHz
  - Tras filtrado paso bajo con fc=3400 Hz (atenuación > 25 dB para f_c > 4600 Hz)
- Cuantificación no uniforme robusta: aproximación ley A
- Codificación a 8bits/muestra
  - Los bits pares se invierten (codificación de canal) para evitar largas secuencias de ceros con canal silencioso
- 64 kbits/seg (y sus jerarquías TDM –e.g, 24x64-)

PCM no uniforme: Rec. UIT-T Rec. G.711 (II)

Cuantificación G.711: aproximación rectilínea a ley-A

- A=87,6 –aprox 87.56-

$$g(x)=\begin{cases}\dfrac{1+\ln A|x|}{1+\ln A}\operatorname{sgn}(x) & \dfrac{1}{A}\le|x|\le A\\ \dfrac{Ax}{a+\ln A} & 0\le|x|\le\dfrac{1}{A}\end{cases}$$

Se normaliza el rango dinámico al valor de sobrecarga

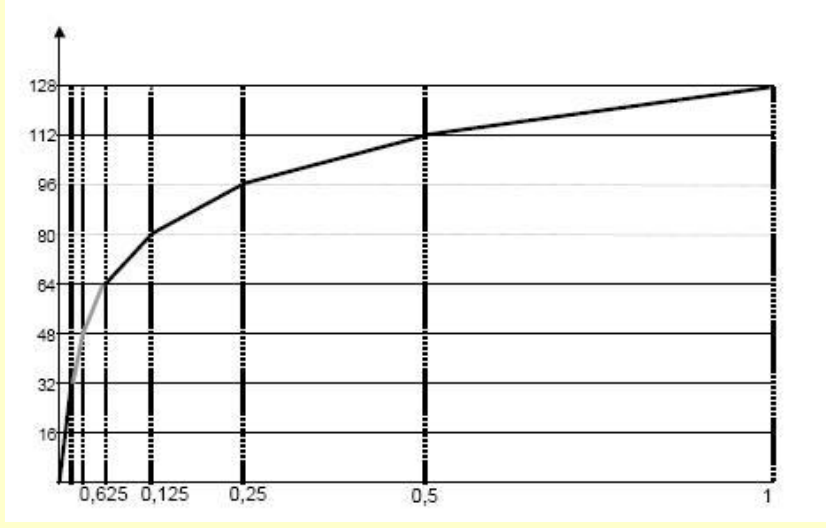
- V [voltios] => V/V_{sc} [UTN –Unidad de Tensión Normalizada-]

Se divide el rango dinámico en 13 segmentos simétricos

- (6+1/2)*2 [cada uno la mitad del anterior salvo el origen]

Cada segmento se divide en 16 intervalos de cuantificación (IC) uniformes

PCM no uniforme: Rec. UIT-T Rec. G.711 (III)



PCM no uniforme: Rec. UIT-T Rec. G.711 (IV)

Se reparten los 8 bits de cuantificación en:

- 1 para el signo
- 3 para los segmentos
  - 8 segmentos: 0..7
- 4 para la cuantificación uniforme
  - 16 ICs por segmento
  - Cada segmento un Δ distinto

Segmento	7	6	5	4	3	2	1	0
Ancho (UTN)	1/2	1/4	1/8	1/16	1/32	1/64	1/128	1/128
Δ_i (UTN)	½*1/16	1/4*1/16	1/8*1/16	1/16*1/16	1/32*1/16	1/64*1/16	1/128*1/16	1/128*1/16

- Se puede tratar como un cuantificador uniforme de 12 bits (1+11)
  - S0==S1= $1/2^7 * 1/2^4=1/2^{11}$

PCM no uniforme: Rec. UIT-T Rec. G.711 (V)

Cuantificación (cuenta de la vieja)

- Normalizar a UTN
- Identificar y codificar signo (positivo 1, negativo 0)
- Valor absoluto
- Identificar y codificar segmento (3 bits => 8 segmentos [S0==S1])
- Identificar y codificar intervalo dentro del segmento (4 bits => 16 IC)
  - Resto = V(UTN)-ai(UTN), siendo ai el extremo inferior del segmento

Reconstrucción (cuenta de la vieja)

- Identificar segmento y sumar extremo inferior del segmento
- Identificar IC y sumar: (IC+1/2) Δ_i
- Identificar y poner signo
- Desnormalizar

Segmento	7	6	5	4	3	2	1	0
Ancho (UTN)	1/2	1/4	1/8	1/16	1/32	1/64	1/128	1/128
Δ_i (UTN)	½*1/16	1/4*1/16	1/8*1/16	1/16*1/16	1/32*1/16	1/64*1/16	1/128*1/16	1/128*1/16

Ejercicio de clase 15: PCM no uniforme: Rec. UIT-T Rec. G.711

Sea un G.711 con Vsc=1,5 voltios

Codificar y decodificar una muestra V=0,7505 voltios

Comparar con PCM uniforme

Ejercicio de clase 15: PCM no uniforme: Rec. UIT-T Rec. G.711 – solución (I)

Sea un G.711 con Vsc=1,5 voltios

Codificar una muestra V=0,7505 voltios

Ejercicio de clase 15: PCM no uniforme: Rec. UIT-T Rec. G.711 – solución (II)

Sea un G.711 con Vsc=1,5 voltios

Decodificar

Ejercicio de clase 15: PCM no uniforme: Rec. UIT-T Rec. G.711 – solución (III)

Sea un G.711 con Vsc=1,5 voltios. Codificar y decodificar una muestra V=0,7505 voltios. Comparar con PCM uniforme

PCM no uniforme: Rec. UIT-T Rec. G.711 - ejercicio propuesto 8

Sea un cuantificador de 8 bits por muestra con valor de sobrecarga 1,5 voltios. Para las siguientes muestras (V1=1,2 voltios; V2=0,3 voltios; V3 = 0,05 voltios) calcular

- Código binario
- Valor de reconstrucción
- Error de reconstrucción

Para cuantificación PCM uniforme y G.711.

Compare razonadamente los resultados.

Índice

Codificación análisis-síntesis

En lugar de codificar la forma de onda se codifica un conjunto de parámetros de un modelo de generación de la señal a codificar.

Lleva asociada dos tipos de pérdidas:

- Pérdidas por imperfecciones del modelo (síntesis/análisis –complejidad-)
- Pérdidas por codificación con pérdidas de los parámetros del modelo

Se usan principalmente para señales vocales (Codificación Linear Predictiva), pues el modelo de generación de voz está bien estudiado y es aceptablemente bueno.

Codificación análisis-síntesis: LPC

LPC: Linear Predictive Coding (Codificación Linear Predictiva)

- Para voz/habla

Excitación del tracto vocal (filtro variable en tiempo) por vibraciones de las cuerdas vocales: ruido –sonidos sordos- versus tono a frecuencia fundamental (pitch) –sonidos sonoros-

$$x_n = \sum_{i=1}^{p} a_i x_{n-i} + G w_n$$

- Filtro todo polos: a_i i=1..p (p polos)
- Excitación w_n
- Ganancia G (volumen)
- Voz estacionaria en t (20-30 ms)
- Codificación
  - Sordo/sonoro 1 bit
  - Frecuencia fundamental 6 bits
  - Ganancia 5 bits
  - Cada a_i de 8 a 10 bits
- Compresión LPC 4.800 bps (cuantificaciónn vectorial, telefonía móvil)
- Compresión PCM 56.000 bps (8*8000 = 64.000 -> cod estadística, telefonía fija)

Índice

Codificación transformacional

Se basan en trabajar en un dominio transformado de la fuente.

- Las técnicas predictivas se basan en calcular una predicción de la muestra y codificar la diferencia entre la muestra real y su predicción. Si la predicción es buena casi todas las muestras tendrán valores alrededor del cero (fdp fuertemente no uniforme)
- Las técnicas en el dominio frecuencial tienen en común el llevar a cabo en primer lugar una transformación (e.g., DCT, FFT) y llevar a cabo las etapas de codificador (predicción, cuantificación, codificación estadística) sobre las muestras en ese dominio en lugar del dominio natural (píxel o muestra)
- Las técnicas denominadas híbridas constan de una primera etapa de predicción y una de transformación sobre la señal diferencia.

Codificación transformacional: PCM diferencial (I)

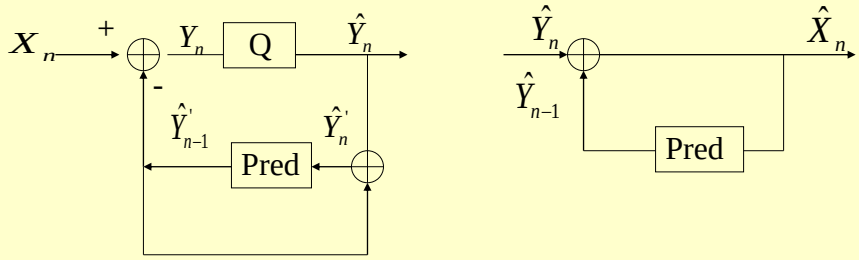
Generalmente las muestras contiguas de una señal natural están correladas (ni el sonido ni las imágenes tienen, en general, variaciones bruscas)

Por lo tanto se codifica la diferencia entre la muestra original y una predicción de la misma

- La predicción se hace a partir de P muestras anteriores, siendo P el orden del predictor
- $\hat{y}'_n = \Sigma_{i=1}^{P} a_i \hat{y}'_{n-i}$
- Si la predicción es buena (se aprovecha la correlación) la diferencia a codificar será pequeña y generalmente con fdp Laplaciana.
- En el caso de imágenes (2D) la predicción se hace sobre píxeles anteriores "cercanos"
  - Por ejemplo (p=4): (i-1,j-1), (i-1,j), (i-1,j+1), (i,j-1)
- IMPORTANTE: Como en cualquier decodificador la predicción se tiene que hacer sobre las muestreas decodificadas, para que el codificador y decodificador estén sincronizados (usen las mismas muestras). Si no fuese así, se acumularía un error adicional al de cuantificación que haría que el sistema no funcionase.

Codificación transformacional: PCM diferencial (II)

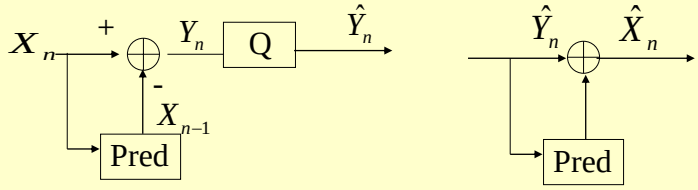
Esquema correcto



$$\hat{Y}'_{n-1} = \sum_{i=1}^{P} a_i \cdot \hat{Y}'_{n-i-1}$$

Codificación transformacional: PCM diferencial (III)

Esquema incorrecto



$$x'_{n-1} = \sum_{i=1}^{P} a_i \cdot x_{n-i-1}$$

$$\hat{Y}_{n-1} = \sum_{i=1}^{P} a_i \cdot \hat{Y}_{n-i-1}$$

Codificación transformacional: PCM diferencial: ejercicio propuesto 9

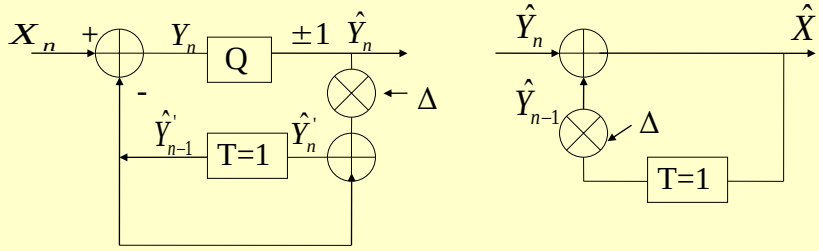
Demostrar que de los dos esquemas anteriores el primero es el correcto y el segundo el incorrecto.

Codificación transformacional: Modulación Delta (I)

Es una versión simplificada de DPCM que se basa en cuantificar con 1 bit, lo que implica que se sube o se baja un valor Δ para "ir siguiendo" a la señal original.

Δ es el parámetro fundamental

- Si Δ grande => ruido granular grande en zonas "planas"
- Si Δ pequeño => distorsión de sobrecarga en las pendientes "fuertes"
- Opción Δ adaptativa



- Se puede hacer una implementación sencilla mediante acumuladores

Codificación transformacional: Modulación Delta (II)

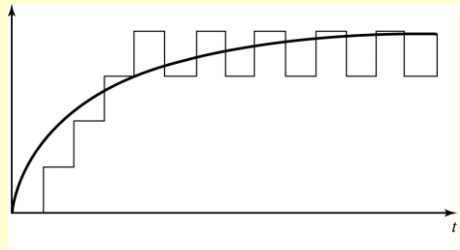


Figure 6.24
Large Δ and granular noise.

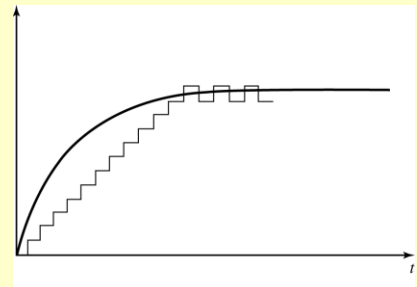
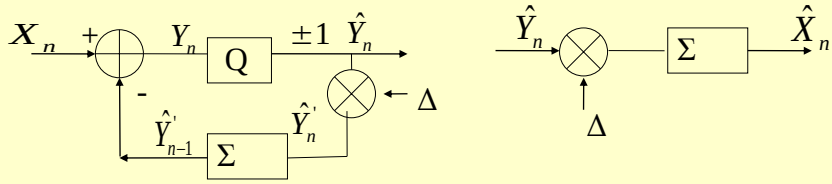


Figure 6.25
Small Δ and slope-overload distortion.

Codificación transformacional: Modulación Delta (III)



Codificación transformacional: codificación en dominios transformados (I)

En lo visto hasta el momento se trabaja en el dominio “natural” de la señal (muestras en el tiempo y espacio).

Aunque los codificadores predictivos se suelen incluir dentro de los transformacionales (principalmente porque tienen un esquema “idéntico”), cuando se habla de codificación transformacional generalmente se entiende que se trabaja en el dominio transformado “frecuencial”.

- Las pérdidas suelen aparecer por la cuantificación de los coeficientes transformados: si no hubiese cuantificación, la etapa de compresión transformacional sería sin pérdidas.

Codificación transformacional: codificación en dominios transformados (II)

Las codificaciones transformacionales más usadas son:

- Descomposición en subbandas: el rango de frecuencias de la señal se divide en distintas bandas mediante un banco de filtros.
  - Cada subbanda se codifica de forma independiente.
  - La salida de cada filtro se submuestrea de forma que el número de muestras se mantiene constante (aunque suelen ser reales en lugar de enteros –la entrada son muestras cuantificadas-)
  - Un tipo de "subbandas" (realmente son descomposiciones en el dominio natural con escalado en tiempo) muy de actualidad son las wavelets (ondículas) que se usan principalmente en codificadores escalables (e.g., JPEG2000)
- Transformación lineal por bloques: La señal de entrada se divide en bloques de muestras no solapadas y se le aplica una transformación lineal.
  - La transformación lineal concentra la energía (fdp no uniforme) y permite descartar muestras (codificación con pérdidas)
  - La transformación óptima es la transformación de Karhunen-Loewe (los coeficientes resultantes tienen fdp que son gaussianas incorreladas)
  - Generalmente se usa la DFT (Discrete Fourier Transform) –coeficientes complejos reales- o la DCT (Discrete Cosine Transform) –coeficientes reales-

Codificación transformacional: codificación en dominios transformados (III)

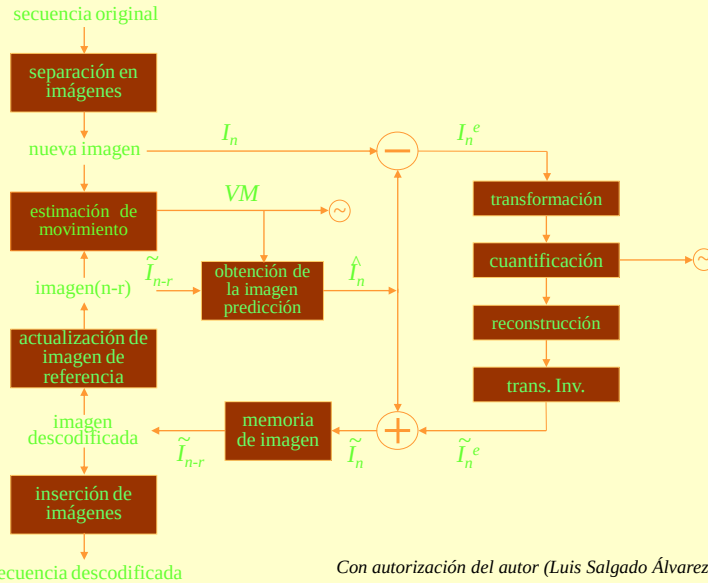
Tras aplicar la transformación los coeficientes son cuantificados y codificados mediante un codificador estadístico.

En decodificación se aplica el proceso inverso.

Se pueden aplicar en esquemas híbridos:

- Predicción previa a la transformación (generalmente lineal por bloques)
  - Codificadores de vídeo (predicción con compensación de movimiento)

Codificación transformacional: codificación en dominios transformados (IV)



Con autorización del autor (Luis Salgado Álvarez de Sotomayor, UPM)

Codificación transformacional: JPEG (I)

El estándar JPEG (1991) proporciona las especificaciones para lograr la compresión de imágenes fijas con un gran margen de calidad y niveles de compresión.

Características:

- Resolución arbitraria de la imagen original
- Varios modelos de color y formatos de muestreo
- Algoritmos de compresión sin pérdidas y con pérdidas
  - Modos de codificación: secuencial (*baseline*), progresivo, secuencial sin pérdidas, jerárquico
- La calidad depende del contenido de la imagen y de la calidad deseada
  - 0,15 bpp (reconocible)
  - 0.25 – 0.5 bpp (útil_ moderada a buena)
  - 0.5 - 0.75 bpp (buena a muy buena)
  - 0.75 - 1.50 bpp (excelente)
  - 1.50 – 2.00 bpp (indistinguible)

Codificación transformacional: JPEG (II)

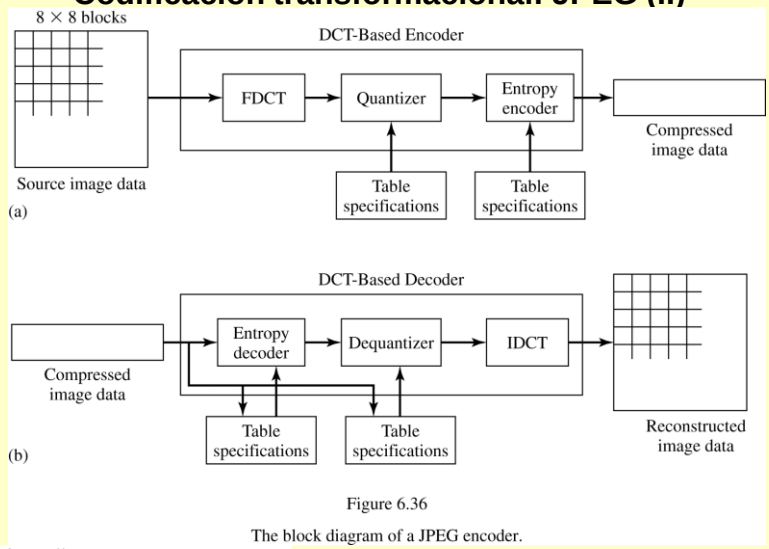


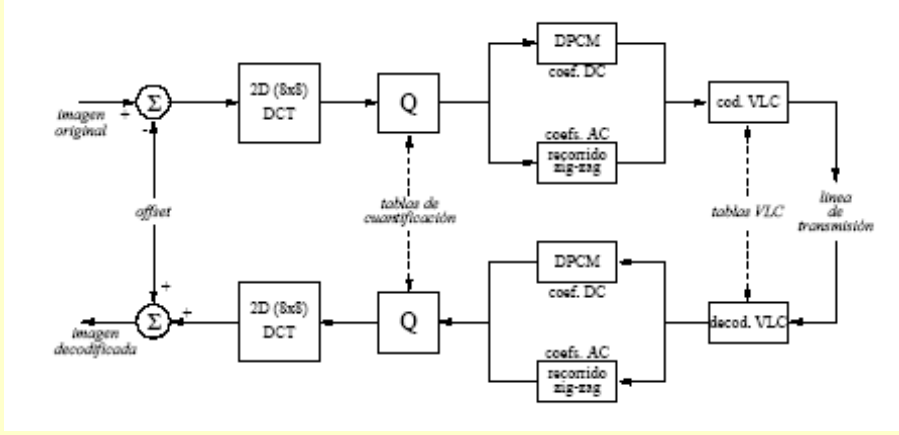
Figure 6.36

The block diagram of a JPEG encoder.

John G.Proakis / Masoud Salehi
Communication Systems Engineering, 2nd. Edition

Codificación transformacional: JPEG baseline (I)

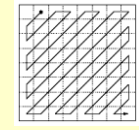
JPEG modo secuencial basado en DCT (baseline system)



Con autorización del autor (Luis Salgado Álvarez de Sotomayor, UPM)

Codificación transformacional: JPEG baseline (II)

- Offset: su valor es $2^{nbits-1}$. Permite que los coeficientes DC de la DCT tengan un valor medio de 0, sin afectar a los coeficientes AC
- DCT: se calcula la DCT de bloques de 8x8 píxeles
- Q: plantea una cuantificación sencilla mediante tablas de cuantificación (matriz de visibilidad), que aplica un cuantificador uniforme distinto a cada uno de los 64 coeficientes (distintas para luminancia y color). La cuantificación se lleva a cabo dividiendo en el coeficientes de la matriz de visibilidad (Δ del cuantificador uniforme, sin valor de reconstrucción en el cero) y redondeando al entero más cercano.
- DPCM: los coeficientes DC se codifican mediante DPCM
  - $Diff=DC_i-DC_{i-1}$
- Zig-zag: los coeficientes AC se reordenan
- VLC: codificación entrópica mediante tablas Huffman predefinidas
  - Unas para DC y otras para AC (tras extensión RLE)



Codificación transformacional: JPEG baseline (III)

- En decodificación es necesario conocer las tablas utilizadas (matriz de visibilidad y tablas Huffman), por lo que hay que transmitir en el bitstream esa sobrecarga
  - Generalmente tablas predefinidas y la sobrecarga se minimiza
- La reconstrucción (“descuantificación”) se hace multiplicando por el coeficiente de la matriz de visibilidad

Codificación transformacional: JPEG baseline (IV)

Codificación entrópica coeficientes AC

- La reordenación en zig-zag ordena por orden frecuencial creciente y debido a la matriz de visibilidad (basada en el sistema visual humano) se logra que tras el zig-zag aparezcan muchas secuencias de ceros.
- A cada secuencia de ACs se le aplica una extensión RLE que genera ternas (Z, L, V)
  - Z: número de ceros previos al valor no nulo
  - L: es la categoría que indica el número de bits para representar el valor no nulo
  - V: valor no nulo
- Al final se añade un símbolo EOB (End of Block)
- (Z,L) y EOB se codifican mediante un Huffman predefinido
- V de forma independiente

Codificación transformacional: JPEG baseline (V)

Zero run length	Amplitude Category	Code length	Codeword
0	1	2	00
0	2	2	01
0	3	3	100
0	4	4	1011
0	5	5	11010
0	6	6	111000
0	7	7	1111000
1	1	4	1100
1	2	6	111001
1	3	7	1111001
1	4	9	111110110
2	1	5	11011
2	2	8	11111000
3	1	6	111010
3	2	9	111110111
4	1	6	111011
5	1	7	1111010
6	1	7	1111011
7	1	8	11111001
8	1	8	11111010
9	1	9	111111000
10	1	9	111111001
End of Block (EOB)		4	1010

Codificación transformacional: JPEG baseline (VI)

U =

87	89	101	106	118	130	142	155
85	91	101	105	116	129	135	149
86	92	96	105	112	128	131	144
92	88	102	101	116	129	135	147
88	94	94	98	113	122	130	139
88	95	98	97	113	119	133	141
92	99	98	106	107	118	135	145
89	95	98	107	104	112	130	144

DCT

F(u,v) =

898.0000	-149.5418	26.6464	-14.0897	0.7500	-5.7540	3.5750	0.0330
12.1982	-16.5235	-7.6122	5.2187	-0.2867	-1.9909	8.4265	1.2591
5.3355	-2.6557	2.3410	-9.9277	2.4614	4.4558	-3.1945	-3.1640
1.9463	-2.7271	1.5106	2.8421	-2.1336	-2.7203	-2.7510	5.4051
0.7500	-2.0745	0.8610	0.2085	2.5000	1.8446	2.0787	2.4750
7.9536	-2.6624	2.6308	0.4010	0.4772	3.3000	1.7394	0.3943
-4.1042	-0.1650	-0.6945	0.0601	0.0628	-0.7874	-0.8410	0.3496
-3.4688	2.3804	0.1559	0.8696	0.1142	-0.5240	-3.9974	-5.6187

Q =

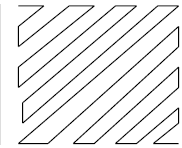
16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

F ° (u,v) = Redondear { F(u,v) / Q(u,v) } =

56	-14	3	-1	0	0	0	0
1	-1	-1	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

F ° (u,v) =

56	-14	3	-1	0	0	0	0
1	-1	-1	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0



Secuencia unidimensional: 56 -14 1 0 -1 3 -1 -1 EOB

Codificación transformacional: JPEG progresivo

JPEG progresivo:

- Tras la cuantificación (igual que en el *baseline*) los coeficientes se almacenan en un buffer y antes de transmitirlos se reordenan para transmitir la información secuencialmente en diversas pasadas de "calidad".
  - Primero calidad baja y luego mejorando
- Existe dos métodos: selección espectral y aproximaciones sucesivas
  - Selección espectral: los coeficientes de la DCT se agrupan por bandas espectrales y cada pasada transmite un banda

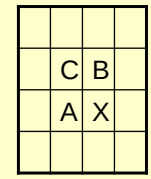
    Más sencillo, pero genera efecto de bloques en la formación
  - Aproximaciones sucesivas: cada coeficiente se transmite con un número de bits, empezando por los más significativos y finalizando con los más significativos.

    Más complejo, menor efecto de bloques, mayor compresión
  - Se suelen combinar (cada banda se transmite mediante aproximaciones sucesivas)

Codificación transformacional: JPEG sin pérdidas

JPEG modo secuencial sin pérdidas

- Predictor: diversas predicciones posibles sobre 3 píxeles
  - sin predicción
  - Predicción orden 1: A, B, ó C
  - Predicción orden 2: (A+B)/2
  - Predicción orden 3: (A+B-C), A+(B-C)/2, ó B+(A-C)/2

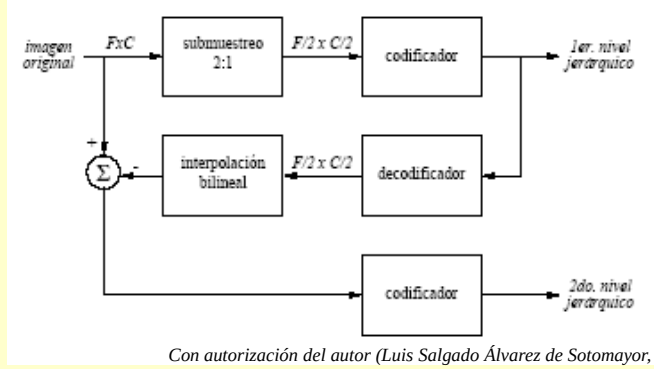


- Codificador estadístico
  - Se aplica un Huffman con tablas de especificaciones de los códigos

Codificación transformacional: JPEG jerárquico

JPEG modo jerárquico

- Diversas resoluciones espaciales
  - Versus modo progresivo que tiene distintas pasadas (de la misma resolución) a distinta calidad.
- Es necesario un filtro previo al submuestreo para evitar el aliasing, que tiene que ser compatible con el interpolador (suelen ser interpoladores bilineales)



Con autorización del autor (Luis Salgado Álvarez de Sotomayor, UPM)

Índice

- *Introducción*
- *Fundamentos*
- *Algoritmos de codificación de fuente sin pérdidas*
- *Teoría Tasa-Distorsión*
- *Cuantificación*
- ***Codificadores***
  - ***Introducción***
  - ***Codificación por forma de onda***
  - ***Codificación análisis-síntesis***
  - ***Codificación transformacional***

Índice

- Introducción
- Fundamentos
  - Modelado de fuentes
  - Teorema de codificación de fuente
- Algoritmos de codificación de fuente sin pérdidas
  - Introducción
  - Codificación Huffman
  - Modificación de fuente (extensión de fuente): RLC-ZRLC, M2F
  - Codificación aritmética
  - Codificación Lempel-Ziv
- Teoría Tasa-Distorsión
  - Introducción
  - Entropía diferencial
  - Función Tasa-Distorsión
    Introducción, Distorsión, Teorema Tasa-Distorsión
- Cuantificación
  - Introducción
  - Cuantificación escalar
  - Cuantificación uniforme
  - Cuantificación no uniforme
  - Cuantificación vectorial
- Codificadores
  - Introducción
  - Codificación por forma de onda
  - Codificación análisis-síntesis
  - Codificación transformacional

Bibliografía

- John G. Proakis, Masoud Salehi, “Communication Systems Engieering”, 2nd ed., Prentice Hall, 2002.
- D. Salomon, “Data Compression”, 3rd ed., Springer, 2004.