Transmisión de Datos
Codificación de Canal

José M. Martínez
Video Processing and Undestanding Lab (VPULab)
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
  - **Motivación**
  - Estrategias ARQ versus FEC
  - Modelo de canal de comunicación
  - Capacidad de canal
  - Teorema de codificación de canal ruidoso
  - Limites de la comunicación
- Códigos de canal
- Modulación codificada
- Aplicaciones de códigos de canal

Motivación (I)

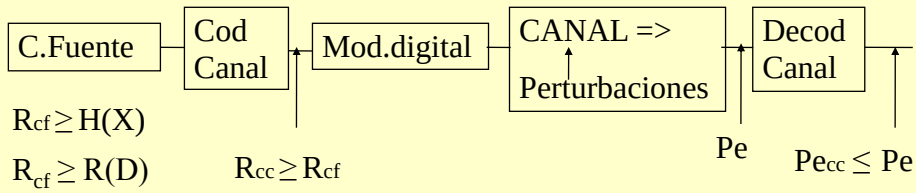
La codificación de fuente adapta las fuentes para su transmisión digital con una serie de límites en tasa y distorsión (en el caso de codificación con pérdidas).

Sin embargo a la hora de transmitir (o almacenar) las secuencias de símbolos codificados (generalmente secuencias de {0,1}) los errores son siempre posibles, siendo la Probabilidad de Error (Pe: nos referiremos a Pe_b, independientemente de la modulación) en transmisión función del tipo de modulación, ancho de banda y potencia de las señales que se usan para transmitir la señal, perturbaciones que incidan sobre el canal/medio (incluyendo defectos en soporte de almacenamiento), …

Si bien la Teoría de la Comunicación permite diseñar sistemas con baja Pe, en la práctica estos sistemas pueden implicar costes elevados o puede existir la necesidad de Pe=0.

Por lo tanto surge la necesidad de la Codificación de Canal para reducir/eliminar los errores en la transmisión (almacenamiento)-recepción (recuperación) del flujo de símbolos codificados procedente del codificador de fuente.

Motivación (II)



Índice

- Introducción
  - Motivación
  - **Estrategias ARQ versus FEC**
  - Modelo de canal de comunicación
  - Capacidad de canal
  - Teorema de codificación de canal ruidoso
  - Limites de la comunicación
- Códigos de canal
- Modulación codificada
- Aplicaciones de códigos de canal

Estrategias ARQ versus FEC (I)

ARQ: Automatic Repeat reQuest

- Se hace uso de códigos que detectan errores y cuando aparecen se solicita la retransmisión hasta que llega sin error.
  - Interesa que ARQ esté en niveles inferiores de la pila de protocolos para reducir el tiempo de reacción (espera de usuario) y los recursos consumidos

FEC: Forward Error Correction

- Se hace uso de códigos (más complejos) que adicionalmente a la detección son capaces de corregir errores.
  - Se transmiten más bits de redundancia pero al reducirse las retransmisiones se compensa la tasa efectiva total
  - Son los únicos a tener en cuenta en los sistemas sin canal de retorno (difusión, almacenamiento, …)

Estrategias ARQ vs FEC (II)

En las técnicas de corrección de errores (FEC) el error es corregido por el propio receptor por medio de la redundancia que introducía el código.

- El resto del capítulo se centrará en estas técnicas.

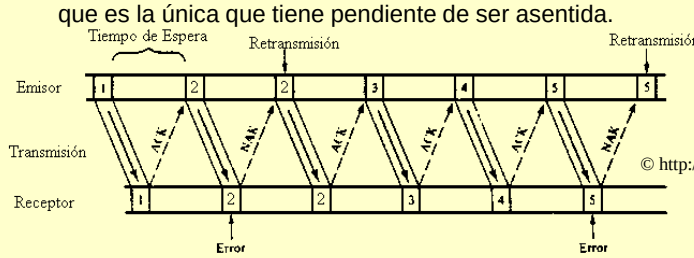
En el caso de las técnicas de detección de errores (ARQ) es necesario tener un código que detecte el mayor número posible de errores ya que si se detecta un error lo que se hace es pedir una retransmisión por parte del emisor, y si no se detecta error alguno, se supone que la secuencia de bits ha llegado sin errores.

- Estás técnicas se estudian en “redes”, pero …
- Existen tres tipos principales de técnicas ARQ:
  - ARQ de parada y espera
  - ARQ de envío continuo y rechazo simple
  - ARQ de envío continuo y rechazo selectivo

Estrategias ARQ vs FEC (III)

ARQ parada y espera

- En este sistema de transmisión, el emisor envía una trama y espera a que le llegue el asentimiento del receptor para enviar la siguiente (es posible el funcionamiento de este sistema partiendo de hipótesis simplificadoras: flujo unidireccional, los mensajes no se pierden –tiempo de espera-, siempre se detectan los errores). El receptor puede enviar un asentimiento positivo (ACK): la trama me ha llegado sin errores; o bien un asentimiento negativo (NAK): ha ocurrido un error. Si al emisor le llega un NAK, retransmite la última trama, en caso contrario transmite la siguiente. En este sistema el emisor sólo tiene que tener en memoria la última trama que ha enviado ya que es la única que tiene pendiente de ser asentida.

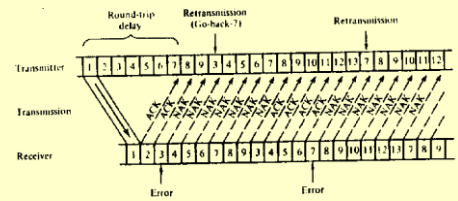


© http://jungla.dit.upm.es/~trdt/

Estrategias ARQ vs FEC (IV)

ARQ de rechazo simple

- En este caso, se supone que el emisor no espera a recibir un asentimiento del receptor sino que continua transmitiendo tramas que a su vez almacena en buffer hasta que sean asentidas: es una ventana deslizante en el emisor. Para diferenciar una trama de las demás les añade un número de secuencia supuestamente infinito, pero que no aumenta el número de bits de redundancia (es uno de los problemas en la práctica). El receptor asiente cada trama con su número correspondiente lo que libera la trama correspondiente en el buffer del emisor. Si una trama es errónea, el emisor vuelve atrás y retransmite a partir de esa trama (lo que hace inviable este sistema para probabilidades de error elevadas). El receptor sólo tiene que almacenar una trama en su registro pues al final siempre le llegan en orden.

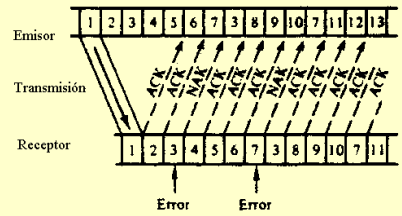


© http://jungla.dit.upm.es/~trdt/

Estrategias ARQ vs FEC (V)

ARQ de rechazo selectivo

- Para evitar perder eficiencia en transmisión, se busca repetir solo las tramas con error y no el resto. Para eso se usa el emisor del ARQ anterior: transmisión continua salvo que solo retransmite la trama defectuosa (lo sabe por el número de secuencia del asentimiento). El receptor se complica ya que ha de guardar en un registro todas las tramas posteriores a un error hasta que le llegue la retransmisión de la trama para poder entregarlas el orden. Esto complica el sistema bastante: son necesarias ventanas deslizantes tanto en receptor como en emisor y para probabilidades de error bajas no da una gran diferencia en eficacia respecto del sistema ARQ anterior.



© http://jungla.dit.upm.es/~trdt/

Índice

Modelo de canal de comunicación

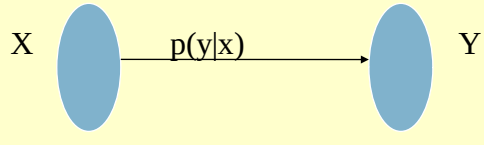
Canal de comunicación
- Medio a través del cual se puede transmitir/recibir información
- Medio en el cual se puede almacenar/recuperar información
  - Cables metálicos, fibras ópticas, espacio libre, ionosfera, discos magnéticos, discos ópticos, …

Las entradas a un canal de comunicación son señales que se entregan a su salida
- En otro lugar (medios de transmisión)
- En otro momento (medios de almacenamiento)

Al existir una relación entre entrada y salida se puede modelar como un Sistema
- Existen muchos factores que influyen en que la entrada no coincida con la salida
  - Atenuación, no linealidad, B limitado, propagación multitrayecto, ruido, defectos de medios físicos, …
- La relación entrada-salida en un canal de comunicación (sistema) suele ser compleja (relación estocástica).

Modelo de canal discreto

Canal discreto
- Los canales continuos (variables continuas) se pueden discretizar al tener ancho de banda limitado
- Relación entre alfabeto de entrada (X) y alfabeto de salida (Y)



- Un canal puede tener memoria (e.g., IES)
- Para un canal sin memoria y extensión de fuente de orden n
  - $p(y|x)=\prod p(y_i|x_i)$

Modelo de canal discreto binario simétrico

Para un canal sin memoria y extensión de fuente de orden n

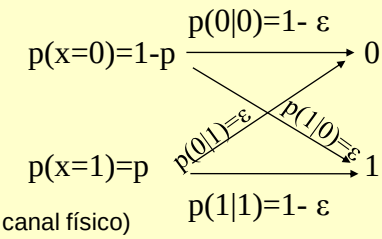
- $p(y|x)=\prod p(y_i|x_i)$

Si el canal es binario

- Y=X={0,1}

Si canal simétrico

- p(y<>x)=1-p(y=x)
  - p(1|0)=p(0|1)= ε
  - p(0|0)=p(1|1)=1- ε
  - ε = Pe (probabilidad de error del canal físico)



Si canal AWGN y señales binarias antipodales

- $\varepsilon=Pe_{canal}=Q(\sqrt{2Є_b/No})$
- Son canales limitados en potencia
  - No se puede aumentar la potencia para reducir Pe

Índice

- Introducción
  - Motivación
  - Estrategias ARQ versus FEC
  - Modelo de canal de comunicación
  - **Capacidad de canal**
  - Teorema de codificación de canal ruidoso
  - Limites de la comunicación
- Códigos de canal
- Modulación codificada
- Aplicaciones de códigos de canal

Capacidad de Canal

Codificación de fuente

- Límite de R sin error: $H(X) \leq R_{cf} \leq R_{tx}$
- Límite de R con distorsión menor que D: $R(D) \leq R_{tx}$

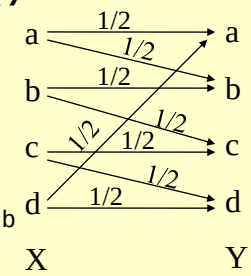
Codificación de canal:

- Existe otro límite para la transmisión sobre canal de comunicación
- Este límite indica que se puede limitar la Pe de transmisión siempre que la velocidad de transmisión (R_{tx}) sea menor que un límite denominada Capacidad de Canal (C).

Teorema de Capacidad de Canal Ruidoso [Shannon 1948]

- La limitación que producen las perturbaciones en un canal no es la "fiabilidad" de la transmisión sino la velocidad de la transmisión
  - Reduciendo la velocidad de transmisión se puede hacer Pe tan pequeña como se quiera

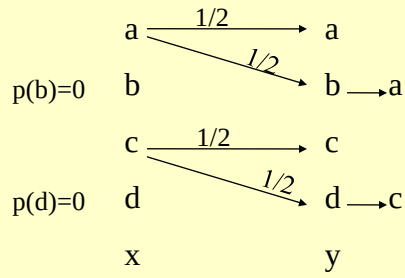
Capacidad de canal: ejemplo (I)



Sea el siguiente canal de comunicación

- En este canal se puede transmitir 4 símbolos
  - 2 bits/símbolo (R)
- En ese caso siempre hay probabilidad de error
  - E.g., si recibo b, es equiprobable haber emitido a o b
  - Decisión al azar ($\varepsilon = 1 - \varepsilon = 0.5$)
- Sin embargo, si con ese canal (posibles señales) transmito solamente dos señales
  - 2 señales => 1 bit/símbolo (R'=R/2)
- Tengo un canal sin error
  - Si solamente transmito a y c, llegue lo que llegue, no tengo error
  - Al reducir R reduzco Pe

Capacidad de canal: ejemplo (II)

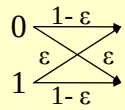


Esencia del teorema de codificación de canal

- Si las entradas están alejadas, sus salidas coincidirán (Pe) poco

Capacidad de Canal: Canal binario simétrico (I)

El modelo de canal binario simétrico no permite aplicar directamente codificación de canal



Es necesario hacer extensión de fuente para tener más símbolos y por lo tanto poder "jugar" a no transmitir todos.

Por lo tanto el fundamento es hacer bloques de n bits (extensión de fuente binaria) y no transmitir los 2^n bloques posibles (símbolos), sino un subconjunto de ellos (2^k) suficientemente alejados

- Centroides de clusters ... PEA … vs codificación de fuente con pérdidas

El "problema" reside en como elegir los 2^k a transmitir.

Capacidad de Canal: Canal binario simétrico (II)

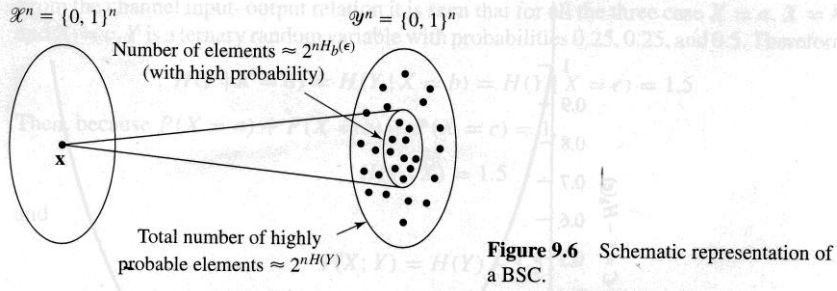


Figure 9.6 Schematic representation of a BSC.

Capacidad de Canal: Canal binario simétrico (III)

Para cada cadena de n símbolos (binarios) de entrada y un canal binario simétrico con Pe= ε, la salida será distinta en nε bits

Se puede demostrar que para cada bloque/cadena de n símbolos existen aproximadamente $2^{nHb(\varepsilon)}$ (realmente $\binom{n}{n\varepsilon}$) cadenas de salida altamente probables, que difieren de la de entrada en nε bits.

Las señales de salida altamente probables (alfabeto de palabras código Y son más que X, aunque todas con el mismo número de bits) serán aproximadamente $2^{nH(Y)}$ (según PEA).

Por lo tanto, el cociente entre todas las posibles secuencias de salida y las posibles secuencias correspondientes a una entrada (por el error de transmisión) será el número de secuencias de X que permitirían no crear confusión (otra cosa es saber cuáles son)

$M=2^{nH(Y)}/\ 2^{nHb(\varepsilon)} = 2^{n(H(Y)-Hb(\varepsilon))}$.

R (bits útiles/codificación canal) = #bits útiles por secuencia/bits en cada secuencia = $\log_2 M/n = H(Y)-Hb(\varepsilon)$

- ε depende del canal (no se puede controlar tras la instalación)
- H(Y) sin embargo depende de p(x) y ε, de forma que se puede maximizar R eligiendo la p(x) que maximice H(Y)

Capacidad de Canal: Canal binario simétrico (IV)

Se puede demostrar que para un canal binario simétrico

- ε = Pe -> 0 cuando n -> ∞
- R_{max} = C = 1 – Hb(ε)

- ε = 0 ó ε = 1 => C = 1
  - Se puede transmitir igual de bien si no hay errores como si todo son errores (vale con negar la señal)
    - –Si siempre dice la verdad o miente, siempre puedo acertar
  - El peor caso (C=0) cuando ε = 0.5 (no puedo transmitir nada sin error, pues es completamente aleatorio).

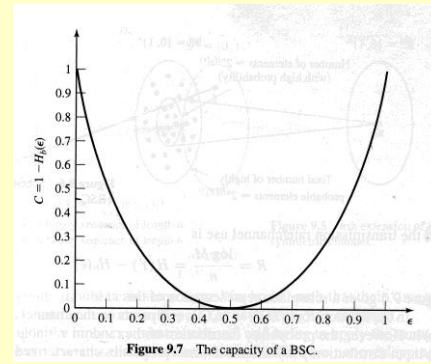


Figure 9.7 The capacity of a BSC.

Índice

Teorema de Codificación de Canal Ruidoso

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

C: cuánta información se puede transmitir sin generar errores en la transmisión (por confusión)

- Rtx = caudal
- C = cauce

Si Rtx ≤ C se puede transmitir sin error

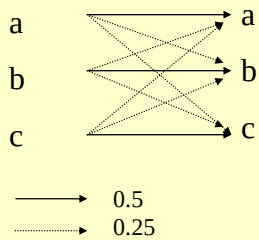
Si Rtx > C no se puede transmitir sin error (se desborda)

Se puede demostrar que para canal AWGN

- $C=W \log(1+(P/N_0W))$ [bis/seg] = $\log(1+(P/N_0W))/2$ [bits/transmisión] = $\log_2 M/n = \max_{p(x)} I(X;Y)$

Teorema de Capacidad de Canal: ejemplo

Calcular la capacidad del canal de la figura



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

Teorema de Capacidad de Canal: ejemplo - solución

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

$H(Y|X) = p(a).H(Y|x=a)+p(b).H(Y|x=b)+p(c).H(Y|x=c)$

- $H(Y|x=a)= -\sum p(yi|x=a).\log p(yi|x=a) =$
  $= - 0.5\log 0.5 – 0.25\log 0.25 – 0.25\log 0.25 =1.5$
- $H(Y|x=b)= H(Y|x=c)= H(Y|x=a)=1.5$

- $H(Y|X)=1.5[p(a)+p(b)+p(c)]=1.5$
  - Si no se cumpliese $H(Y|x=b)= H(Y|x=c)= H(Y|x=a)$, sería necesario conocer $p(x_i)$

$C = \max_{p(x)} \{H(Y)-1.5\}= \max_{p(x)} (H(Y))-1.5$

$\max_{p(x)} (H(Y))$ => será el caso de Y uniforme, pero no está claro que exista generalmente una X que produzca Y uniforme sobre un canal dado. En este caso particular (canal simétrico), Y será uniforme si X es uniforme.

- Si X uniforme pi=1/3 => H(Y)=H(X)=H(1/3,1/3,1/3)=1,585
- C = 1,585-1,5 = 0,085 bits (útiles)/transmisión
- Por lo tanto aunque transmita 3 símbolos (log 3 bits), cada transmisión solo podrá tener 0,085 útiles, de forma que si n=100, tendré 8,5 útiles, lo que implica que para transmitir 8 sin error necesitaría 92 de redundancia (… y sigo sin saber cuales).

Índice

Límites de la comunicación (I)

C=W log(1+(P/N_oW)) [bis/seg]

- Si aumenta P, aumenta C logarítmicamente (menor que lineal)
- Si aumenta W, C aumenta linealmente y disminuye logarítmicamente – pasa más ruido-
- Por lo tanto existe un límite de C con W

$$\lim_{W\to\infty} C = \log e \cdot \frac{P}{N_0} = 1.44 \cdot \frac{P}{N_0}$$

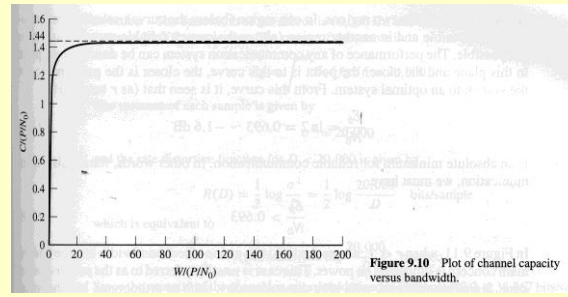


Figure 9.10 Plot of channel capacity versus bandwidth.

Límites de la comunicación (II)

En la práctica R < C
R < W log(1+(P/N_oW))
R/W = r < log(1+(P/N_oW))
ε_b = P/R
r < log(1+(P/N_o(R/r)))
r < log(1+r(ε_b/N_o))

Comunicación fiable por debajo de la curva r= log(1+r(ε_b/N_o))

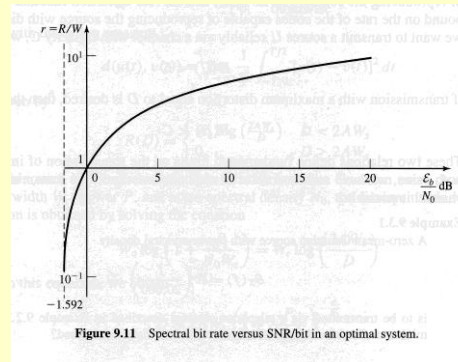


Figure 9.11 Spectral bit rate versus SNR/bit in an optimal system.

Límites de la comunicación (III)

Límite asíntotico en $\varepsilon_b/N_o = -1{,}592$ [dB]

$\varepsilon_b/N_o > -1{,}6$ dB => $\varepsilon_b/N_o = 0{,}693 = \ln 2$

- $\varepsilon_b/N_o > \ln 2$ versus $\varepsilon_b/N_o > 2 \ln 2$ (Constelaciones ortogonales en Teoría de la Comunicación: pre-Shannon => solamente puedo jugar con potencia y tiempo de símbolo)

- r<<1 => limitación en potencia (aumento W): constelaciones ortogonales
- r>>1 => limitación en ancho de banda (aumento P): constelaciones "concentradas" –e.g., QAM 256-

Límites de la comunicación (IV)

Sin pérdidas:

- $C \geq R_{tx} \geq R_{cfsp} \geq H(X)$

Con pérdidas

- $C \geq R_{tx} \geq R_{cfcp} \geq R(D_{max})$

Índice

- ***Introducción***
  - ***Motivación***
  - ***Estrategias ARQ versus FEC***
  - ***Modelo de canal de comunicación***
  - ***Capacidad de canal***
  - ***Teorema de codificación de canal ruidoso***
  - ***Limites de la comunicación***
- Códigos de canal
- Modulación codificada
- Aplicaciones de códigos de canal

Índice

- Introducción
- Códigos de canal
  - **Introducción**
  - Códigos lineales
  - Códigos cíclicos
  - Códigos convolucionales
  - Códigos basados en combinación
- Modulación codificada
- Aplicaciones de códigos de canal

Códigos de Canal: Introducción

La codificación de canal mejora la comunicación extremo-a-extremo reduciendo la Pe y el número de retransmisiones, sin modificar la SNR del sistema de comunicación subyacente (mejorar el medio de transmisión para reducir el ruido, poner amplificadores antes –para controlar el factor de ruido- y mejores -para aumentar la potencia-, …)

Para que se cumplan las condiciones del teorema de codificación de canal es necesario usar n bits para transmitir un bloque de k bits de información (n>k). Por lo tanto existe n-k bits de redundancia que en función de las restricciones que impongan (las cuales si no se cumplen en recepción indican la existencia de un error) darán lugar a códigos "más o menos" óptimos.

- Al no usarse las 2^n secuencias, sino solamente 2^k se cumplen las condiciones de codificación de canal. Queda encontrar las 2^k secuencias a transmitir que consigan reducir/eliminar la confusión entre las recibidas.
- Un código de canal se caracteriza por (n, k, ϵ, t)
  - k: número de bits de información
  - n: número de bits de la palabra código
  - ϵ: número de errores que es capaz de detectar
  - t: número de errores que es capaz de corregir
  - **Para que el código sea útil Pe_{bit} n≤{ϵ,t}**

Códigos de Canal: Códigos de paridad (I)

Los códigos más simples son los códigos con 1 bit de paridad

- (n,k,1,0) = (k+1, k, 1, 0)
- Pueden ser de paridad par (número de 1s par => p=0) o impar (número de 1s impar => p=0)
- Paridad par

$$p = x_1 \oplus x_2 \oplus x_3 \oplus .... \oplus x_k$$

$$\bar{c} = x_1 x_2 x_3 .... x_k \, p$$

- Paridad impar

$$p = x_1 \oplus x_2 \oplus x_3 \oplus .... \oplus x_k \oplus 1$$

$$\bar{c} = x_1 x_2 x_3 .... x_k \, p$$

Códigos de Canal: Códigos de paridad (II)

En recepción se calcula si se cumple la restricción: en este caso, que el número de 1s (en k) sea par o impar y coincida con el bit de paridad correspondiente (se denomina síndrome al resultado de esa comprobación)

$$\bar{r} = [(x_1 \oplus e_1),(x_2 \oplus e_2),...,(x_k \oplus e_k),(p \oplus e_{k+1})]$$

$$\bar{s} = [r_1 \oplus r_2 \oplus ... \oplus r_k \oplus r_{k+1}]$$

$$= [(x_1 \oplus e_1) \oplus (x_2 \oplus e_2) \oplus ... \oplus (p \oplus e_{k+1})]$$

$$= \underbrace{[(x_1 \oplus x_2 \oplus ... \oplus x_k) \oplus p]}_{p \oplus p = 0} \oplus \underbrace{(e_1 \oplus e_2 \oplus ... \oplus e_{k+1})}_{\begin{array}{l}1 \text{ si } \#\text{impar } e_i = 1 \\ 0 \text{ si } \#\text{par } e_i = 1\end{array}}$$

Si existe un número impar de errores se detecta, pero si es par no, por que los errores se cancelan.

Este sistema requiere técnicas ARQ y solamente es útil si $Pe_{bit} n \leq 1$.

Se puede añadir un segundo bit de paridad impar.

Códigos de Canal: Corrección simple

Los códigos de paridad son códigos detectores, pero no correctores. Para corregir, adicionalmente a la detección, hay que posicionar el error, lo que implica "trabajar" con matrices de 2 o más dimensiones (la transmisión será en serie).

Un ejemplo sencillo (14,8)

x1	x2	x3	x4	p1
x5	x6	x7	x8	p2
p3	p4	p5	p6	

Se calculan los síndromes por filas y columnas, y la intersección posiciona el error

x1	x2	x3	x4	p1	s1
x5	x6	x7	x8	p2	s2
p3	p4	p5	p6		
s3	s4	s5	s6		

- Solamente corrige un error y pueden enmascararse varios (útil $Pe_{bit} n \leq 1$)
- Los bits de paridad pueden tener error

Códigos de Canal: Decisión *soft* o *hard*

La decodificación de canal se basa en el cálculo del síndrome sobre los bits recibidos, que provienen de las señales demoduladas. Por lo tanto el demodulador, que devuelve la secuencia de bits en función de la etapa de decisión posterior al filtro adaptado (o acumulador), puede dejar pasar o no información sobre la fiabilidad, y aprovechar la posible correlación entre los bits de la palabra código.

Para decidir si una señal demodulada se asigna al símbolo 1 ó 0 se pueden tomar decisiones *soft* o *hard*

- *Soft*: la decisión se toma en función del vector recibido (palabra código) y con umbrales de fiabilidad (e.g., 8 niveles), de forma que posteriormente el decodificador de canal puede usar esta información para mejorar la decodificación.
  - Más compleja, pero mucho mejor. Suele usarse para códigos que trabajan en el límite teórico (e.g., Turbo Códigos).
- *Hard*: la decisión se toma bit a bit de la palabra código y de forma binaria (0 si por debajo de un umbral, 1 si por encima)
  - Más sencilla. Suele usarse para códigos de bloque.

Códigos de canal: Tipos

Los códigos de canal se agrupan en:

- Códigos lineales de bloque (sin memoria)
  - Los Códigos cíclicos de bloque son una particularización
- Códigos convolucionales (con memoria)
- Códigos combinados

Adicionalmente a los códigos existen técnicas que permiten también llevar a cabo codificación de canal. El ejemplo más importante es la corrección de errores a ráfagas, ya que los códigos no se comportan bien en ese caso (están diseñados para canales con errores aleatorios).

La técnica más usada para los canales con errores a ráfagas consiste en añadir un entrelazador entre el codificador de canal y el modulador (entre demodulador y decodificador en el receptor).

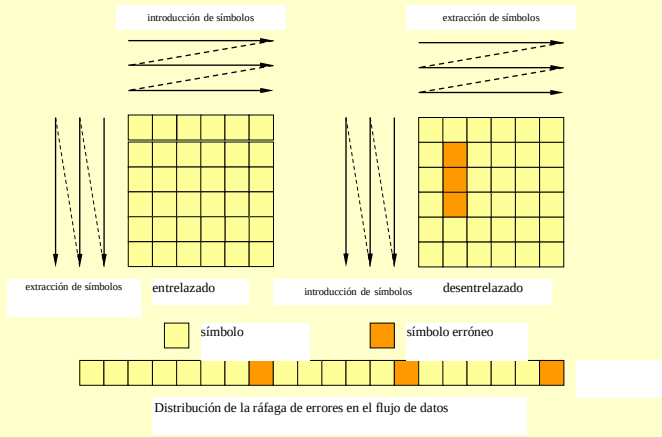
- El caso más simple de entrelazador introduce las palabras códigos por filas y extrae la información por columnas, logrando dispersar los errores de ráfagas y convertirlos en “pseudo-aleatorios” para ser luego procesados por el código de canal.

Códigos de canal: Entrelazado (I)

El entrelazado no supone un aumento de redundancia. Solamente se trata de un reordenamiento de los símbolos generados por el codificador externo. El funcionamiento esquemático es el siguiente:

- los símbolos generados por el codificador externo son introducidos en una matriz de almacenamiento línea a línea.
- Posteriormente estos símbolos son enviados a la siguiente etapa extrayéndolos de la matriz por columnas.
- Así, dos símbolos adyacentes generados por el codificador de canal externo, tras el entrelazado, estarán separados entre sí por tantos símbolos como entren en una columna.
- En el proceso de decodificación, los símbolos recibidos son introducidos por columnas, lo que provoca que las posibles ráfagas de errores estarán situadas en columnas. Por último, los símbolos se extraen por filas hacia el decodificador externo, con lo que las ráfagas de error se distribuyen en el flujo de datos, posibilitando su mejor corrección, al haber convertido un error a ráfagas en una serie de errores aleatorios.

Códigos de canal: Entrelazado (II)



Índice

Códigos de lineales: Introducción (I)

Un código de bloque (n,k) queda definido por $M=2^k$ secuencias de tamaño n bits denominadas **palabras código**.

- $C=\{c_1,c_2,\ldots c_M\}$
- $c_i=\{0,1\}^n$

Un código de bloque es lineal si la combinación lineal de palabras código es también una palabra código

$$\overline{c}_i \oplus \overline{c}_j = \overline{c}_k$$

Por lo tanto, en un código lineal siempre tiene que existir el elemento neutro

$$\overline{c}_1 = \underbrace{(0,0,0,\ldots,0)}_{n \text{ bits}}$$

$$\overline{c}_i \oplus \overline{c}_i = \overline{c}_k = \overline{c}_1$$

$$\overline{c}_i \oplus \overline{c}_1 = \overline{c}_i$$

Códigos de lineales: Introducción (II)

La linealidad del código depende de las palabras código, no del mapeo entre mensaje (x_i) y palabra código (c_i)

Normalmente, los códigos lineales se diseñan tales que:

$$\left.\begin{array}{l}\bar{x}_1 \rightarrow \bar{c}_1 \\ \bar{x}_2 \rightarrow \bar{c}_2\end{array}\right\} \quad \bar{x}_1 \oplus \bar{x}_2 \rightarrow \bar{c}_1 \oplus \bar{c}_2$$

Aunque esta propiedad no es necesaria para asegurar la linealidad del código, a partir de ahora trabajaremos con códigos lineales que la cumplan.

- Se verá más adelante sus ventajas.

Ejercicio de clase 16: Códigos lineales

Sea el código de bloque C(5,2)={00000, 10100, 01111, 11011}

- Demostrar que es lineal

Siendo los mensajes a transmitir X={00, 01, 10, 11}

- Calcule una asignación de palabras código a mensaje que cumpla la propiedad

$$\left.\begin{array}{l}\bar{x}_1 \rightarrow \bar{c}_1 \\ \bar{x}_2 \rightarrow \bar{c}_2\end{array}\right\} \quad \bar{x}_1 \oplus \bar{x}_2 \rightarrow \bar{c}_1 \oplus \bar{c}_2$$

- Y otra que no la cumpla

Comente los resultados

Ejercicio de clase 16: Códigos lineales – solución (I)

Sea el código de bloque C(5,2)={00000, 10100, 01111, 11011}

- **Demostrar que es lineal**

Ejercicio de clase 16: Códigos lineales solución (II)

Sea el código de bloque C(5,2)={00000, 10100, 01111, 11011}

Siendo los mensajes a transmitir X={00, 01, 10, 11}

- **Calcule una asignación de palabras código a mensaje que cumpla la propiedad**

- **Y otra que no la cumpla**

Códigos lineales: Definiciones

Peso Hamming de una palabra código

$w(c_i) = \#1\text{'s}$

Distancia Hamming entre dos palabras código

$d^H(c_i, c_j) = \#\text{elementos distintos bit a bit} = w(c_i \oplus c_j)$

Distancia mínima de un código

$d_{min} = \min_{i \neq j} d^H(c_i, c_j)$

Peso mínimo de un código

$w_{min} = \min_{c_i \neq 0} w(c_i)$

Teorema:

En un código lineal $w_{min} = d_{min}$

Códigos lineales: ejercicio propuesto 10

Demostrar que para un código lineal la distancia mínima del código coincide con el peso mínimo del código.

Códigos lineales: codificación

Una vez obtenidos los códigos, la codificación se lleva a cabo separando la secuencia de entrada en bloques de k bits y generando como salida la palabra código de n bits correspondientes.

La asignación de la palabra código se puede hacer de varias maneras:

- Tabla de "Look up"
  - Se tiene que guardar en memoria una tabla de M (2^k) elementos, cada uno con el mensaje (k bits) y su palabra código asociada (n bits).
  - Para cada mensaje se debe buscar en la tabla
- Al ser un código lineal existe una relación entre los bits del mensaje y los bits de la palabra código, por lo tanto se puede sustituir la tabla por una serie de operaciones lineales bit a bit o mediante una matriz.

Códigos lineales: Matriz generatriz (I)

En un código lineal sean las siguientes palabras (vectores de k bits):

$$e_1 = \overbrace{(1,0,0,...,0)}^{k}, e_2 = \overbrace{(0,1,0,...,0)}^{k}, ..., e_k = \overbrace{(0,0,0,...,1)}^{k}$$

$$g_i = codigo(e_i) \quad i \in [1,k]$$

Puedo escribir cada mensaje $\bar{x} = \{0,1\}^k = (x_1, x_2, ...x_k)$ como

$$\bar{x} = \sum_{i=1}^{k} x_i \cdot \bar{e}_i$$

A cada mensaje x hay que asignarle una palabra código c

Códigos lineales: Matriz generatriz (II)

$$c(\bar{x}) = c(\sum_{i=1}^{k} x_i \cdot \bar{e}_i) = \sum_{i=1}^{k} x_i \cdot c(\bar{e}_i) = \sum_{i=1}^{k} x_i \cdot \bar{g}_i = \bar{x} \begin{bmatrix} \bar{g}_1 \\ \bar{g}_2 \\ \dots \\ \bar{g}_k \end{bmatrix} = \bar{x} \begin{pmatrix} g_{11} & \dots & g_{1n} \\ \dots & \dots & \dots \\ g_{k1} & \dots & a_{kn} \end{pmatrix}$$

$$\underset{1xn}{\bar{c}} = \underset{1\times k}{\bar{x}} \cdot \underset{k\times n}{G}$$

- Por lo tanto, se corrobora la linealidad de estos códigos, ya que se pueden definir mediante una matriz G llamada Matriz generatriz (o generadora).
- La existencia de G hace la codificación muy sencilla y eficiente, pues para M grandes las tablas de “Look Up” ocupan mucho en memoria y requieren algoritmos de búsqueda óptimos (la subóptima sería la búsqueda lineal), mientras que G implica simplemente una multiplicación vector-matriz.

Códigos lineales: Matriz generatriz - ejemplo

Calcular la matriz generatriz del código lineal sistemático definido por:

- 00 -> 00000
- 01 -> 01111
- 10 -> 10100
- 11 -> 11011

Códigos lineales: Matriz generatriz – ejemplo – solución

Calcular la matriz generatriz del código lineal sistemático definido por:

- 00 -> 00000, 01 -> 01111, 10 -> 10100, 11 -> 11011

Se trata de un código (5,2), por lo que habrá que calcular los g_i (g_1 y g_2)

- $g_1=c(e_1)=c(10)=10100$
- $g_2=c(e_2)=c(01)=01111$

$$G = \begin{bmatrix} 10100 \\ 01111 \end{bmatrix}$$

Una vez obtenida G se puede calcular el código y comprobar que las palabras código están y son sistemáticas

- 00 * G = 00000
- 01 * G = 01111
- 10 * G = 10100
- 11 * G = 11011

Ejercicio de clase 17: Códigos lineales: Matriz generatriz

Calcular la matriz generatriz del código lineal sistemático (5,3) que incluye las siguientes palabras código {(10010),(01001),(10101)}

Ejercicio de clase 17: Códigos lineales: Matriz generatriz - solución (I)

Calcular la matriz generatriz del código lineal (5,3) sistemático que incluye las siguientes palabras código {(10010),(01001),(10101)}

Ejercicio de clase 17: Códigos lineales: Matriz generatriz - solución (II)

Tras tener G se puede generar todo el código y verificar su linealidad, su forma sistemática y que incluye las palabras código de partida

$$\bar{c} = \bar{x} G$$

Mensaje	Palabra código
000	
001	
010	
011	
100	
101	
110	
111	

Códigos lineales: Matriz generatriz sistemática

El código del ejemplo anterior cumple que cada palabra código empieza por el mensaje seguido por (n-k) bits de redundancia (podría ser al revés).

Estos códigos lineales se denominan sistemáticos, siendo condición necesaria y suficiente para que un código lineal sea sistemático (por delante/detrás) que G tenga la siguiente estructura:

$$G_{k\times n} = [I_{k\times k} \mid P_{k\times(n-k)}]$$

$$G_{k\times n} = [P_{k\times(n-k)} \mid I_{k\times k}]$$

Códigos lineales: Matriz de chequeo de paridad

Por Álgebra se sabe que se puede generar un código dual (n,n-k) al definido por la matriz G, tal que ambos códigos son ortogonales.

$$\underset{kxn}{G} \cdot \underset{nx(n-k)}{H^T} = \underset{kx(n-k)}{0}$$

H se denomina matriz de chequeo de paridad (se usa en decodificación).

Si el código C es sistemático (por delante) G=[I P] :

$$H_{(n-k)\times n} = [-P_{(n-k)\times k}{}^T \mid I_{(n-k)\times(n-k)}]$$

$$\text{Si código binario } c = \{0,1\}^n \Rightarrow -P^T = P^T$$

$$H_{(n-k)\times n} = [P_{(n-k)\times k}{}^T \mid I_{(n-k)\times(n-k)}]$$

Códigos lineales: matriz H - ejemplo

Calcular la matriz H correspondiente al código lineal definido por la siguiente matriz generatriz

$$G = \begin{bmatrix} 10100 \\ 01111 \end{bmatrix}$$

Códigos lineales: matriz H – ejemplo - solución

Calcular la matriz H correspondiente al código lineal definido por la siguiente matriz generatriz

Códigos lineales: Códigos Hamming

Son códigos lineales (n, k, 2, 1)

Restricciones:

$$\left.\begin{array}{l} n = 2^m - 1 \\ k = 2^m - m - 1 \\ d_{\min} = 3 \end{array}\right\} \; m \geq 2$$

Propiedades: $R_c = \frac{k}{n}$, si $m \uparrow\uparrow \; R_c \to 1$

- Rc es tasa de compresión por lo que en Codificación de Canal es menor que 1 (expande). Lo ideal sería 1 que indicaría no expansión (e.g., entrelazadores)

Como n son todas las secuencias de m bits (menos 1), se obtiene primero la matriz H (n columnas de (n-k) (=m) elementos/filas) poniendo todas las secuencias de m bits menos el elemento nulo, y posteriormente se obtiene G.

Lo más sencillo es generar códigos Hamming sistemáticos, pero pueden existir códigos Hamming no sistemáticos.

Códigos lineales: Códigos Hamming - ejemplo

Calcular un código Hamming (7,4) sistemático

Códigos lineales: Códigos Hamming – ejemplo - solución

$$\left.\begin{aligned} n &= 7 = 2^m - 1 \\ k &= 4 = 2^m - m - 1 \end{aligned}\right\} \quad m = 3$$

~~000~~
~~001~~ Ik
~~010~~ Ik
011
~~100~~ Ik
101
110
111

$$H_{3\times 7} = [P^T_{3\times 4} \mid I_{3\times 3}]$$

$$H_{3\times 7} = \left[\begin{array}{cccc|ccc} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{array}\right]$$

$$G_{4\times 7} = [I_{4\times 4} \mid P_{4\times 3}]$$

$$G_{4\times 7} = \left[\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{array}\right]$$

Ejercicio de clase 18: Códigos lineales: Códigos Hamming

Calcule la matriz generatriz de un código Hamming (3,1) sistemático

Ejercicio de clase 18: Códigos lineales: Códigos Hamming – solución

Calcule la matriz generatriz de un código Hamming (3,1) sistemático

Ejercicio propuesto 11 - Códigos lineales: Códigos Hamming

Calcule la matriz generatriz de un código Hamming n=15 sistemático.

Códigos lineales: Decodificación (I)

En decodificación de códigos bloque hay que dividir la secuencia recibida en bloques de n bits (entrada del decodificador) que darán lugar a la secuencia de salida formada por una serie de mensajes decodificados de k bits.

- Tabla de "Look up"
  - Para cada palabra código se busca en la tabla el mensaje
  - Si el error genera una palabra "no código" solamente soy capaz de detectar el error.
- Al ser un código lineal se puede hacer uso de las propiedades (restricciones) del mismo

$$\bar{s} = \bar{y} \cdot \mathrm{H}^T$$

$$\text{si } \bar{s} \neq (0,0,0,...,0) \Rightarrow \text{error}$$

Códigos lineales: Decodificación (II)

El objetivo principal de los códigos de canal es aumentar la distancia entre las señales que se transmiten de forma que se reduzca la Pe.

- El objetivo es que las señales transmitidas estén lo más alejadas posibles en el espacio n-dimensional.
- Una forma de saber lo alejadas que están las señales es la d^H entre palabras código.
- Calcular la d^H entre todas la palabras código es poco eficiente, de forma que para comparar códigos se suele comparar en función de la d_{min} (=w_{min} para códigos lineales)
- Para igual (n,k), una mayor d_{min} indica que las palabras códigos están menos correladas (código más separable) => mejor comportamiento del código de canal.

Códigos lineales: Decodificación (III)

Se puede demostrar que las capacidades de un código de bloque son

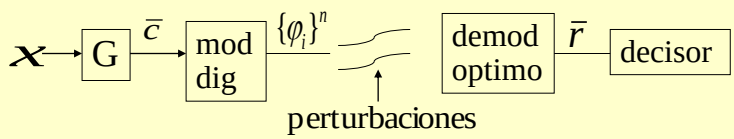
$$\varepsilon = \mathrm{d}_{\min} - 1$$

$$t = \left\lfloor \frac{d_{\min} - 1}{2} \right\rfloor$$

$$\mathrm{d}_{\min} = \varepsilon + t + 1 \quad , \varepsilon \geq t$$

Sin olvidar que se parte de la hipótesis de canales con errores aleatorios con $Pe_{bit}n<\{\varepsilon,t\}$

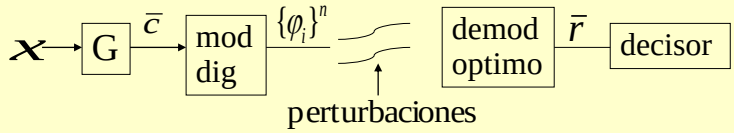
Códigos lineales: Decodificación blanda



Decisor blando (óptimo):

- Demodulación óptima
  - La señal r(t) tras pasar por un banco de filtros adaptados da lugar a un vector de muestras r.
  - Se decide como vector decodificado el punto de la constelación más cercano a r (distancia euclídea –números reales-)

Códigos lineales: Decodificación dura



Decisor duro (subóptimo –y más frecuente-):

- Demodulación subóptima
  - Se decide binariamente cada componente de la señal r(t) [r_k]
  - Posteriormente se selecciona la palabra código más cercana en distancia Hamming

Códigos lineales: Decodificación - ejemplo

Sea un código C(3,1)={000,111} que se modula mediante una modulación binaria antipodal (e.g., PSK) con $\varepsilon_s = 1$:

$$111 \rightarrow \{\varphi_1(t), \varphi_1(t), \varphi_1(t)\} \rightarrow (1,1,1)$$

$$000 \rightarrow \{-\varphi_1(t), -\varphi_1(t), -\varphi_1(t)\} \rightarrow (-1,-1,-1)$$

Si la salida del demulador es

$$\bar{r} = (0.5, 0.5, -3)$$

Calcular el mensaje decodificado con decisión blanda/dura.

Comente los resultados.

Códigos lineales: Decodificación – ejemplo - solución

Decisión blanda

Decisión dura

Códigos lineales: Decodificación sistemática dura - Matriz estándar

Para los códigos lineales existe la posibilidad de decodificación dura de forma sistemática mediante la matriz estándar.

Siendo $C=\{c_1,c_2, \ldots, c_M\}$, se define la matriz estándar del código como una matriz de $2^{(n-k)}$ filas por 2^k columnas tal que:

1. La primera fila son los c_i (M elementos) –empezando por (00…0)-
2. La siguiente fila se inicia con un vector de n {0,1} (e_i) cuyo peso sea mínimo y no esté todavía en la matriz. Esa fila se completa sumando al primer elemento de esa filas los elementos de la primera fila (e_i+c_i).
3. Se repite 2 hasta que no haya más secuencias de 0's y 1's de longitud n

Cada fila se denomina **coset** y cada e_i **coset leader**

Códigos lineales: Decodificación sistemática dura – Propiedades de la Matriz estándar

- Todos los elementos son distintos
- Tengo 2^n elementos = 2^k columnas x filas $\Rightarrow$ 2^{n-k} filas
- Si dos elementos y_1 e y_2 son del mismo coset, sus síndromes son iguales

Ejercicio propuesto 12 - Códigos lineales: Decodificación sistemática dura

Demostrar que el síndrome de todos los elementos de un coset de una matriz estándar es el mismo.

Códigos lineales: Decodificación sistemática dura – ejemplo

Calcular las matrices G, H y estándar del código lineal sistemático C(5,2)={00000, 10100, 01111, 11011}

Códigos lineales: Decodificación sistemática dura – ejemplo – solución (I)

Calcular las matrices G, H y estándar del código lineal sistemático C(5,2)={00000, 10100, 01111, 11011}

Códigos lineales: Decodificación sistemática dura – ejemplo – solución (II)

Calcular las matrices G, H y estándar del código lineal sistemático C(5,2)={00000, 10100, 01111, 11011}

Códigos lineales: Decodificación sistemática dura – ejemplo – solución (III)

Calcular las matrices G, H y estándar del código lineal sistemático C(5,2)={00000, 10100, 01111, 11011}

Ejercicio de clase 19 - Códigos lineales: Decodificación sistemática dura

Calcular la matriz estándar del código definido por:

- 00 -> 0000, 01 -> 0111, 10 -> 1011, 11 -> 1100

Ejercicio de clase 19 - Códigos lineales: Decodificación sistemática dura – solución

Calcular la matriz estándar del código definido por:

- ***00 -> 0000, 01 -> 0111, 10 -> 1011, 11 -> 1100***

Ejercicio propuesto 13 - Códigos lineales: Decodificación sistemática dura

Generar el código lineal C(6,3) que incluye como palabras código las siguientes: {(100101),(010111),(111001)}, así como sus matrices generatriz (G), de chequeo de paridad (H) y estándar.

Comentar las capacidades detectoras y correctoras del código, tanto a nivel general (para cualquiera de las matrices estándar posibles, como para la matriz estándar generada).

Códigos lineales: Decodificación sistemática dura (I)

Por cuestiones de memoria, no se suele guardar en memoria la matriz estándar, sino únicamente los síndromes y coset leaders. Por lo tanto en decodificación no se puede usar búsqueda exhaustiva y luego buscar por la columna la palabra código.

Por lo tanto la decodificación se basa (asumiendo decodificación dura) en buscar $d^H_{min}(c_i,y)$, para lo cual se busca el coset leader de y calculando su síndrome y a partir del mismo obteniendo el patrón de error (coset leader) que se suma a y para obtener la palabra código decodificada.

Códigos lineales: Decodificación sistemática dura (II)

Pasos sistemáticos:

- Encontrar el vector de muestras demoduladas r^n (reales)
- Tomar una decisión binario por umbral: r^n -> $y=\{0,1\}^n$
- Calcular el síndrome: $s=yH^T$
- Encontrar el coset leader del coset al que pertence y mediante el síndrome
- Decodificar c sumando (módulo 2) el coset leader a y.

La mejora que se consigue mediante decodificación soft (frente a hard en AWGN) es de apenas 2 dB y es mucho más complejo.

Además si se hace decodificación bit a bit pero con 8 niveles (decisión hard-soft) la diferencia con soft es de 0.1 dBs

- En la literatura hard(bit a bit)-soft(8 niveles) puede encontrarse como soft, pero es un compromiso entre los dos extremos.

Índice

Índice

Códigos cíclicos: Introducción

Los códigos cíclicos son códigos lineales de implementación sencilla

Tipos:

- Binarios: BCH (Bose, Chaudhuri, Hocquenghen)
- No binarios: RS (Reed Solomon)

Se basan en polinomios en lugar de matrices

Su característica diferenciadora es que cualquier desplazamiento cíclico de una palabra código es palabra código

Códigos cíclicos: ejemplo

C={c_1, c_2, c_3, c_4}={000, 110, 101, 011}

- $c_1^{(n)}=c_1$
- $c_2^{(1)}= c_3$
- $c_2^{(2)}= c_4$
- $c_2^{(3)}= c_2$
- $c_3^{(1)}= c_4$
- $c_3^{(2)}= c_2$
- $c_3^{(3)}= c_3$
- $c_4^{(1)}= c_2$
- $c_4^{(2)}= c_3$
- $c_4^{(3)}= c_4$

C={c_1, c_2, c_3, c_4}={000, 010, 101, 111}

- No cíclico (si lineal)

Códigos cíclicos: Estructura (I)

Se representan mediante polinomios (polinomio de palabra código)

$$C(p)=\sum_{j=1}^{n} c_j p^{n-j} = c_1 p^{n-1} + c_2 p^{n-2} + \ldots + c_{n-1} p + c_n$$

$$\text{ó } = c_1 + c_2 p + \ldots + c_{(n-1)} p^{n-2} + c_n p^{n-1}$$

$$\text{ó } = c_0 + c_1 p + \ldots + c_{(n-2)} p^{n-2} + c_{(n-1)} p^{n-1}$$

$$C(p)=(c_1, c_2, \ldots, c_{n-1}, c_n)$$

$$C^{(1)}(p)=(c_2, c_3, \ldots, c_{(n-1)}, c_n, c_1) = c_2 p^{n-1} + c_3 p^{n-2} + \ldots + c_n p + c_1$$

Sea $p \cdot C\ (p) = c_1 p^n + c_2 p^{n-1} + c_3 p^{n-2} + \ldots + c_{(n-1)} p^2 + c_n p$

Si se suma $c_1 + c_1 \Rightarrow$ sumar 0

$$p \cdot C\ (p) = c_1 p^n + c_1 + c_2 p^{n-1} + c_3 p^{n-2} + \ldots + c_{(n-1)} p^2 + c_n p + c_1$$

$$p \cdot C\ (p) = c_1 (p^n + 1) + C^{(1)}(p)$$

Por lo tanto $C^{(1)}(p)$ es el resto de dividir $p \cdot C\ (p) / (p^n + 1)$

Códigos cíclicos: Estructura (II)

Se puede demostrar análogamente

$$C(p)^{(1)} = p \cdot C\ (p) \bmod (p^n + 1)$$
$$C(p)^{(2)} = p^2 \cdot C\ (p) \bmod (p^n + 1)$$

...

$$C(p)^{(i)} = p^i \cdot C\ (p) \bmod (p^n + 1)$$

Ejercicio propuesto 14 - Códigos cíclicos

Demostrar razonadamente

$$C(p)^{(n)} = p^n \cdot C\ (p) \bmod (p^n + 1) = C(p)$$

Códigos cíclicos: Teorema del polinomio generador

En cualquier código cíclico (n,k) existe un polinomio g(p) de grado (n-k) llamado polinomio generador que cumple que:

$$g(p) = p^{n-k} + g_2 p^{n-k-1} + g_3 p^{n-k-2} + ... + g_{n-k} p + 1$$

g(p) es factor de ($p^n + 1$)

Siendo X(p) polinomio de secuencia de información

$$X(p) = x_1 p^{k-1} + x_2 p^{k-2} + x_3 p^{k-3} + ... + x_{k-1} p + x_k$$

Se cumple

$$C_X(p) = X(p) \cdot g(p)$$

Códigos cíclicos: ejemplo (I)

Código cíclico (7,4) $\Rightarrow$ g(p) de grado 3

$$p^7 + 1 = a(p) \cdot b(p) \cdot ... \cdot g(p)$$

$$p^7 + 1 = (p+1) \cdot \underbrace{(p^3 + p^2 + 1)}_{g(p)} \cdot \underbrace{(p^3 + p + 1)}_{g(p)}$$

$$C(p) = X(p) \cdot (p^3 + p^2 + 1)$$

$$X(p) = x_1 p^3 + x_2 p^2 + x_3 p + x_4$$

Tabla de 16 palabras mensaje (0000-1111) a código

Códigos cíclicos: ejemplo (II)

$$C(p) = X(p) \cdot (p^3 + p^2 + 1)$$
$$X(p) = x_1 p^3 + x_2 p^2 + x_3 p + x_4$$

$$1010 \Rightarrow X(p) = p^3 + p$$
$$C(p) = (p^3 + p)(p^3 + p^2 + 1) =$$
$$= p^6 + p^4 + p^5 + p^3 + p^3 + p =$$
$$= p^6 + p^5 + p^4 + p$$

No sistemático

Cumple la propiedad deseable

- X=x1+x2 => C=c1+c2
- Todos los cíclicos

0000	0000000
0001	0001101
0010	0011010
0011	0010111
0100	0110100
0101	0111001
0110	0101110
0111	0100011
1000	1101000
1001	1100101
1010	**1110010**
1011	1111111
1100	1011100
1101	1010001
1110	1000110
1111	1001011

Ejercicio propuesto 15 - Códigos cíclicos

Calcular el otro código cíclico (7,4)

$$p^7 + 1 = a(p) \cdot b(p) \cdot \ldots \cdot g(p)$$
$$p^7 + 1 = (p+1) \cdot \underbrace{(p^3 + p^2 + 1)}_{g(p)} \cdot \underbrace{(p^3 + p + 1)}_{g(p)}$$
$$C(p) = X(p) \cdot (p^3 + p + 1)$$

Códigos cíclicos: Matriz generatriz sistemática

Los códigos cíclicos son lineales y por lo tanto se pueden definir igualmente con G (y H). Si bien para un código lineal no existe una única G, si es cierto que existe una única G sistemática.

$$G = [I_k \mid P] = \begin{bmatrix} \overline{g}_1 \\ \overline{g}_2 \\ \dots \\ \overline{g}_k \end{bmatrix}$$

$$\overline{g}_i = (00\underset{\uparrow i}{1}00\underset{\uparrow k}{} \mid p_{i,1} p_{i,2} \dots p_{i,n-k}) \quad 1 \le i \le k$$

$$g_i(p) = p^{n-i} + \underbrace{p_{i,1} \cdot p^{n-k-1} + p_{i,2} \cdot p^{n-k-2} + \dots + p_{i,n-k}}_{p^{n-i} \bmod g(p)} = X_i(p) \cdot g(p)$$

Códigos cíclicos: Matriz generatriz sistemática - ejemplo

Calcular G sistemática

$(7,4) \rightarrow g(p) = p^3 + p^2 + 1$

$n = 7$

$i = 1,\dots,4$

$$p^6 \bmod (p^3 + p^2 + 1) = p^2 + p$$
$$p^5 \bmod (p^3 + p^2 + 1) = p + 1$$
$$p^4 \bmod (p^3 + p^2 + 1) = p^2 + p + 1$$
$$p^3 \bmod (p^3 + p^2 + 1) = p^2 + 1$$

$$G = \left[\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{array}\right]$$

Ejercicio de clase 20 - Códigos cíclicos: Matriz generatriz sistemática

Calcular G sistemática

$(7,4) \rightarrow g(p) = p^3 + p + 1$

$n = 7$

$i = 1,...,4$

Ejercicio de clase 20 - Códigos cíclicos: Matriz generatriz sistemática - solución

Calcular G sistemática

Códigos cíclicos: Códigos BCH

Los códigos BCH (Bose, Chaudhuri, Hocquenghem) son una subclase de códigos cíclicos que se pueden diseñar para corregir t errores. Su atractivo reside en la existencia de algoritmos eficientes de decodificación (y corrección) basados en las propiedades de los polinomios (no en síndrome y matriz estándar).

Para cada {m,t} existe un BCH con los siguientes parámetros:

$$n = 2^m - 1$$
$$n - k \le mt$$
$$d_{\min} = 2t + 1$$

Al ser {m,t} arbitrarios existe un gran número de posibilidades de códigos BCH.

- Se encuentran tabulados
  - Table 9.1 del Proakis (coeficientes de g(p) en octal)

Códigos cíclicos: Reed-Solomon

Subconjunto de códigos BCH no binarios

- Para cada c=(c_1,c_2,…c_n) los elementos c_i pertenecen a un alfabeto q-ario.
  - Generalmente q=2^k.
- k bits de información se “mapean” a un elemento c_i
- K elementos c_i se codifican con N elementos c_i
  - RS(N,K)

$$N = q - 1 = 2^k - 1$$
$$K = 1, 2, ..., N - 2$$
$$D_{\min} = N - K + 1$$
$$R_c = \frac{K}{N}$$

- Funcionan bien en combinación con modulaciones q-arias.
- Funcionan bien frente a ráfagas de errores, pues una ráfaga afecta a pocos símbolos q-arios, de forma que el código puede corregir esos pocos símbolos (muchos si binario)
- Se usan en los CDs de audio (audio-CD, no mp3)

Índice

- Introducción
- Códigos de canal
  - Introducción
  - Códigos lineales
  - ***Códigos cíclicos***
    - ***Introducción***
    - ***Estructura***
    - ***Teorema del polinomio generador***
    - ***Matriz generadora sistemática***
    - ***Códigos BCH***
    - ***Códigos R-S***
  - Códigos convolucionales
  - Códigos basados en combinación
- Modulación codificada
- Aplicaciones de códigos de canal

Índice

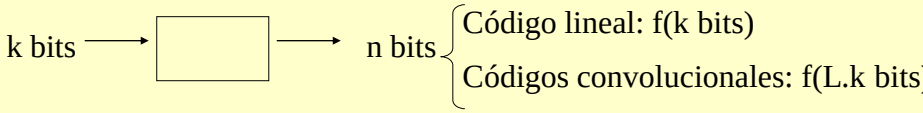
- Introducción
- Códigos de canal
  - Introducción
  - Códigos lineales
  - Códigos cíclicos
  - **Códigos convolucionales**
    - **Introducción**
    - **Representación: máquina de estados, secuencias generadoras, diagrama de estados, diagrama Trellis**
    - **Codificación**
    - **Códigos catastróficos**
    - **Decodificación óptima: algoritmo de Viterbi**
    - **Función de transferencia**
  - Códigos basados en combinación
- Modulación codificada
- Aplicaciones de códigos de canal

Códigos convolucionales: Introducción

Los códigos convolucionales se diferencian principalmente de los lineales de bloque por tener memoria.

Al igual que los códigos de bloque, se asignan n bits de salida cada bloque de k bits de entrada, pero los n bits no dependen únicamente de los k actuales sino de un número entero de bloques (L) de k bits de entrada.

Los códigos convolucionales dependen por lo tanto de una serie de estados anteriores ((L-1)*k) por lo que su codificación se puede ver como una máquina de estados finitos.

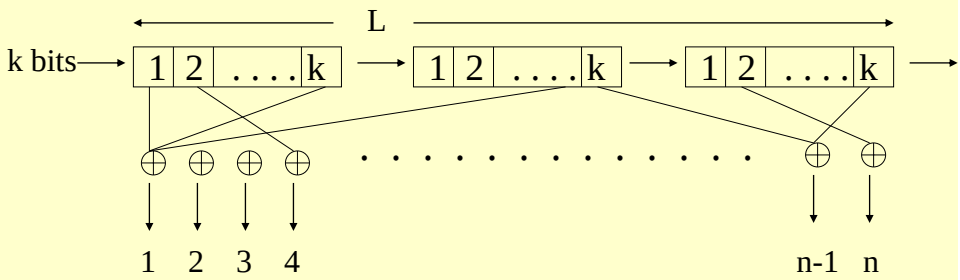


Códigos convolucionales: Representación

Al tratarse de una máquina de estados finitos, se puede representar de esa forma, pero no es la única. Un código convolucional se suele representar normalmente de una de las siguientes cuatro representaciones equivalentes (cada una apropiada para un aspecto, interpretación, implementación, …)

- Máquina de estados
- Secuencias generadoras
- Diagrama de estados
- Diagrama Trellis

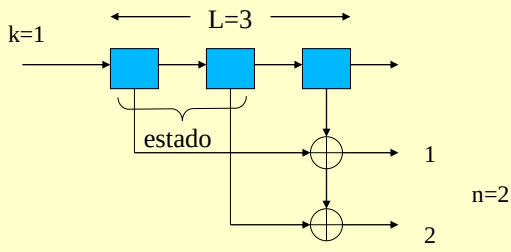
Códigos convolucionales: Máquina de estados



Los n bits de salida no solamente dependen de los k de entrada, sino también de los (L-1)k anteriores.

Los registros son una máquina de estados con $2^{k(L-1)}$ estados

Códigos convolucionales: Máquina de estados – ejemplo

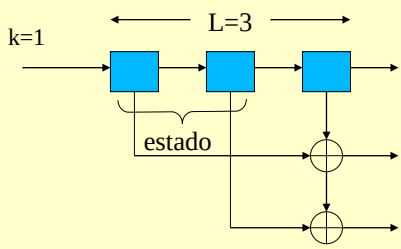


Códigos convolucionales: Secuencias generadoras

Otra forma de representar un código convolucional en lugar de la máquina de estado son las secuencias generadoras g1..gn cada una con kL elementos binarios

- Cada g_{ij}=1 (i=1..n, j=1..kL) indica que la celda j de la máquina de estados está conectada a la salida i

Códigos convolucionales: Secuencias generadoras - ejemplo



Secuencias generadoras:

g1=[1 0 1]

g2=[1 1 1]

Códigos convolucionales: Diagrama de estados

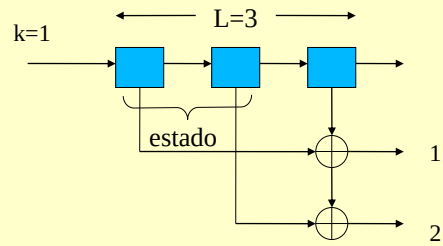
Como los códigos convolucionales tienen memoria finita se pueden representar (como cualquier máquina de estados finitos) con un diagrama de estados.

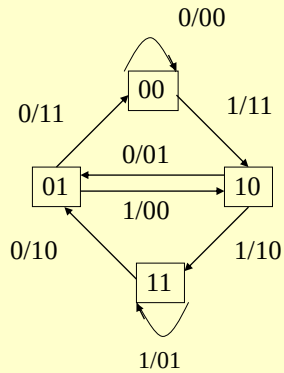
- $2^{k(L-1)}$ estados
- Cada uno de los estados tendrá 2^k transiciones de entrada y 2^k transiciones de salida.

Códigos convolucionales: Diagrama de estados - ejemplo

$2^{(L-1)k} = 2^2 = 4$ estados/nodos

$2^k = 2$ arcos (entrada y salida)





Códigos convolucionales: Diagrama de Trellis

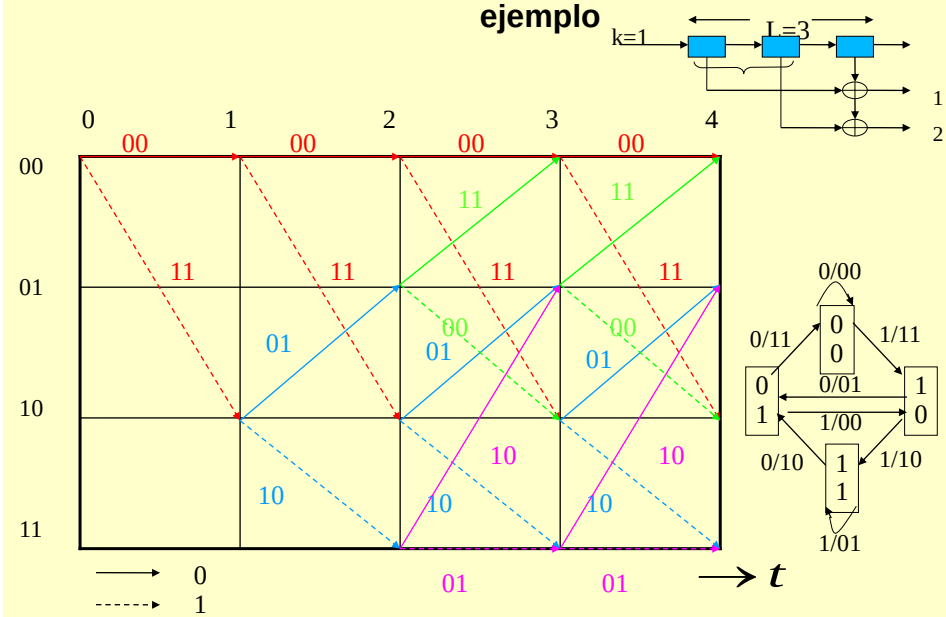
Otra forma de especificar los códigos convolucionales es su Diagrama de Trellis (enrejado)

- Representa la transición entre estados según evoluciona la codificación
- Se parte siempre del reposo (los Lk bits a cero)
- $2^{k(L-1)}$ estados
- En régimen permanente cada uno de los estados tendrá 2^k transiciones de entrada y 2^k transiciones de salida.

Representación gráfica

- Si k=1, solamente hay dos transiciones, de forma que se puede representar con líneas continuas/discontinuas cada transición debida a una entrada {0,1}
- Sobre cada transición se indica la salida que genera esa transición
  - o se verá su utilidad en decodificación
- Es recomendable hacer uso de colores

Códigos convolucionales: Diagrama de Trellis - ejemplo



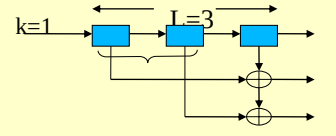
Ejercicio propuesto 16 - Códigos convolucionales: representación

Sea un código convolucional (3,1) con g1=[1 1 0], g2=[0 1 0], y g3=[1 0 1]. Dibujar la máquina de estados, el diagrama de estados y el diagrama Trellis de este código (Septiembre 2005)

Códigos convolucionales: Codificación

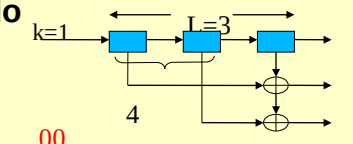
1. Se parte del estado de reposo (todos los registros a 0)
2. El mensaje de entrada se divide en bloques de k bits.
   - ★ Se asume mensaje de un número de bits múltiplo de k.
3. Cada k bits de entrada se calculan (y transmiten) n bits de salida.
4. Al terminar el mensaje se añaden (L-1)k ceros y se generan por lo tanto (L-1)n bits de salida quedando todos los registros de nuevo en el estado de reposo a la espera de un nuevo mensaje.

Códigos convolucionales: Codificación - ejemplo

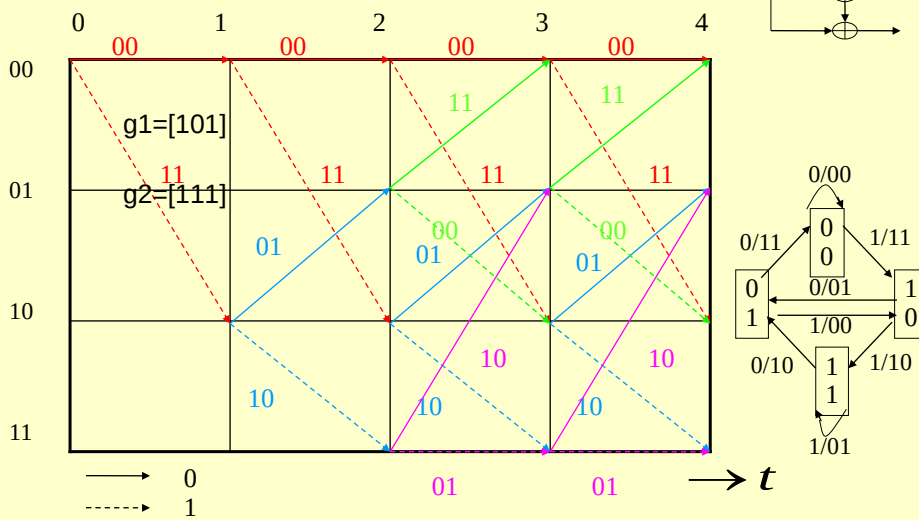


Codificar X={1101011} con el código convolucional descrito mediante la máquina de estados del diagrama.

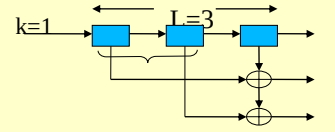
Códigos convolucionales: Codificación - ejemplo



Codificar X={1101011}



Códigos convolucionales: Codificación – ejemplo - solución



X={1101011}

Se parte del estado de reposo

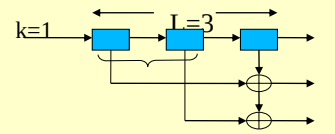
Al final hay que añadir (L-1)k ceros = (3-1)1 ceros =2 ceros

X*={1101011.00}

C={11.10.10.00.01.00.10.10.11}

Estado	Bit entrada	Salida
00	1	11
10	1	10
11	0	10
01	1	00
10	0	01
01	1	00
10	1	10
11	0*	10
01	0*	11
00 (reposo)		

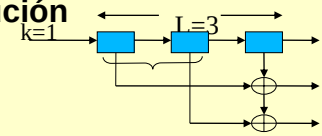
Ejercicio de clase 21 - Códigos convolucionales: Codificación



Codificar X={1010101} con el código convolucional descrito mediante la máquina de estados del diagrama.

Ejercicio de clase 21 - Códigos convolucionales: Codificación - solución

X={1010101}



Estado	Bit entrada	Salida

Ejercicio propuesto 17 - Códigos convolucionales: codificación

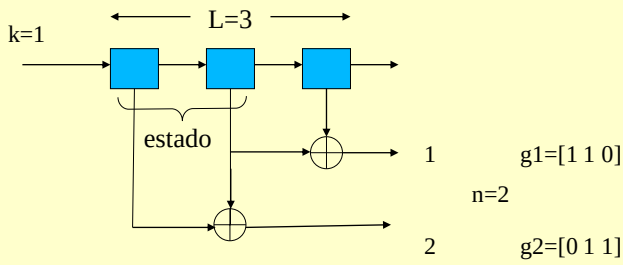
Para el código convolucional (3,1) con g1=[1 1 0], g2=[0 1 0], y g3=[1 0 1] (ejercicio propuesto anterior). Codificar la secuencia de información X={1011} (Septiembre 2005)

Códigos convolucionales: Códigos catastróficos

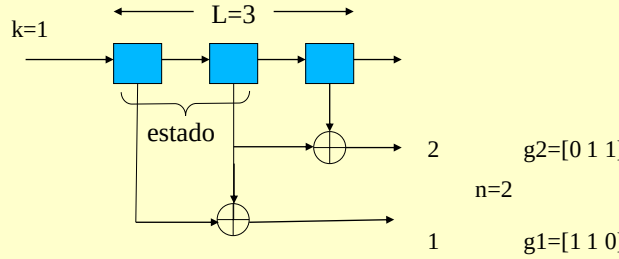
Los códigos convolucionales buscan “mapear” secuencias de información a palabras código lo más diferentes posible.

Cuando secuencias de información muy distintas se mapean a códigos con pocas diferencias tenemos los **códigos catastróficos** que como es lógico deben de ser evitados.

Códigos convolucionales: Códigos catastróficos – ejemplo



Códigos convolucionales: Códigos catastróficos – ejemplo - solución



Si x={1111…11}
- x'={1111…1100}
- c={10.01.00.00…..00.00.10.01)
  d=4

Si x_1={0000…00}
- x_1'={0000…0000}
- c_1={00.00.00.00…..00.00.00.00)
  d=0

Muy poco diferentes para
d(x,x_1)=longitud de la secuencia

Códigos convolucionales: Decodificación

En decodificación de códigos de bloque se hablaba de decodificación blanda (soft) basada en distancia euclídea, y decodificación dura (hard) basada en distancia Hamming de cada componente individual.
- En ambos casos se buscaba la secuencia que estaba a una distancia mínima de una secuencia posible (palabra código), decodificándose esa (con posibilidad de equivocación).

En códigos convolucionales esto es equivalente a buscar una secuencia a distancia mínima en el Trellis, o lo que es lo mismo, la secuencia de máxima verosimilitud (ML: Maximum-Likelihood).
- Por lo tanto se tiene que hacer uso de un algoritmo de búsqueda en el Trellis de esa secuencia de ML.
- El algoritmo óptimo más popular (y costoso e "ineficiente") es el algoritmo de Viterbi, que en decodificación de códigos convolucionales se suele usar en modalidad dura (hard).
  - El algoritmo de Viterbi aparece en muchos temas de "señal": IES, reconocimiento de voz, clasificación de patrones, ….

Códigos convolucionales: Decodificación Viterbi dura (I)

El objetivo es encontrar la secuencia codificada c a mínima distancia Hamming de la secuencia recibida y (tras decisión dura => por umbral bit a bit)

- Habrá m "saltos", cada uno correspondiente a n bits de salida (equivalentes a k bits de entrada)
  - Cada secuencia a decodificar tendrá por tanto m.n bits y generará m.k bits de salida decodificada
  - Cada subsecuencia de n bits será y_i y se buscará la c_i posible que se encuentre a mínima distancia
- Para cada salto i ($1 \le i \le m$) se buscará $d_H(c_j,y_i)$ (j=[1..estados*ramas])
- Para la secuencia y de entrada se buscará aquella secuencia codificada posible c, tal que $d_H(c,y)$ sea mínima
  - Si fuese decodificación blanda sería igual pero con distancia euclídea y espacio n-dimensional.

Códigos convolucionales: Decodificación Viterbi dura (II)

El problema genérico a resolver es por lo tanto:

- Dado un vector de entrada y (recibido y decodificado duro) hay que encontrar el camino a través del Trellis (posibilidades de codificación) que empezando en el estado nulo, vuelva al estado nulo tras m saltos y cuya distancia sea la mínima con alguna secuencia posible c.
- El problema se puede resolver fácilmente gracias a que la distancia d(c,y) es suma de $d_i(c_i,y_i)$ independientes.

El número de ramas que entran (y salen) de cada estado del Trellis (en permanente) es 2^k, siendo $2^{(L-1)k}$ el número de estados.

Códigos convolucionales: Decodificación Viterbi dura (III)

Sea Λ_{i-1} el conjunto de 2^k estados (de entrada) "supervivientes" conectados al estado $S_i=l$ ($1 \le i \le m$)

- i=0 => estado de reposo previo
- El camino óptimo $S_0 \rightarrow S_{i-1} = \lambda$ se denomina (camino) superviviente.

Si el camino $S_0 \rightarrow S_i = l$ es el camino óptimo (o uno de ellos) entonces tiene que ser la concatenación del camino $S_0 \rightarrow S_{i-1} = \lambda$ ($\lambda \in \Lambda_{i-1}$) y la rama óptima entre el $S_{i-1}=\lambda$ (o uno de ellos) (superviviente de distancia mínima) y el estado $S_i=l$.

- Aquella que tenga distancia mínima entre c_i e y_j.

Por lo tanto, para calcular el superviviente de $S_i=l$ es suficiente tener los supervivientes de Λ_{i-1} y las métricas para todo $S_{i-1}=\lambda$, así como las distancias de todos los caminos $\Lambda_{i-1} \rightarrow S_i$

- Se selecciona como superviviente el (2^k) de mínima métrica (o varios si la misma)

Se empieza en $S_0=0$ y se termina en $S_m=0$, siendo el último superviviente el camino óptimo (máxima verosimilitud)

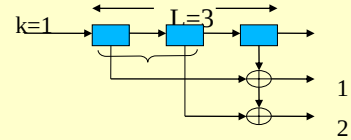
- $\mu(S_0=0,S_i=l)=\min_{\lambda \in \Lambda_{i-1}}\{\mu(S_0=0,S_{i-1}=\lambda)+\mu(S_{i-1}=\lambda,S_i=l)\}$

Códigos convolucionales: Algoritmo Viterbi para decodificación óptima de código convolucional

Sea la secuencia recibida y (tras decisión por umbral bit a bit)

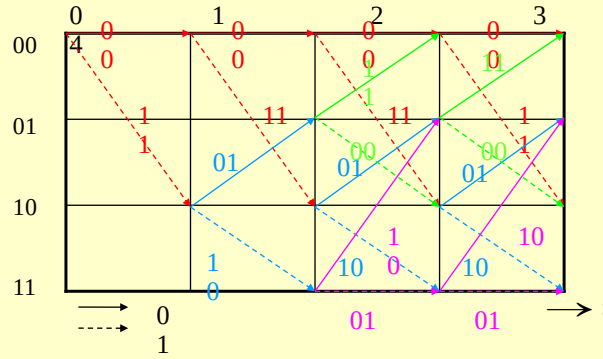
1. y se divide en m subsecuencias yi de n bits
2. Se dibuja el Trellis para m transiciones. Las (L-1) últimas solamente transiciones correspondientes a subsecuencias de entrada de ceros
3. Se empieza en S_0(solamente en estado 0) con métrica acumulada=0
4. Se calcula cada transición $\{s_i\} \rightarrow \{s_{i+1}\}$
5. Se suman los acumulados de cada estado de $\{s_i\}$ con la distancias asociadas a cada transición para calcular todos los candidatos acumulados en s_{i+1}.
6. Para cada s s_{i+1} se eligen los caminos que dan menor acumulado (supervivientes): mínima métrica
7. Si he terminado (i=m) → 8, si no i++ → 4
8. Voy hacia atrás por el superviviente para calcular la secuencia de máxima verosimilitud y su mensaje de información asociado (bits de entrada que generan cada transición)

Códigos convolucionales: Decodificación Viterbi – ejemplo

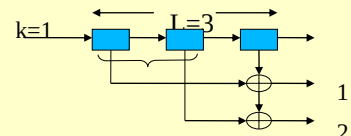


Para el código descrito por la máquina de estados y Trellis de la derecha decodificar la secuencia

y= {01.10.11.11.01.00.01}



Códigos convolucionales: Decodificación Viterbi – ejemplo

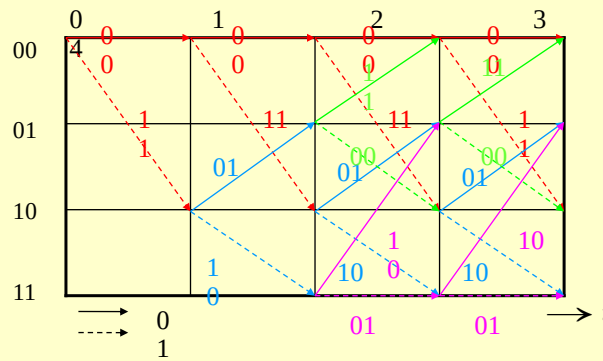


Para el código descrito por la máquina de estados y Trellis de la derecha decodificar la secuencia

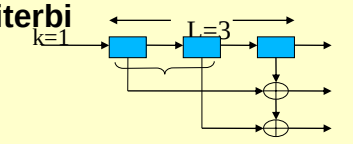
y= {01.10.11.11.01.00.01}

La secuencia de ML es c={11.10.10.11.00.00.00} y el mensaje decodificado m={1.1.0.0.0}

- Mensaje emitido más probable
- $d_H(c,y)=4$
  - No existe ninguna otra posible secuencia c en el Trellis con menor distancia
  - En otros casos podría ser igual

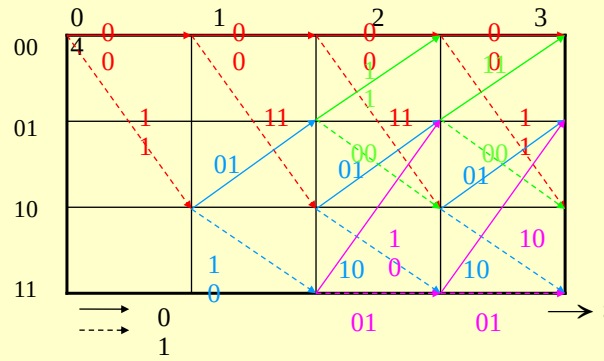


Ejercicio de clase 22 - Códigos convolucionales: Decodificación Viterbi

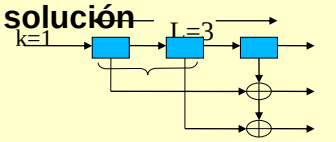


Para el código descrito por la máquina de estados y Trellis de la derecha decodificar la secuencia

y= {11.01.11.00.11.01.11}



Ejercicio de clase 22 - Códigos convolucionales: Decodificación Viterbi - solución



Para el código descrito por la máquina de estados y Trellis de la derecha decodificar la secuencia

y= {11.01.11.00.11.01.11}

Ejercicio propuesto 18 - Códigos convolucionales: Decodificación Viterbi

Para el código convolucional (3,1) con g1=[1 1 0], g2=[0 1 0], y g3=[1 0 1] y habiendo codificado la secuencia de información X={1001} (ejercicio propuesto anterior), se recibe la secuencia 101101100010001000. Decodifique la secuencia de información recuperada mediante el algoritmo de Viterbi. (Septiembre 2005)

Códigos convolucionales: Decodificación Viterbi – consideraciones prácticas

El problema del algoritmo de Viterbi es el retardo y las necesidades de memoria, ya que no se puede empezar a decodificar hasta tener toda la secuencia transmitida y hay que almacenar todos los candidatos. Además las secuencias de códigos convolucionales suelen ser largas (para aprovechar mejor las propiedades de los códigos).

Se suelen buscar soluciones subóptimas, siendo una de ellas la **truncación camino-memoria**.

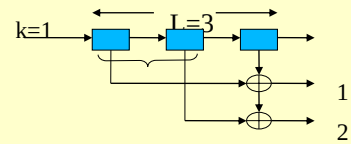
- Se decide cada δ saltos y el resto de la secuencia no varía la decisión tomada.
- El retardo es menor (n δ bits de salida, k δ bits de entrada) y solamente es necesaria memoria para los supervivientes de δ estados (más los intermedios hasta la decisión).
- Simulaciones han demostrado que para δ≈5L la degradación en el rendimiento es despreciable.
- A mayor tamaño de la secuencia de codificación Viterbi se comporta peor, de forma que para longitudes muy grandes se utilizan otros esquemas de búsqueda de camino de máxima verosimilitud subóptimos.

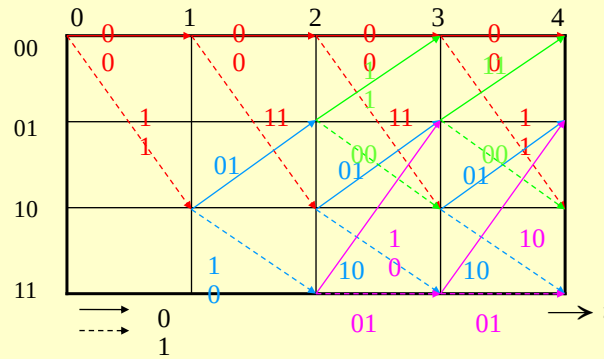
Códigos convolucionales: Función de transferencia

- La función de transferencia de un código convolucional proporciona información sobre todos los caminos a través del Trellis que empiezan en el estado nulo y vuelven a él
  - tras salir del mismo => se ignoran los bucles del estado nulo
- Según lo visto anteriormente cualquier secuencia código de un convolucional es un cambio a través del Trellis que empieza en el estado nulo y vuelve a él
- Para obtener la función de transferencia se divide el estado nulo en dos: partida y llegada
- Para cada arco conectando dos estados se define una función $D^{\alpha}N^{\beta}J$
  - α: número de 1's en la secuencia de salida
  - β : número de 1's en la secuencia de entrada
- La función de transferencia de un código convolucional es la función de transferencia del grafo del Trellis entre dos estado nulos y se denota por T(D,N,J).
  - Reglas de obtención de funciones de transferencia en grafos
    - Matemática discreta
- Cada elemento de T(D,N,J) se corresponde con un camino nulo-nulo (salvo el bucle nulo)
  - El exponente de D es el número de 1's en la palabra código correspondiente a ese camino (su distancia Hamming)
  - El exponente de N es el número de 1's en la palabra mensaje correspondiente
  - El exponente de J son las ramas (saltos) entre esos dos estados para cada camino

Códigos convolucionales: Función de transferencia – ejemplo

Para el código descrito por la máquina de estados y Trellis de la derecha calcular la función de transferencia





Códigos convolucionales: Función de transferencia – ejemplo – solución (I)

Estados

- 00 – a (a', a'')
- 01 – b
- 10 – c
- 11 – d

a' -> a'' (se ignora)

a' -> c (1/11)

c ->

- b (0/01) ->
  - a'' (0/11)
  - c (1/00)
- d (1/10) ->
  - b (0/10)
  - d (1/01)

Códigos convolucionales: Función de transferencia – ejemplo – solución (II)

Ecuaciones

- $X_c=X_{a'} D^2NJ + X_b NJ$
- $X_b=X_c DJ + X_d DJ$
- $X_d = X_c DNJ + X_d DNJ$
- $X_{a''} = X_b D^2J$

- $X_{a'}=(X_c-X_bNJ)/(D^2NJ)$ [1]
- $X_b=X_cDJ+X_dDJ$ [2]
- $X_d=(X_cDNJ)/(1-DNJ)$ [3]
- [2]+[3] => $X_c=X_b(1-DNJ)/(DJ)$ [4]
- $X_{a''} = X_b D^2J$ [5]
- [1]+[4]+[5] => $X_{a'}= \{X_{a''}(1-DNJ-DNJ^2)\}/\{(D^2J)(DJ)(D^2NJ)\}$

Función de transferencia

- $T(D,N,J)=X_{a''}/X_{a'} = (D^5NJ^3)/(1-DNJ-DNJ^2)$

Códigos convolucionales: Función de transferencia – ejemplo – solución (III)

Función de transferencia

- $T(D,N,J)=X_{a''}/X_{a'} = (D^5NJ^3)/(1-DNJ-DNJ^2)$

En forma polinómica: $T(D,N,J)=D^5NJ^3+D^6N^2J^4+D^6N^2J^5+D^7N^3J^5+\ldots$

- D^5NJ^3: Existe un camino con 5 1's de salida, 1 1 de entrada y 3 saltos
  - Como hay dos estados de reposo, $d_H=5=w$
    Esto es similar a la $d_{min}=w_{min}$ en códigos bloque
  - El exponente mínimo de D se denomina "distancia libre" (d_{free})
    Juega un papel similar a la d_{min} de códigos de bloque
  - Este camino se corresponde a la entrada 1 (1.00 => 11.01.11)
- $T(D,N,J)=\Sigma_{d>=dfree}{}^{\infty} a_d D^d N^{f(d)} J^{g(d)}$
  - $ad_{free}=1$
  - $a_d=\{0,1\}$ $d>d_{free}$
  - Pueden repetirse (caminos con igual número de 1's de salida, …)
  - No aparecen todos los caminos correspondientes a una secuencia de entrada debido a que hasta que no se sale del estado nulo no cuenta, y cuando se llega a el se acaba de contar.
    Por ejemplo: 01.00 sería D^5NJ^4, pero es D^5NJ^3

Índice

Índice

Códigos basados en combinación: Introducción

El rendimiento de los códigos de bloque y códigos convolucionales depende de las propiedades de distancia (d_{min} y d_{free}, respectivamente)
- Para diseñar códigos de bloque con d_{min} grande es necesario aumentar n y por tanto la complejidad del codec
- Para diseñar códigos convolucionales con d_{free} grande es necesario aumentar el número de estados, la longitud de la secuencia, … lo que aumenta los requisitos de memoria y el retardo

Para mejorar las prestaciones de un código de canal sin aumentar exponencialmente la complejidad se han propuesto soluciones basadas en la combinación de códigos “simples”.
- La decodificación es generalmente subóptima al basarse en la combinación de los decodificadores de los códigos “simples”, aunque suele ser suficiente en la mayoría de los casos.

Las tipos de códigos combinados más usados son:
- Códigos Producto
- Códigos concatenados (serie)
- Turbo Códigos

Los códigos en paralelo (generalmente con códigos de bloque sistemáticos) añaden redundancia por diversidad pero no son en si una combinación de códigos (aunque tienen su utilidad)

Códigos producto

Son códigos combinados que se basan en dos códigos de bloque C1(n1,k1) y C2(n2,k2)

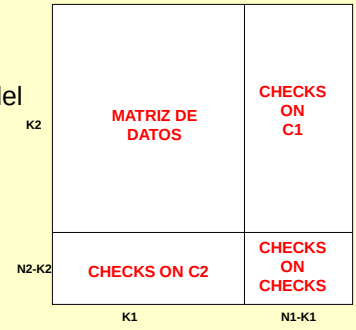
- Podrían usarse más códigos (dimensiones)

Se puede demostrar que la distancia mínima del código producto es el producto de las distancias mínimas de cada código bloque.

Son similares a crucigramas, aplicándose los códigos a cada mensaje

- Cada código a una dimensión del mensaje ordenado en una matriz

Se suelen usar códigos sistemáticos (la matriz de datos no variará)



Códigos producto: Codificación

- Los datos del mensaje se agrupa en matrices de k1xk2 y se calculan bits de redundancia aplicando el C1 por filas y el C2 por columnas (o viceversa)
- Aparecen unos bits de redundancia cruzada (producto) al proteger el C1 los bits (n2-k2) de C2 y viceversa.
- Tras la codificación se transmite por filas o columnas.

Códigos producto: Codificación – ejemplo

Sean los códigos definidos por las matrices

$$G_1 = \begin{bmatrix} 10001 \\ 01001 \\ 00110 \end{bmatrix} \qquad G_2 = \begin{bmatrix} 10001 \\ 01011 \\ 00110 \end{bmatrix}$$

Codificar mediante el código producto C1xC2 la secuencia de información [100100001]

Códigos producto: Codificación – ejemplo - solución

C1(5,3), C2(5,3) => el código producto codifica bloques de 3x3 bits y genera 25 bits (5x5)

X=[100100001] => [100; 100; 001]

$$CP_{xx} = \begin{bmatrix} 100xx \\ 100xx \\ 001xx \\ xxxxx \\ xxxxx \end{bmatrix} \quad CP_{1x} = \begin{bmatrix} 10001 \\ 10001 \\ 00110 \\ xxxxx \\ xxxxx \end{bmatrix} \quad CP_{12} = \begin{bmatrix} 10001 \\ 10001 \\ 00110 \\ 10111 \\ 00000 \end{bmatrix} \quad CP_{x2} = \begin{bmatrix} 100xx \\ 100xx \\ 001xx \\ 101xx \\ 000xx \end{bmatrix} \quad CP_{12} = \begin{bmatrix} 10001 \\ 10001 \\ 00110 \\ 10111 \\ 00000 \end{bmatrix}$$

Cf=[10001;10001;00110:10111;00000] (por filas)

Cc=[11010;00000;00110:00110;11010] (por columnas)

Ejercicio de clase 23 - Códigos producto: Codificación

Sean los códigos definidos por las matrices

$$G_1 = \begin{bmatrix} 101 \\ 011 \end{bmatrix} \qquad G_2 = \begin{bmatrix} 10010 \\ 01001 \\ 00111 \end{bmatrix}$$

Codificar mediante el código producto C1xC2 la secuencia de información [100101]

Ejercicio de clase 23 - Códigos producto: Codificación - solución

Códigos producto: Decodificación

- Tras recibir los datos, se crea la matriz por filas o columnas
- Se decodifica (subóptimamente) iterativamente aplicando en serie cada uno de los códigos
  - Se pueden usar técnicas blandas para mayor flexibilidad en el proceso iterativo
  - Existen algoritmos complejos de decodificación óptima

Códigos producto: Decodificación - ejemplo

Sean los códigos definidos por las matrices

$$G_1 = \begin{bmatrix} 10001 \\ 01001 \\ 00111 \end{bmatrix} \qquad G_2 = \begin{bmatrix} 10001 \\ 01011 \\ 00110 \end{bmatrix}$$

Decodificar mediante el código producto C1xC2 la secuencia recibida [00001 11001 00010 10101 10010] (transmitida por filas).

Comentar los resultados siendo el mensaje transmitido [100100001]

Códigos producto: Decodificación – ejemplo – solución (I)

Siendo X=[100100001] =>
Cf=[10001;10001;00110:10111;10011]

Si Y= [00001;11001;00010:10101;10010] el error ha sido
[10000;01000;00100:00010;00001] (todos los de 1 bit)

C1 y C2 => t=0 => no pueden corregir todos estos errores

Generar AS1 y AS2

Decodificar hasta que no haya cambios

Ver resultados

Códigos producto: Decodificación – ejemplo – solución (II)

Generar AS1 y AS2

Decodificar hasta que no haya cambios

Ver resultados

$$C1=\begin{pmatrix}00000\\00111\\01001\\01110\\10001\\10110\\11000\\11111\end{pmatrix}$$

$$AS1=\begin{pmatrix}00000&00111&01001&01110&10001&10110&11000&11111\\10000&10111&11001&11110&00001&00110&01000&01111\\00100&00011&01101&01010&10101&10010&11100&11011\\00010&00101&01011&01100&10011&10100&11010&11101\end{pmatrix}$$

$$C2=\begin{pmatrix}00000\\00110\\01011\\01101\\10001\\10111\\11010\\11100\end{pmatrix}$$

$$AS2=\begin{pmatrix}00000&00110&01011&01101&10001&10111&11010&11100\\10000&10110&11011&11101&00001&00111&01010&01100\\01000&01110&00011&00101&11001&11111&10010&10100\\00100&00010&01111&01001&10101&10011&11110&11000\end{pmatrix}$$

Códigos producto: Decodificación – ejemplo – solución (III)

Corregimos matriz recibida con AS1 y AS2

Recibida C1: Filas C2:Columnas C1:Filas C2:Columnas C1:Filas

$$\begin{pmatrix}00001\\11001\\00010\\10101\\10010\end{pmatrix} \quad \begin{pmatrix}10001\\01001\\00000\\10001\\10110\end{pmatrix} \quad \begin{pmatrix}10111\\00001\\10000\\10001\\10110\end{pmatrix} \quad \begin{pmatrix}00111\\01001\\00000\\10001\\10110\end{pmatrix} \quad \begin{pmatrix}00111\\10001\\00000\\10001\\10110\end{pmatrix} \quad \begin{pmatrix}00111\\10001\\00000\\10001\\10110\end{pmatrix}$$

Iguales

Transmitida

$$\begin{pmatrix}10001\\10001\\00110\\10111\\10011\end{pmatrix}$$

Secuencia “recibida”: [000.110.000]
Secuencia recuperada: [001.100.000]
Secuencia transmitida: [100.100.001]
De 3 errores se han corregido 1 !!!!
$d_{min1}=2$, $d_{min2}=2$, $d_{minp}=4$
Decodificación “subóptima”

Ejercicio de clase 24 - Códigos producto: Decodificación

Sean los códigos definidos por las matrices

$$G_1 = \begin{bmatrix}101\\011\end{bmatrix} \qquad G_2 = \begin{bmatrix}10010\\01001\\00111\end{bmatrix}$$

Decodificar mediante el código producto C1xC2 la secuencia recibida Y=[11101 01001 00101] (por columnas).

Comentar los resultados siendo el mensaje transmitido [101110]

Ejercicio de clase 24 - Códigos producto: Decodificación– solución (I)

Ejercicio de clase 24 - Códigos producto: Decodificación - solución (II)

Ejercicio de clase 24 - Códigos producto: Decodificación - solución (III)

Ejercicio propuesto 19 - Códigos producto

Sean dos códigos de bloque definidos por sus matrices generatrices G1=[1 0 0 01; 0 1 0 0 1; 0 0 1 1 0] y G2 = [1 0 0 1 1 0 ; 0 1 0 0 1 1 ; 0 0 1 1 0 1].

- Calcular las capacidades detectoras y correctoras de esos códigos, y del código producto resultante de su combinación (C1xC2).
- Codificar para transmisión la secuencia de información [1 0 0 1 0 0 0 0 1 1 1 1 0 1 0 1 1 0] mediante el código producto C1xC2.
- Transmitiendo por filas, sea el patrón de error en el canal [1 0 0 0 0 ; 0 0 0 0 0 ; 0 1 0 0 0; 0 0 0 0 0 ; 0 0 0 0 0; 0 0 0 0 0] y [0 0 0 0 1; 0 1 0 0 0 ; 0 0 0 0 0; 0 0 0 0 0 ; 0 0 0 0 0; 0 0 0 0 0]. Calcular el bloque recibido y proceder a su decodificación.

Códigos producto: comparativa

Del ejemplo

- Siendo X=[100100001] => Cf=[10001;10001;00110:10111;10011]
- Si Y= [00001;11001;00010:10101;10010] el error ha sido [10000;01000;00100:00010;00001] (todos los de 1 bit)
- Secuencia transmitida: [100.100.001]
- Secuencia recuperada CP: [**0**0**1**.100.00**0**] 3 bits erróneos
- Secuencia recuperada C1: [100.**01**0.001] 2 bits erróneos
- Secuencia recuperada C2: [100.100.001] 0 bits erróneos

Pero todo depende del patrón de error

- Para comparativas "serias" en simulaciones hay que reproducir patrones de error (por definición aleatorios)
  - o generadores de números aleatorios con semilla

Códigos concatenados (I)

Consisten en la combinación de dos códigos en serie (en paralelo no mejoran realmente, pero también tienen su utilidad al "ahorrar" tasa binaria)

- Código externo (outer code)
- Código interno (inner code)

Siendo R_{in} la tasa del código interno y R_{out} la del externo, la tasa del código concatenado será $R_{cc}= R_{in} R_{out}$

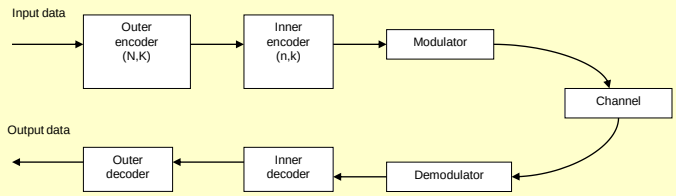
El código interno suele tener mayor impacto en la ejecución global del código

- Si los dos códigos son de bloque, la distancia mínima del concatenado es el producto de las distancias mínimas.

Códigos concatenados (II)

Normalmente el código externo es un código q-ario y el interno interno es un código binario (de bloque o convolucional)

- El externo toma K bloques de k bits y genera N bloques de k bits (símbolos q-arios)
  - Se suelen usar códigos Reed-Solomon (cíclico q-ario)
- El interno toma cada bloque de k bits y genera bloques de n bits, por lo que al modulador le entran bloques de n bits.
  - Se suelen usar códigos convolucionales con decodificación Viterbi blanda.



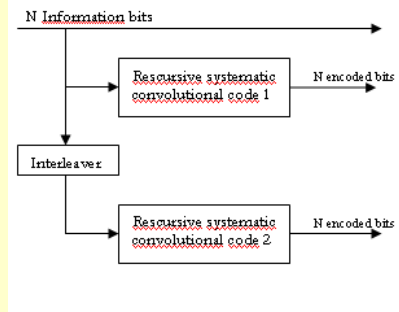
Turbo Códigos: Introducción (I)

Son una clase especial de códigos concatenados (mejoran también en combinación paralela al combinarse con iteraciones con realimentación)

Existe un entrelazador (interleaver) entre los dos códigos

- en serie
- en paralelo.

Entrelazado (pseudo-aletario de varios miles de bits) permite un excelente rendimiento con una baja SNR en códigos cuyas palabras códigos son muy largas



Turbo Códigos: Codificación (I)

La codificación consiste en

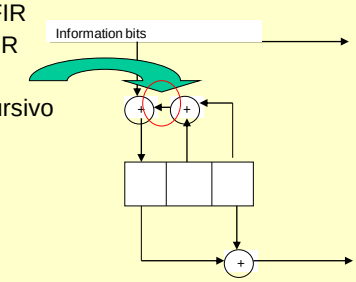
- dos códigos
  - Normalmente se suele usar el mismo código (sistemático)
- un entrelazador de longitud N.

Los códigos suelen ser códigos convolucionales recursivos y sistemáticos (RSCC) de tasa ½

Los códigos convolucionales recursivos ≠ no recursivos

- existencia de realimentación en el desplazamiento de los registros
  - código convolucional no recursivo → FIR
  - código convolucional recursivo → IIR

recursivo



Turbo Códigos: Codificación (II)

Los N bits de información entran en el primer codificador.

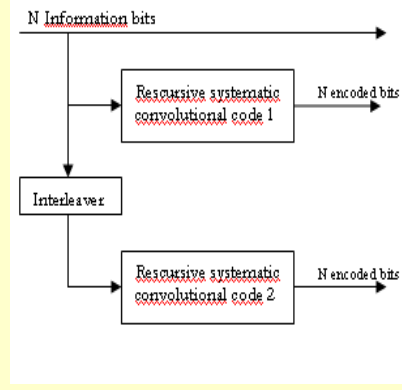
Los mismos bits de información son entrelazados antes de entrar en el segundo codificador

Después de la codificación

- N bits de información
- 2N bits de chequeo de paridad

Un total de 3N bits, son transmitidos al canal

- Tasa → R=1/3



Turbo Códigos: Entrelazador *(interleaver)*

El entrelazado es muy largo
- del orden de miles de bits.

Entrelazado pseudoaleatorio
- funciona bien

Se puede obtener alguna mejora en el código con una elección inteligente del mismo
- Más notable en un entrelazador de longitud corta.

Códigos convolucionales no recursivos
- secuencia de ceros de relleno en un mensaje garantiza que el encoder vuelva al estado cero

Códigos convolucionales recursivos
- volver al estado cero requiere un relleno de la secuencia de información con una secuencia distinta de cero.
- Esto es debido a que en la mayoría de los casos es imposible con un entrelazado devolver dos códigos con el estado todo ceros.

Turbo Códigos: Decodificación (I)

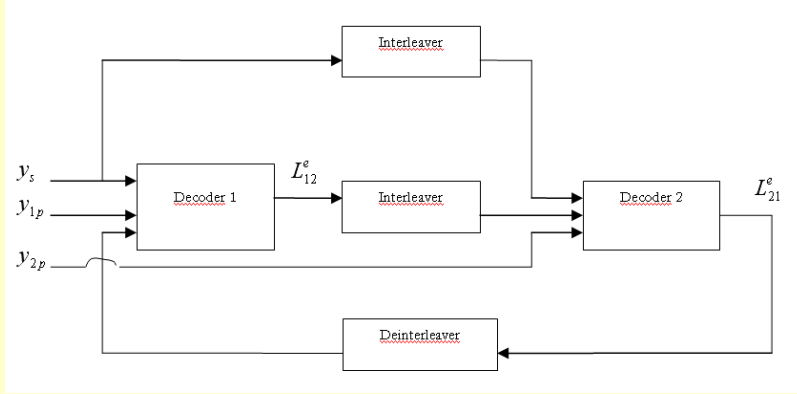
Como está compuesto de 2 códigos → un algoritmo **iterativo** es apropiado para la decodificación

Cualquier método de decodificación que produzca la similitud (*likelihood*) de los bits en la salida puede ser usado en un esquema iterativo de decodificación.
- Uno de estos esquemas de decodificación es el método de decodificación *maximum a posteriori* (MAP) de Bahl, Cocke, Jelinek y Raviv (BCJR) y variaciones de este.
- Otro método popular con una baja complejidad es el algoritmo de Viterbi de salida blanda (Soft-Output Viterbi Algorithm SOVA)

Usando cualquiera de estos dos métodos, las similitudes de los diferentes bits son calculadas y pasadas al segundo decodificador.

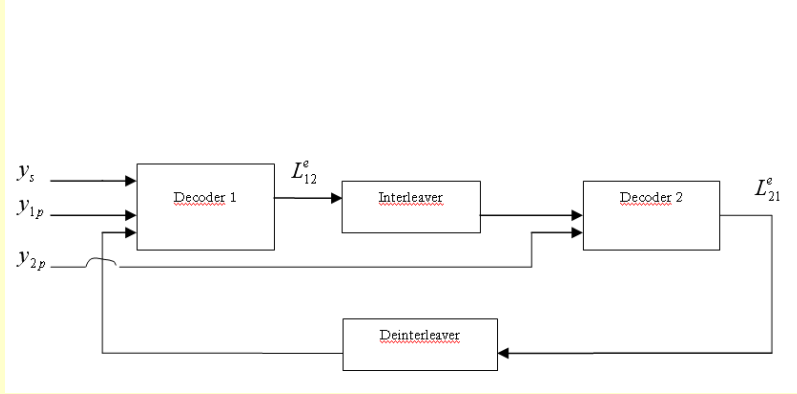
El segundo decodificador, calcula los ratios de similitud y los pasa al primer decodificador; este proceso se repite hasta que la similitud sugiera una alta probabilidad de decodificación correcta para esos bits.

Turbo Códigos: Decodificación (II)



- y_s: bits sistemáticos → información a transmitir
- y_{1p}: bits de paridad para el codificador 1
- y_{2p}: bits de paridad para el codificador 2
- L^e_{12}: Información de similitud del decodificador 1 al 2
- L^e_{21}: Información de similitud del decodificador 1 al 2

Turbo Códigos: Decodificación (III)



- y_s: bits sistemáticos → información a transmitir
- y_{1p}: bits de paridad para el codificador 1
- y_{2p}: bits de paridad para el codificador 2
- L^e_{12}: Información de similitud del decodificador 1 al 2
- L^e_{21}: Información de similitud del decodificador 1 al 2

Turbo Códigos: Rendimiento (I)

Los Turbo Códigos se caracterizan por un excelente rendimiento con una baja SNR.

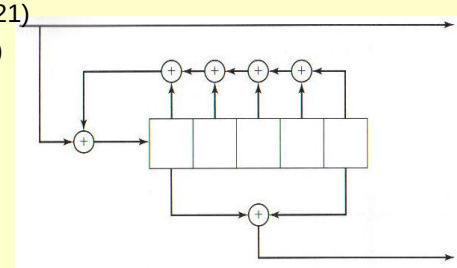
- Es posible acercarse a 0.7dB del límite de Shannon para una baja SNR

El rendimiento mejora con

- el aumento de la longitud del entrelazador
- el número de iteraciones en decodificación.

El turbo código original lo estudió Berrou et al. (1993) usando el siguiente codificador convolucional recursivo y sistemático

- g1 =[1 0 0 0 1] hacia delante (octal21)
- g2 =[1 1 1 1 1] hacia atrás (octal37)
- Código RSCC 21/37
- N = 65536
- R=1/2



Turbo Códigos. Rendimiento (II) *

El resultado del rendimiento del algoritmo de decodificación BCJR

- Después de 18 iteraciones, el rendimiento de este código está a tan sólo 0,7dB del límite de Shannon (Proakis2002)
- La probabilidad de error decrece de forma brusca hasta cierto punto; después de este punto, la probabilidad de error decrece muy despacio.
  - Aunque los turbo códigos tenga un excelente rendimiento, tienen sin embargo, una distancia mínima muy mala.

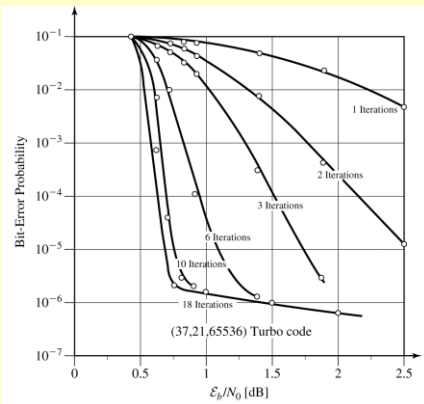


Figure 9.42
The performance plots for the (37,21,65536) turbo code for different number of iterations.

Turbo Códigos: Rendimiento (III) *

Rendimiento bueno vs distancia mínima mala
- el número de caminos con baja distancia es muy pequeño → la multiplicidad de la distancia.

Se puede diseñar el turbo código con una distancia mínima mejor
- un papel mayor de la multiplicidad de la baja distancia

Con baja SNR
- efecto de la multiplicidad es más importante en el rendimiento del código

Con alta SNR
- la distancia mínima juega un papel mayor

De esta manera, el rendimiento del turbo código con alta SNR decrece

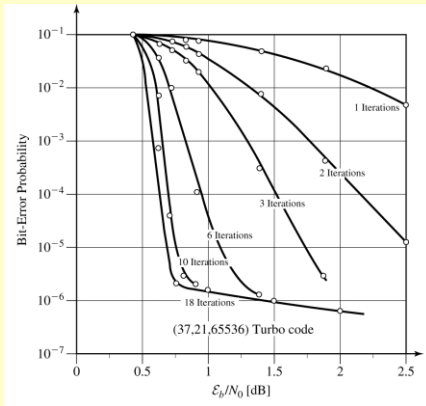


Figure 9.42

The performance plots for the (37,21,65536) turbo code for different number of iterations.

Turbo Códigos: Rendimiento (IV) *

Comparación
- rendimiento del turbo código 37/21
- rendimiento de un código convolucional con tasa ½ y longitud constante igual a 14, usando un decodificador Viterbi de decisión blanda.

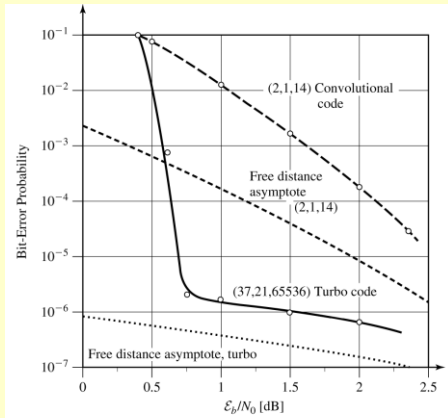


Figure 9.43

Comparison of performance of a turbo code and a convolutional code.

Turbo Códigos: Ejemplos

http://www.comelec.enst.fr/turbocodes/turbo_notes_en.html

- Ejemplo 1
- Ejemplo 2
- Ejemplo 3

Índice

Índice

- Introducción
- ***Códigos de canal***
  - ***Introducción***
  - ***Códigos lineales***
  - ***Códigos cíclicos***
  - ***Códigos convolucionales***
  - ***Códigos basados en combinación***
- Modulación codificada
- Aplicaciones de códigos de canal

Índice

- Introducción
- Códigos de canal
- **Modulación codificada**
  - **Introducción**
  - **Modulación Codificada Trellis**
- Aplicaciones de códigos de canal

Modulación Codificada: Introducción (I)

El control de errores mediante codificación de canal

- Códigos de bloque
- Códigos convolucionales
- Códigos combinados

implica

- (+) Una reducción en la relación S/N requerida
  - Ganancia de Codificación Gc (dB).
- (-) Un incremento del ancho de banda (por tasa binaria)
  - Se incrementa (a igualdad de sistema de modulación) por un factor de 1/Rc.

    Rc=k/n: tasa del codificador de canal

Todo esto es aplicable en sistemas con suficiente ancho de banda (comunicaciones espaciales) pero no es sistemas con ancho de banda restringido (línea telefónica).

Modulación Codificada: Introducción (II)

Codificación y Modulación:

- La tasa aumenta 1/Rc por lo que aumenta el ancho de banda
- Si queremos que no aumente hay que pasar a un tipo distinto de modulación
  - Más eficiente en W => más dimensiones
  - Más dimensiones implica menor distancia euclídea (más Pe), pero se compensa por el aumento de la distancia Hamming => el sistema completo mejora

Supongamos un sistema con modulación (sin codificación) 4-PSK, consigue R/W=2 bits/seg/Hz para una probabilidad de error Pb = 10^{-6}.

- Para esta Pb la relación S/N por bit es 10,5 dB.
- Podemos reducir la relación S/N utilizando señales codificadas pero sin que aumente el ancho de banda.

Elegimos un código con Rc = 2/3, debe ser acompañado de un incremento en el número de símbolos.

- pasamos de 4 símbolos (2 bits/símbolo) a 8 (3bits/símbolo).
- requiere aumentar la S/N por bit en 4dB para mantener la Pb.
- Para proporcionar beneficio, la Gc debe compensar esos 4dB, más una ganancia neta que justifique el complicar el sistema de transmisión (añadir codificación de canal y “complicar” la modulación).

Modulación Codificada: Introducción (III)

Si los procesos de modulación y codificación se realizan por separado:

- se requieren códigos muy potentes, con L grande, para compensar Gc.

Si la modulación es parte integral de la codificación:

- se diseña conjuntamente para incrementar la distancia Euclídea mínima entre pares de señales codificadas.
- se alcanzan altos valores Gc.

La clave es diseñar métodos efectivos para mapear los bits codificados de la señal.

- maximizando la mínima distancia Euclídea entre símbolos.
- En una opción más sistemática que la codificación Gray (minimizar el error entre señales adyacentes)

La técnica más utilizada es la Modulación Codificada Trellis (TCM –Trellis Coded Modulation-)

- se basa en la partición de constelaciones

Modulación Codificada: TCM (Trellis Coded Modulation)

La Modulación Codificada Trellis es un método sencillo para obtener esquemas de modulación codificada con un buen rendimiento.

- Parte de la partición de constelaciones
  - Se basa en la idea de “mapeo de particiones de conjunto” [Ungerboek 1982].
  - El objetivo es separar las constelaciones en diversos niveles de partición en base a la máxima distancia entre las señales de cada nivel.
  - La codificación de canal seleccionará una partición (en función de sus necesidades de tasa) que al tener sus señales más alejadas dará lugar a una mejor decisión entre la familia completa de la constelación.
- Se puede usar con cualquier tipo de código de canal, pero suelen usarse códigos convolucionales (con decodificación Viterbi blanda –sobre distancias euclídeas-).

Modulación Codificada: TCM - Partición 8-PSK

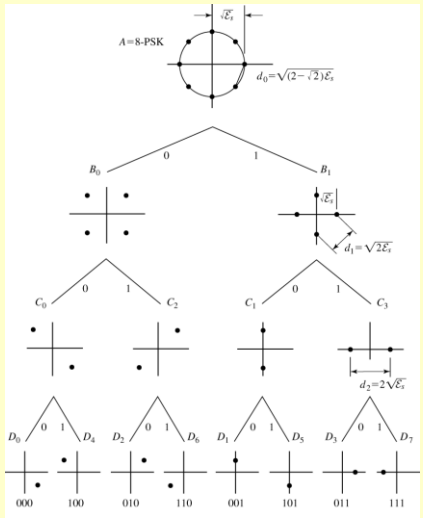


Figure 9.45

Partitioning of an 8-PSK constellation.

A cada nivel de partición aumenta la distancia entre señales

- $d_0^2 = 0.585\ \varepsilon$
- $d_1^2 = 2\ \varepsilon$
- $d_2^2 = 4\ \varepsilon$

Codificación

- La secuencia de bits asignada a las ramas izquierda y derecha produce etiquetas de 3 bits para los puntos de la constelación.
- La asignación de bits no es importante hasta el momento de construir el codificador.

Modulación Codificada: TCM - Partición 16-QAM

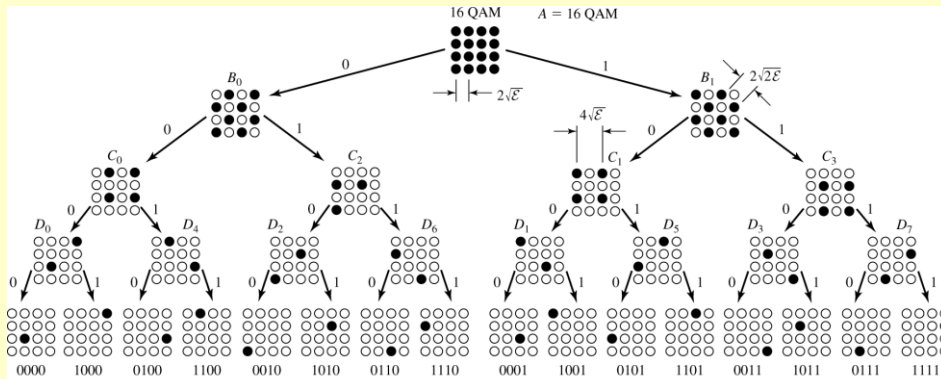


Figure 9.46

Set partitioning of a 16-QAM constellation.

A cada nivel de partición, la distancia entre señales aumenta en $2^{1/2}$

- $d_i^2 = 2\ d_{i-1}^2$

Modulación Codificada: TCM - Particiones

En los dos ejemplos anteriores, se ha realizado una partición hasta el límite, donde cada subconjunto contiene sólo un punto. En general esto puede no ser necesario.

El grado en que se particiona la señal depende de las características del código.

Modulación Codificada: TCM – "Codificación"

Un grupo de m =k1 + k2 bits se separa en dos grupos de k1 y k2 bits.

- Los k1 bits se codifican en n bits.
- Los n bits codificados se usan para seleccionar uno de los 2^n posibles subconjuntos en las particiones.

Los k2 restantes se dejan sin codificación de canal

- se utilizan para seleccionar uno de los 2^{k2} puntos de la señal en cada subconjunto.

Si k2=0 los m bits se codifican con redundancia.

Por lo tanto hace falta una constelación capaz de producir 2^{k2} puntos en un nivel que tenga 2^n elementos: $2^{(n+k2)}$.

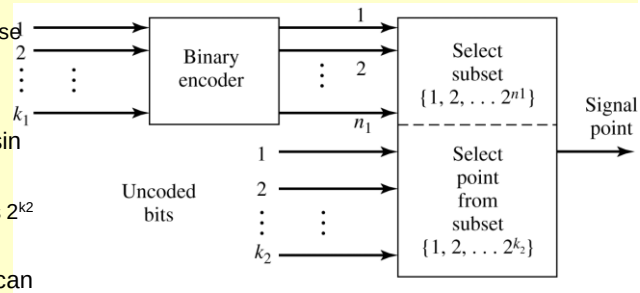


Figure 9.47

The block diagram of a coded-modulation system.

Modulación Codificada: TCM – "Codificación" - ejemplo (I)

Ungerboeck [1982] demostró que con n1=k1+1 y k2=1 y un simple código convolucional se obtiene un sistema TCM con Gc entre 3 y 6 dB.

Para k1=1

- Constelación 8 puntos: 8-PSK
  - 4 elementos en la partición
  - 2 puntos en cada elemento
- CC: valdría cualquier de Rc=1/2

Modulación Codificada: TCM – "Codificación" – ejemplo (II)

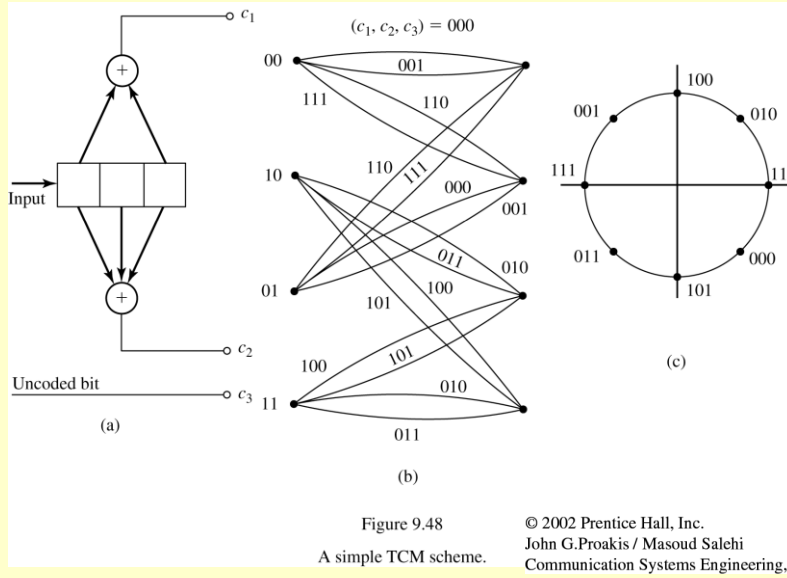


Figure 9.48
A simple TCM scheme.

Modulación Codificada: TCM – "Codificación" – ejemplo (III)

El Trellis de la figura anterior es el Trellis del convolucional, siendo cada pareja de ramas paralelas transiciones entre el mismo elemento de la partición (-estado- definido por el bit k1) y duplicada por el bit k2.

Queda por describir como se asigna cada código del Trellis a los puntos de la constelación:

- Esta asignación se realiza mediante reglas obtenidas por simulaciones y heurística (para cada caso)
  - Separar al máximo las transiciones paralelas (nivel de puntos)
  - Asignar el mismo elemento del nivel superior a las transiciones que tengan el mismo destino (nivel de partición) [{C0=00,C2=10},{C1=01,C3=11}] -> [B0=0,B1=1] (ver figura partición 8-PSK, la asignación binaria se hace de derecha a izquierda empezando por el nivel superior)

Si se seleccionase la 16-QAM habría varias posibilidades:

- Desechando siempre señales que nunca se transmitirían (lo que sería subóptimo desde el punto de vista de rendimiento de la modulación, pero tendría un mejor comportamiento como código de canal)

Modulación Codificada: TCM - Rendimiento

Para ver el rendimiento de una TCM se hace uso de una distancia mínima, denominada en TCM Distancia Euclídea Libre (D_{fed})

- Esta distancia es la mínima (euclídea) entre dos caminos que parten de un mismo nodo y vuelven a un mismo nodo.
- Si bien pueden haber otros caminos, generalmente la mínima distancia se da entre dos caminos paralelos, de forma que D_{fed} suele ser la distancia entre dos caminos paralelos.
  - En el ejemplo visto $D_{fed}=d_2$ ($d_2^2= 4\ \varepsilon$)
  - $Gc= d_{uncoded}^{2/} d_2^2 = 2 \sim 3$ dB (a igual W, pero con mayor complejidad)
- A mayor número de estados mayor Gc
  - 8-256 estados => 3.6-5.75 dB

A nivel W (B), siendo m=k1+k2

- $B_{sc}=B_{k1}+B_{k2}$
- $B_{cc}=B_{sc}/R_c>B_{sc}$
- $B_{TCM}>B_{sc}$, $B_{TCM}<B_{cc}$, $B_{TCM}=B_{k1}/R_c+B_{k2}$

Modulación Codificada: TCM – "Decodificación"

La demodulación-codificación se realiza en dos pasos

- En primer lugar hay que buscar el punto más probable en cada nivel de partición del TCM (el más cercano en distancia euclídea al recibido)
  - Este paso se llama "subset decoding"
  - Para cada transición paralela existirá por tanto solamente una rama
    Discrimino entre ramas paralelas
  - y con su distancia euclídea asociada
- En el segundo paso, con la información anterior se busca en camino de máxima verosimilitud mediante el algoritmo de Viterbi

Índice

Índice

- Introducción
- Códigos de canal
- Modulación codificada
- **Aplicaciones de códigos de canal**

Aplicaciones de códigos de canal

Codificación sin W limitado (e.g., espacio)

- No hay límite en W pero hay mucho error por atenuación y no puedo aumentar potencia (e.g., satélites)
- Interesa mucha protección a costa de W
- Códigos lineales, convolucionales, combinados, …

Codificación con W limitado (e.g., cable telefónico)

- Interesa menor W aún a costa de mayor complejidad (vs aumentar potencia => diafonía)
- TCM

Sistemas de almacenamiento (e.g., CDs, DVDs) [y ruido industral]

- Errores a ráfagas (e.g., fallo físico, “dedazos”, rayazos, …)
- Reed-Solomon

Índice

- Introducción
- Códigos de canal
- Modulación codificada
- ***Aplicaciones de códigos de canal***

Índice

Bibliografía

- John G. Proakis, Masoud Salehi, “Communication Systems Engineering”, 2nd ed., Prentice Hall, 2002.
- *Bernard Sklar, “Digital Communication”, 2nd ed., Prentice Hall, 2001*
- *Shu Lin,Daniel J. Costello, “Error Control Coding”, 2nd ed., Prentice-Hall, 2004*
- http://www.comelec.enst.fr/turbocodes/turbo_notes_en.html