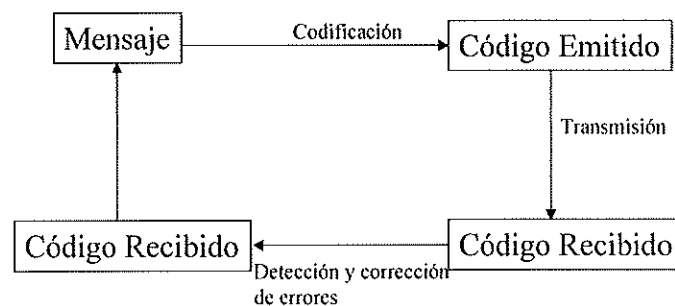


Códigos Autocorrectores

- * Permiten detectar y corregir errores.
- * Agregan redundancia a la información.
- * Los errores pueden producirse tanto en la información como en la redundancia.
- * Utilizados en comunicación de datos, grabacion de CD's, etc.

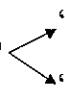
Códigos Autocorrectores

Esquema Básico



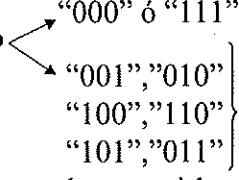
Códigos Autocorrectores

Primera aproximación - Códigos de repetición

- Mensaje : 1 bit "0" ó "1"
- Código enviado : Mensaje duplicado "00" ó "11"
- Código recibido 
 - "00" ó "11" Mensaje recibido correctamente
 - "10" ó "01" Mensaje recibido con errores
- Detecta el error pero no puede corregirlo

Códigos Autocorrectores

Primera aproximación - Códigos de repetición doble

- Mensaje : 1 bit "0" ó "1"
- Código enviado : Mensaje triplicado "000" ó "111"
- Código recibido 
 - "000" ó "111" Mensaje recibido correctamente
 - "001", "010", "100", "110", "101", "011" Mensaje recibido con errores
- Detecta 1 error y puede corregirlo
- Decide por el "más cercano"

Códigos Autocorrectores

Códigos de repetición para detectar n errores

- Mensaje : 1 bit "0" ó "1"
- Código enviado : Mensaje repetido $2*n$ veces (longitud $2*n+1$)
- Detecta n errores y puede corregirlos

Códigos Autocorrectores

Definiciones

- Distancia entre 2 códigos : Cantidad de elementos (bits) distintos
- Código de longitud " n " : subconjunto de posibles "strings" distintos formados tomando " n " símbolos de un alfabeto.
- Un código se define por una n -upla de 3 elementos : (n,M,d) :
 - \Rightarrow " n " : longitud de los códigos.
 - \Rightarrow " M " : cantidad de códigos posibles.
 - \Rightarrow " d " : distancia mínima entre códigos.

Nuestros ejemplos son :

Repetición simple : $(2,2,2)$

Repetición doble : $(3,2,3)$

Códigos Autocorrectores

Objetivos para crear un buen código :

- Minimizar "n" para transmitir más rápido y ocupar menos lugar
- Maximizar "M" para ganar eficiencia
- Maximizar "d" para ganar confiabilidad

Códigos Autocorrectores

Decodificación :

- Si el canal tiene una confiabilidad mayor al 50%, tomamos el código más cercano al recibido
- Si el canal tiene una confiabilidad menor al 50%, tomamos el código más cercano al NOT del recibido

Códigos Autocorrectores

Definición :

Un código corrige “e” errores si, cuando ocurren a lo sumo “e” errores, se puede recuperar el mensaje original

Códigos Autocorrectores

Un código de distancia mínima “d” :

- *DETECTA a lo sumo “ $d/2$ ” errores*
- *CORRIGE a lo sumo “ $(d/2)-1$ ” errores*

Códigos de Hamming

Control de paridad :

Para DETECTAR errores en una tira de bits, alcanza con agregar 1 bit con un valor tal que, la cantidad de bits encendidos sea, por ejemplo, par. Este bit se conoce como bit de paridad

Códigos de Hamming :

La idea es hacer control de paridad sobre grupos de bits para determinar cual de ellos fue cambiado, agregando la mínima cantidad de bits.

Códigos de Hamming

Agregando 3 bits (A,B,C)

Combinaciones de A,B,C	Bits Protegidos
• A-B	→ bit 0
• A-C	→ bit 1
• B-C	→ bit 2
• A-B-C	→ bit 3

El código generado será de 7 bits (0,1,2,3,A,B,C)

Códigos de Hamming

Por cada bit redundante tenemos un conjunto :

$$A = \{0,1,3,A\}$$

$$B = \{0,2,3,B\}$$

$$C = \{1,2,3,C\}$$

Luego utilizamos cada bit redundante (A,B,C) bit de paridad del conjunto correspondiente.

Cualquier cambio en el valor de 1 bit afectará a todos los conjuntos en los que aparece. La intersección de los conjuntos indica el bit errado, permitiendo su corrección

Códigos de Hamming

Ejemplo :

Para proteger el mensaje '0110', los conjuntos son :

$$A = \{0,1,3,A\} = \{0,1,0,A\} \Rightarrow A=1$$

$$B = \{0,2,3,B\} = \{0,1,0,B\} \Rightarrow B=1$$

$$C = \{1,2,3,C\} = \{1,1,0,C\} \Rightarrow C=0$$

El código a enviar será : '0110110'

Códigos de Hamming

Ejemplo (Continuación):

Si el código recibido fuera : '0100110'

Calculamos los conjuntos nuevamente :

$A = \{0,1,3,A\} = \{0,1,0,1\}$ Ok

$B = \{0,2,3,B\} = \{0,0,0,1\}$ Mal

$C = \{1,2,3,C\} = \{0,1,0,0\}$ Mal

Entonces el bit cambiado es el protegido por B-C

Si se cambiara un bit redundante, solo afectará al conjunto que protege.

Códigos de Hamming

Agregando 4 bits (A,B,C,D)

Combinaciones de A,B,C,D	Bits Protegidos	Combinaciones de A,B,C,D	Bits Protegidos
• A-B	→ bit 0	• A-B-C	→ bit 6
• A-C	→ bit 1	• A-B-D	→ bit 7
• A-D	→ bit 2	• A-C-D	→ bit 8
• B-C	→ bit 3	• B-C-D	→ bit 9
• B-D	→ bit 4	• A-B-C-D	→ bit 10
• C-D	→ bit 5		

El código generado será de 15 bits (0,1,2,3,4,5,6,7,8,9,10,A,B,C,D)

Códigos de Hamming

Según la definición de códigos los Códigos de Hamming son :

Agregando 3 bits : (7, 255, 3)

Agregando 4 bits : (15, 32767, 3)

En ambos casos la distancia mínima es 3, ya cualquier cambio en los bits originales cambiará al menos 2 bits redundantes.

Por lo tanto ambos códigos detectan y corrigen 1 error

Códigos de Hamming

Implementación (Para códigos de 7 bits)

Numeremos los bit del código en binario :

bit 1 : 001	==> bit redundante A
bit 2 : 010	==> bit redundante B
bit 3 : 011	==> bit original 0
bit 4 : 100	==> bit redundante C
bit 5 : 101	==> bit original 1
bit 6 : 110	==> bit original 2
bit 7 : 111	==> bit original 3

Códigos de Hamming

Generalizando :

Si agregamos K bits redundantes

Los códigos miden $2^K - 1$ bits

Protegen $2^K - 1 - K$ bits

Son códigos $(2^K - 1, 2^K - 1 - K, 3)$

Interpretación Geométrica

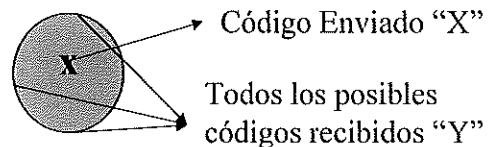
Sea :

K : Cantidad de bits a transmitir

R : Cantidad de bits redundantes agregados

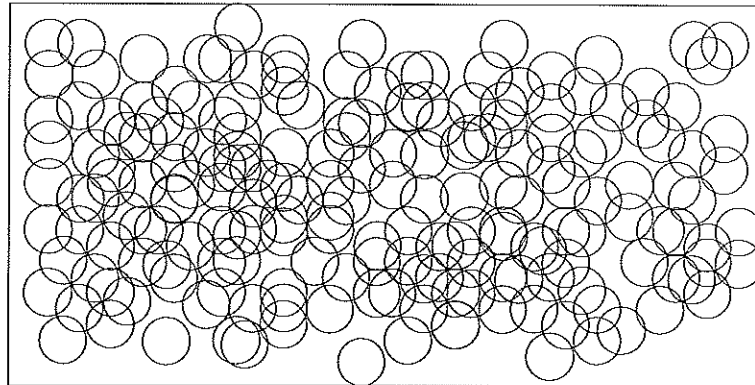
N : Cantidad de bits del código ($N = R + K$)

Por cada código que enviemos, existirá un conjunto de posibles códigos a una distancia 1 del emitido. Podríamos graficarlo de la siguiente forma :



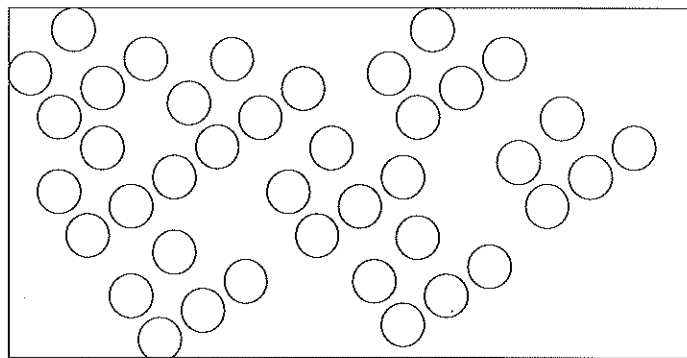
Interpretación Geométrica

Si graficamos todo los posibles códigos "X" con sus respectivos "Y" obtendremos un espacio de la siguiente forma :



Interpretación Geométrica

Si tomamos solo la cantidad de "X" necesarios para representar los $2^K - 1$ mensajes posibles, y eligiendo los códigos de forma tal que no se produzcan intersecciones, debemos obtener un espacio de la siguiente forma :



Códigos Perfectos

Sea :

K : Cantidad de bits a transmitir

R : Cantidad de bits redundantes agregados

Existen :

- $2^{(R+K)}$ códigos posibles.
- 2^K mensajes.
- 2^K códigos utilizados para representar mensajes.
- $2^K * (R+K)$ códigos a distancia 1 de un código que representa a un mensaje

Códigos Perfectos

Un Código es Perfecto si cumple las siguientes propiedades:

a) $2^{(R+K)} = 2^K + 2^K * (R+K)$

b) El código corrige 1 bit de cualquier mensaje

(La propiedad a) es conocida como “Propiedad de empaquetamiento del espacio” y la b) como “Propiedad de códigos disjuntos)

Códigos Perfectos

En palabras :

“Un código es perfecto si todos sus códigos son disjuntos y además empaquetan el espacio”

En otras palabras :

“Un código es perfecto si dado un código cualquiera, o bien representa un mensaje o bien se encuentra a una distancia 1 de un código que representa a un mensaje”

Códigos Perfectos

Veamos si el código de repetición doble es un código perfecto:

Longitud del mensaje $(k) = 1$

Longitud del código $(r) = 3$

Ya vimos que cumple la propiedad (b), veamos la propiedad (a)

$$2^{(R+K)} = 2^K + 2^{K*}(R+K)$$

$$2^{(3+1)} = 2^1 + 2^1*(3+1)$$

$$2^4 = 2 + 2*4$$

$$8 = 8$$

Por lo tanto es perfecto.

En general los códigos de repetición de longitud impar son perfectos.

Códigos Perfectos

Veamos si los códigos de Hamming son códigos perfectos:

Ya vimos que corrigen 1 error, por lo que cumplen la propiedad (b), veamos si cumplen la propiedad (a) :

Para comenzar, si agregamos R bits, protegemos $2^{R-1}-R$ bits, que es la longitud de los mensajes a emitir, por lo tanto:

$$K=2^{R-1}-R$$

Códigos Perfectos

(continuación)

$$2^{(R+K)} = 2^K + 2^{K*(R+K)}$$

Reemplazo con $K=2^{R-1}-R$

$$2^{(R+2^{R-1}-R)} = 2^{(2^{R-1}-R)} + 2^{(2^{R-1}-R) * (R+2^{R-1}-R)}$$

$$2^{(2^{R-1})} = 2^{(2^{R-1}-R)} + 2^{(2^{R-1}-R) * (2^{R-1})}$$

$$2^{(2^{R-1})} = 2^{(2^{R-1}-R)} + 2^{(2^{R-1}-R+R)} - 2^{(2^{R-1}-R)}$$

$$2^{(2^{R-1})} = 2^{(2^{R-1}-R)} + 2^{(2^{R-1})} - 2^{(2^{R-1}-R)}$$

$$2^{(2^{R-1})} = 2^{(2^{R-1})}$$

Por lo tanto son códigos perfectos.