Tecnología Digital IV: Redes de Computadoras

Clase 25: Introducción a la Teoría de la Información (Parte 1)

Emmanuel Iarussi & Lucio Santi

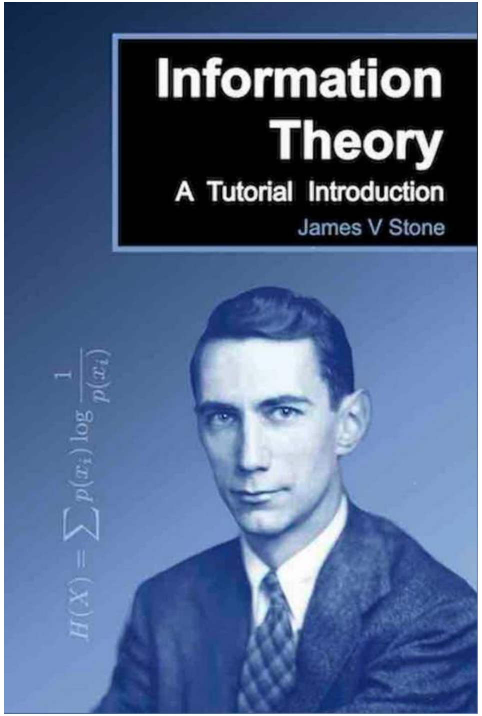
Licenciatura en Tecnología Digital
Universidad Torcuato Di Tella

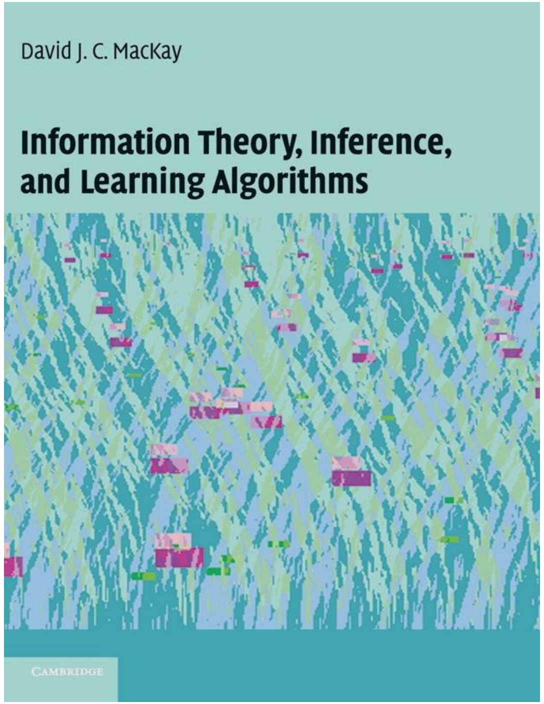
12 de noviembre de 2025

¿Qué es la Teoría de la Información?

- Teoría matemática de la comunicación.
- Estudia la cuantificación, almacenamiento y transmisión de la información.
- Parte integral de la teoría de la probabilidad.
- Desarrollada por Claude Shannon en la década de los 40.

Bibliografía





https://bit.ly/Libro_Mackay

¿En dónde se usa la Teoría de la Información?

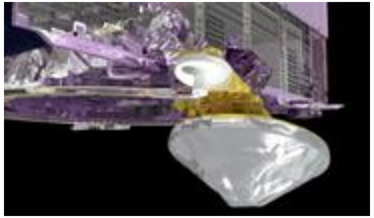
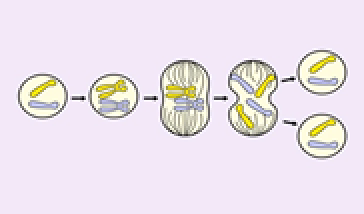
- Diseño de circuitos digitales y computadoras
- Dispositivos de transmisión de datos, Internet
- Métodos de compresión de datos
- Criptografía
- Inferencia estadística, reconocimiento de patrones, aprendizaje automático (*IA* y *Machine Learning*).
- Audio e imágenes
- Física, biología, lingüística, economía, neurociencias, etc.

Problemas de la comunicación

- ¿Se puede ***medir*** la información? ¿Cómo? ¿Cuánta información contiene un mensaje?
- ¿Cuánta información se puede transmitir en un determinado canal?
- Si el mensaje es más grande que la cantidad de información que contiene, ¿se puede representar de otra manera para reducir su tamaño?
- ¿Se puede medir el ***ruido***? ¿Es posible reducir su impacto sobre el mensaje?

Ejemplos de comunicación ruidosa

- Línea telefónica
- El enlace de comunicación de radio con el telescopio Webb
- Células en reproducción: el ADN en las células hijas contiene información de las madres
- Un disco rígido magnético


Repaso de probabilidad

- **Probabilidad**: *noción informal, basada en la cantidad de veces que ocurre un evento particular.*

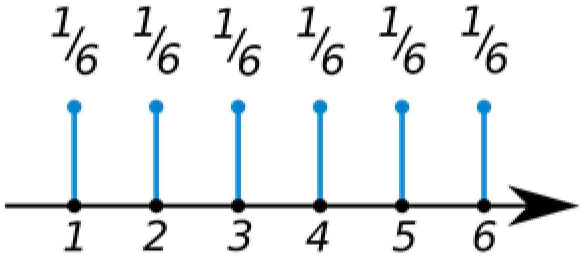
$$p(evento) = \frac{\#casos\ favorables}{\#casos\ posibles}$$

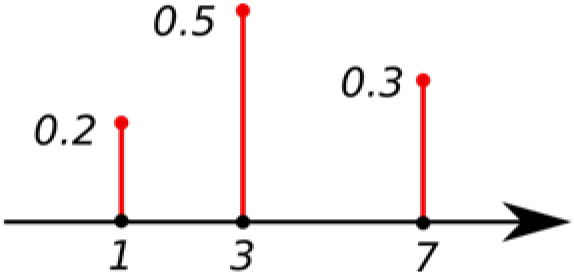
Repaso de probabilidad

- **Variable Discreta**: *variable que solo puede tomar valores dentro de un conjunto de elementos claramente diferenciados entre sí (por ejemplo, una lista de enteros). No acepta cualquier valor: únicamente aquellos que pertenecen al conjunto.*

Repaso de probabilidad

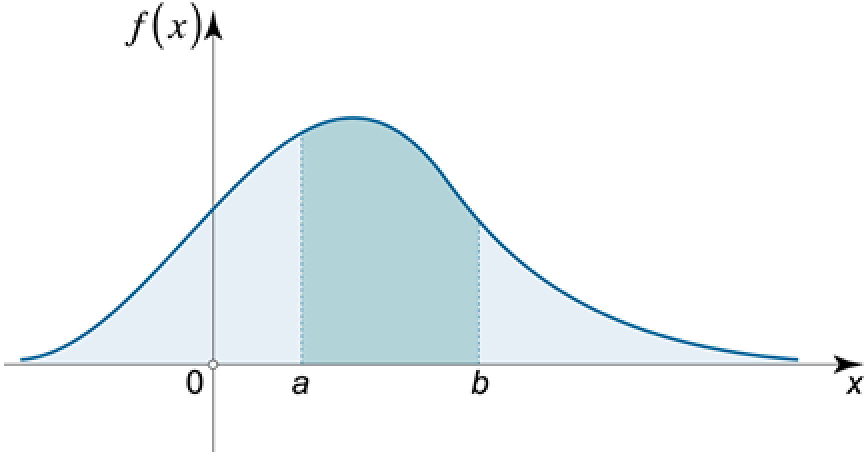
- **Variable Discreta**: *variable que solo puede tomar valores dentro de un conjunto de elementos claramente diferenciados entre sí (por ejemplo, una lista de enteros). No acepta cualquier valor: únicamente aquellos que pertenecen al conjunto.*
- **Función de probabilidad**: *distribución de los valores de probabilidad de una variable discreta.*





Repaso de probabilidad

- **Variable Continua**: *variable que potencialmente puede asumir valores en un intervalo de números reales. Por ejemplo, la estatura de las personas.*
- **Función de densidad**: *distribución de los valores de probabilidad de una variable continua.*



Repaso de probabilidad

- **Variable Aleatoria**: *refiere a un tipo especial de cantidad cuyo valor no es fijo (o es incierto): puede tomar diferentes valores. Es una función que asigna un valor al resultado de un experimento aleatorio. Las variables aleatorias pueden ser discretas o continuas.*

Repaso de probabilidad

- **Variable Aleatoria**: *refiere a un tipo especial de cantidad cuyo valor no es fijo (o es incierto): puede tomar diferentes valores. Es una función que asigna un valor al resultado de un experimento aleatorio. Las variables aleatorias pueden ser discretas o continuas.*
- **Experimento aleatorio:** *es la reproducción controlada de un fenómeno sobre el cual rige algún tipo de incertidumbre acerca del resultado que se obtendrá (tirar un dado, medir una persona que pasa por la calle, sacar una bolilla de lotería, etc).*

Ejemplo: arrojar una moneda

- Una variable aleatoria toma como argumento las posibles salidas del experimento, y les asigna un valor:

$$\underbrace{\begin{array}{l} x_z = sale\ cruz \\ x_c = sale\ cara \end{array}}_{alfabeto} \qquad \underbrace{X}_{variable\ aleatoria}(x_z) = 0, \quad X(x_c) = 1.$$

Alfabeto binario: Contiene dos símbolos, 0 y 1. Este alfabeto es fundamental en la informática y en la transmisión de datos digitales.
Alfabeto ASCII: Consiste en 128 caracteres, incluyendo letras mayúsculas y minúsculas, números, y caracteres especiales.

En teoría de la información, el término "alfabeto" se refiere a un conjunto finito de símbolos que se utilizan para construir mensajes o secuencias de datos. Estos símbolos pueden ser letras, números, caracteres especiales o cualquier otro tipo de símbolo que se utilice en un sistema de comunicación.

Ejemplo: arrojar una moneda

- Una variable aleatoria toma como argumento las posibles salidas del experimento, y les asigna un valor:

$$X = \begin{cases} 0, & si\ sale\ cara, \\ 1, & si\ sale\ cruz. \end{cases}$$

Ejemplo: arrojar un dado

- Definimos una variable aleatoria que vale 0 cuando obtenemos un resultado impar, y 1 cuando es par.

$$X = \begin{cases} 0, & si\ sale\ 1, 3\ o\ 5, \\ 1, & si\ sale\ 2, 4\ o\ 6. \end{cases}$$

Número de resultados = 6 *Número de Valores de resultado = 2*

Variables aleatorias y probabilidades

- En muchos experimentos, los resultados tienen diferentes probabilidades *p*:

$$p(X) = \{p(x_z), p(x_c)\}$$

Variables aleatorias y probabilidades

- En muchos experimentos, los resultados tienen diferentes probabilidades *p*:

$$p(X) = \{p(x_z), p(x_c)\}$$

$$= \{0.5, 0.5\}$$ *moneda "justa"*

Variables aleatorias y probabilidades

- En muchos experimentos, los resultados tienen diferentes probabilidades *p*:

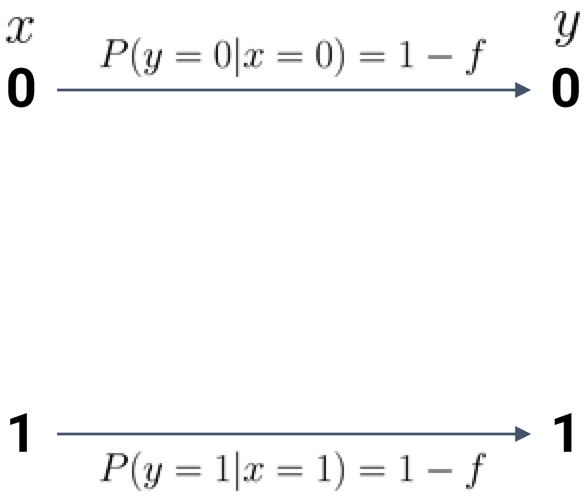
$$p(X) = \{p(x_z), p(x_c)\}$$

$$= \{0.9, 0.1\}$$

- *moneda "cargada"*
- *Caja con bolillas negras y blancas*

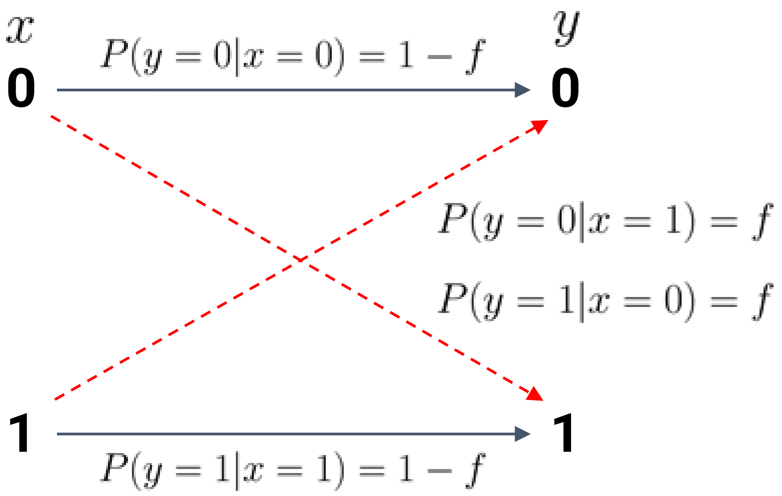
Ejemplo: Canal Binario Simétrico (BSC)

- Supongamos un canal que transmite correctamente cada bit con una probabilidad *(1-f)*



Ejemplo: Canal Binario Simétrico (BSC)

- Supongamos un canal que transmite correctamente cada bit con una probabilidad *(1-f)* e incorrectamente con probabilidad *f*.



Ejemplo: Canal Binario Simétrico (BSC)

- Supongamos un canal que transmite correctamente cada bit con una probabilidad *(1-f)* e incorrectamente con probabilidad *f*.

$$\xrightarrow{\textbf{BSC}}$$

$$P(y=0|x=0)=1-f$$
$$P(y=1|x=0)=f$$
$$P(y=0|x=1)=f$$
$$P(y=1|x=1)=1-f$$

Ejemplo: Canal Binario Simétrico (BSC)

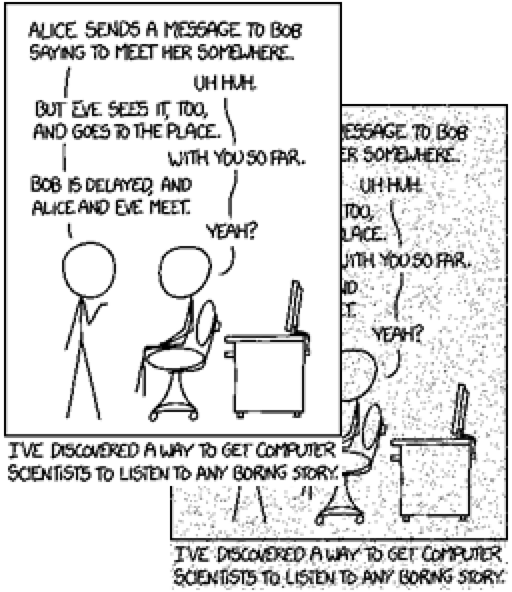
- Supongamos un canal que transmite correctamente cada bit con una probabilidad *(1-f)* e incorrectamente con probabilidad *f*.

BSC →

$$P(y=0|x=0)=1-f$$
$$P(y=1|x=0)=f$$
$$P(y=0|x=1)=f$$
$$P(y=1|x=1)=1-f$$

Ejemplo: Canal Binario Simétrico (BSC)

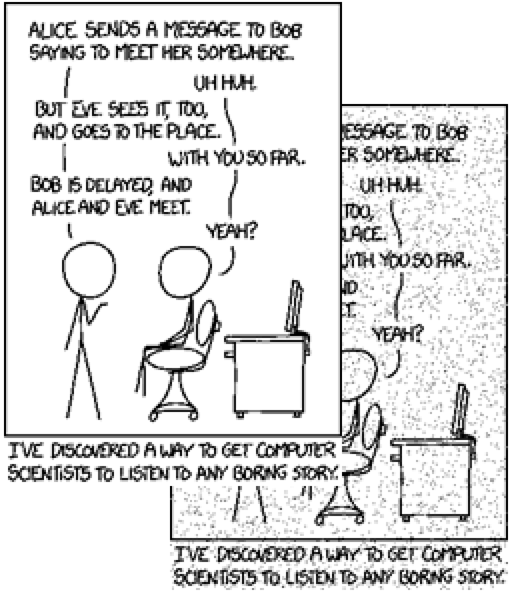
Soluciones:



Ejemplo: Canal Binario Simétrico (BSC)

Soluciones:

- Física: cambiar los componentes por otros más confiables / mejorar la tecnología de comunicación (hardware)

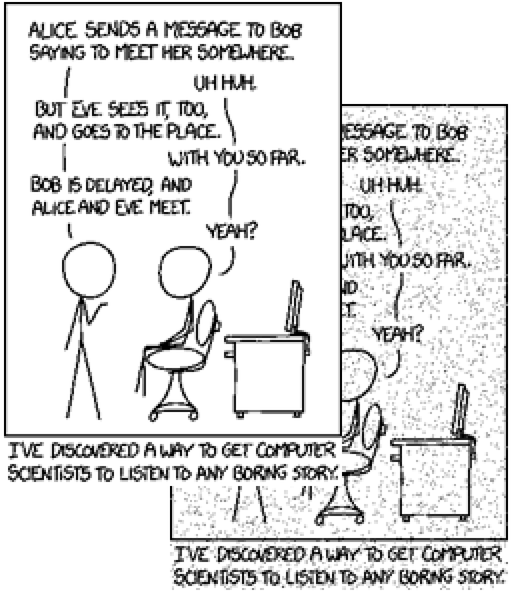


Ejemplo: Canal Binario Simétrico (BSC)

Soluciones:

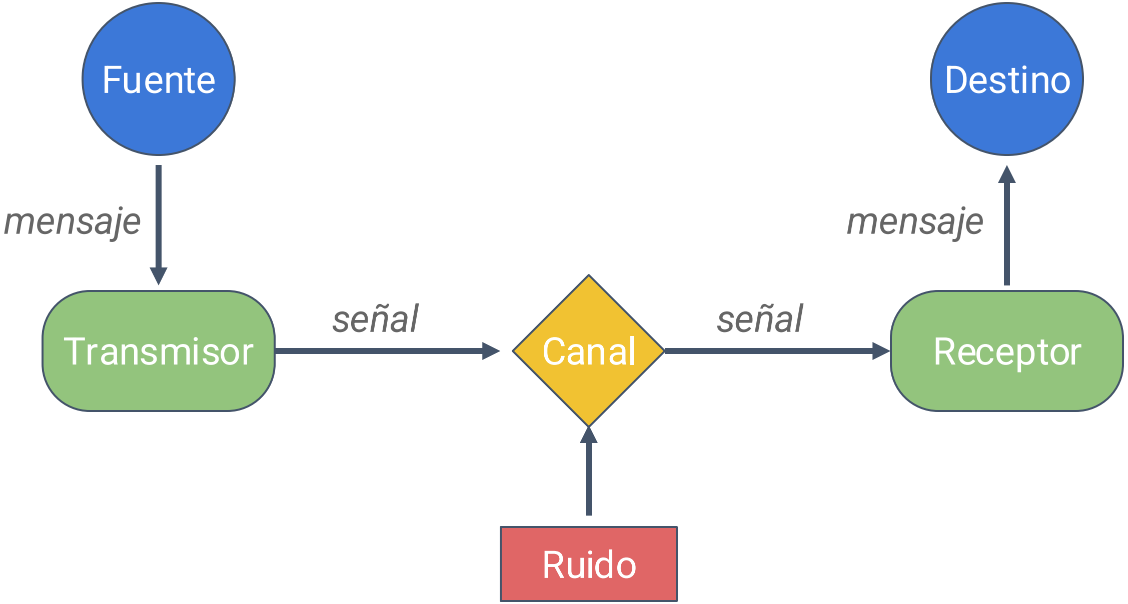
- Física: cambiar los componentes por otros más confiables / mejorar la tecnología de comunicación (hardware)

- Lógica: aceptar el canal ruidoso, detectar y corregir errores utilizando redundancia



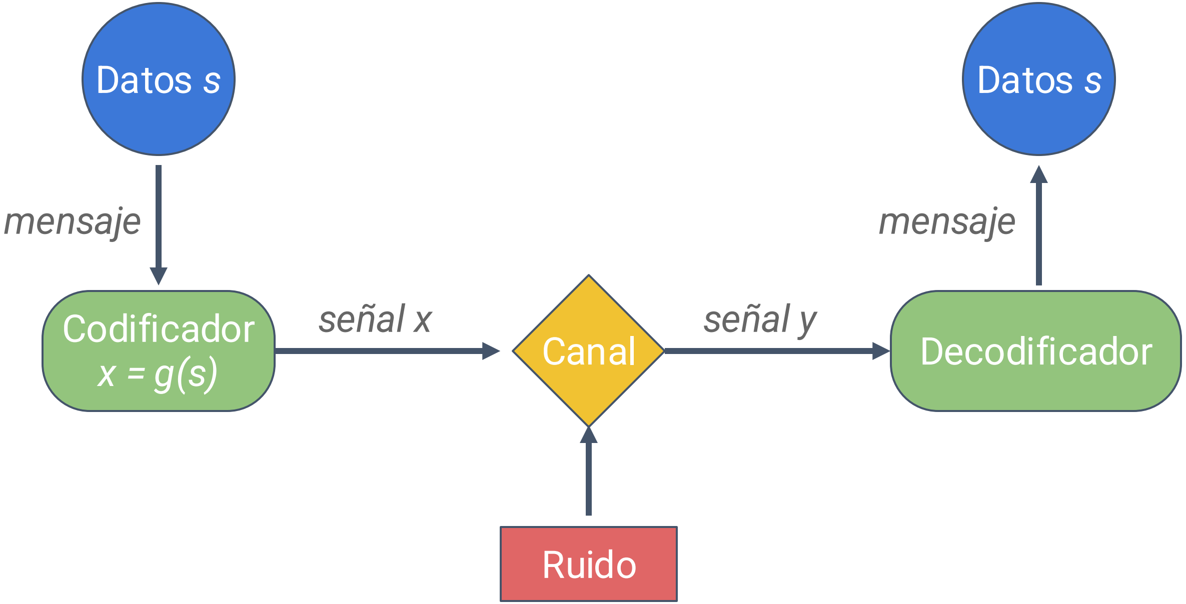
Problema fundamental de la comunicación

- Reproducir en un punto (destino) el mensaje enviado en otro punto (fuente).



Problema fundamental de la comunicación

- Reproducir en un punto (destino) el mensaje enviado en otro punto (fuente).



Variables aleatorias y transmisión de datos

- Un mensaje $\mathbf{s}$ es una secuencia ordenada de k símbolos
- Modelamos cada símbolo como la salida de una variable aleatoria
- Puede tomar valores del alfabeto A_s (α símbolos diferentes)

$$\mathbf{s} = (s_1, \ldots, s_k)$$

$$A_s = \{s_1, \ldots, s_\alpha\}$$

Variables aleatorias y transmisión de datos

- Cada símbolo, además, aparece (o se genera) con una cierta probabilidad:

Datos s

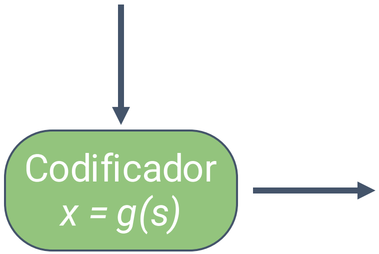
$$A_s = \{s_1, \ldots, s_\alpha\}$$

$$p(S) = \{p(s_1), \ldots, p(s_\alpha)\}$$

"Fuente"

Codificación del mensaje

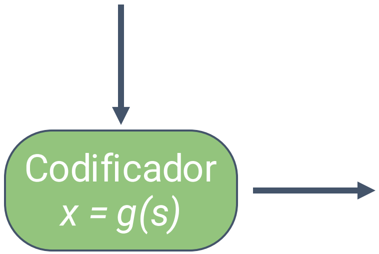
- Antes de ser transmitido, el mensaje original $\mathbf{s}$ puede ser codificado en una nueva secuencia $\mathbf{x}$ de longitud n, que eventualmente puede tener símbolos diferentes.



$$\mathbf{s} = (s_1, ..., s_k)$$
$$g(\mathbf{s}) = (x_1, \ldots, x_n) = \mathbf{x}$$

Codificación del mensaje

- Antes de ser transmitido, el mensaje original $\mathbf{s}$ puede ser codificado en una nueva secuencia $\mathbf{x}$ de longitud n, que eventualmente puede tener símbolos diferentes.
- En este proceso se puede *eliminar la redundancia* natural en los datos (compresión con o sin pérdida) o *sumar redundancia* para ganar robustez frente al ruido.



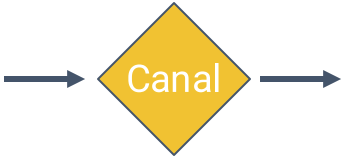
$$g(\mathbf{s}) = (x_1, \dots, x_n) = \mathbf{x}$$

$$A_x = \{x_1, \dots, x_m\}$$

$$p(X) = \{p(x_1), \dots, p(x_m)\}$$

Transmisión del mensaje

- La transmisión del mensaje codificado produce una salida de longitud n en el extremo del canal.
- Modelamos esto con una variable aleatoria Y, que puede tomar valores en A_y.



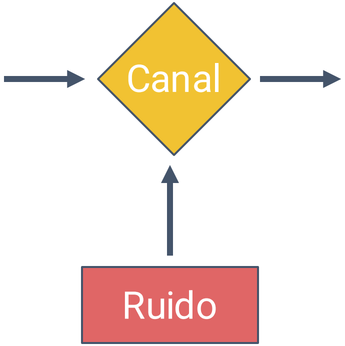
$$\mathbf{y} = (y_1, \ldots, y_n)$$

$$A_y = \{y_1, \ldots, y_m\}$$

$$p(Y) = \{p(y_1), \ldots, p(y_m)\}$$

Transmisión del mensaje

- La transmisión del mensaje codificado produce una salida de longitud n en el extremo del canal.
- Modelamos esto con una variable aleatoria Y, que puede tomar valores en A_y.



$$\mathbf{y} = (y_1, \ldots, y_n)$$

$$A_y = \{y_1, \ldots, y_m\}$$

$$p(Y) = \{p(y_1), \ldots, p(y_m)\}$$

Transmisión del mensaje

- *Las secuencias de salida son interpretadas como si implicaran la presencia de una determinada secuencia de entrada.*
- Si esta interpretación no es correcta (ruido), entonces la comunicación contiene errores.
- El ruido en el canal induce errores en la interpretación de las salidas.

Ejemplo: codificación del mensaje

- Podemos pensar en la función que codifica el mensaje como una tabla *"look-up"*.

- Cada entrada indica cómo se traduce el símbolo de A_s en uno de A_x.

Símbolo	Código
$s_1 = 3$	$x_1 = 000$
$s_2 = 6$	$x_2 = 001$
$s_3 = 9$	$x_3 = 010$
$s_4 = 12$	$x_4 = 011$
$s_5 = 15$	$x_5 = 100$
$s_6 = 18$	$x_6 = 101$
$s_7 = 21$	$x_7 = 110$
$s_8 = 24$	$x_8 = 111$

Distancia de un código

La distancia de un código puede referirse a diferentes tipos de distancia, pero las dos más comunes son la **distancia de Hamming** y la **distancia mínima**:

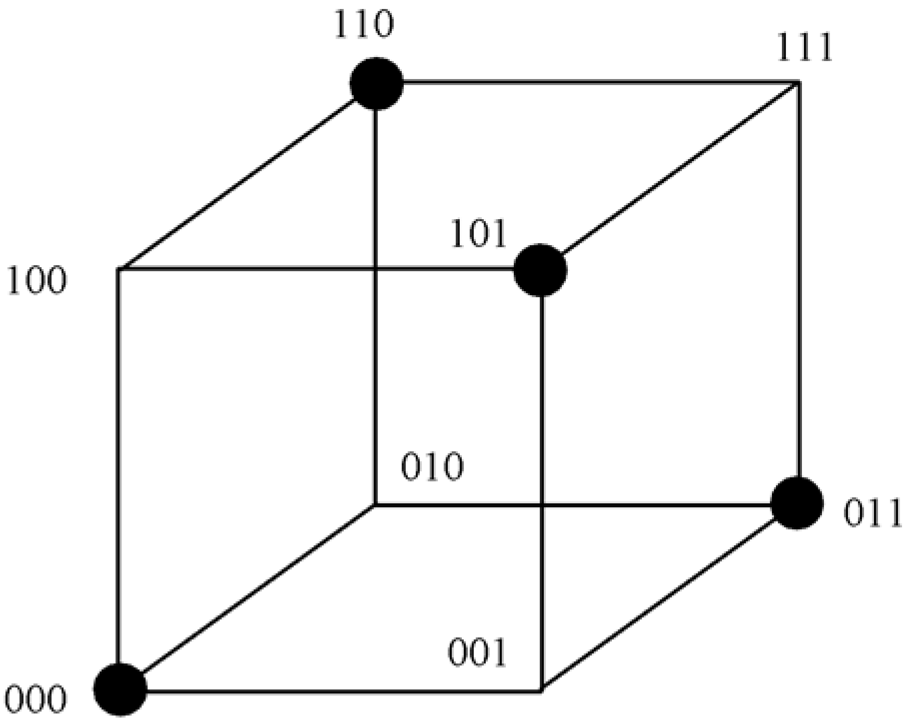
1. **Distancia de Hamming**: Es el número de posiciones en las que dos palabras de código (secuencias de bits) son diferentes. Por ejemplo, la distancia de Hamming entre los códigos 1011101 y 1001001 es 2.
2. **Distancia Mínima**: Es la mínima distancia de Hamming entre cualquier par de palabras distintas en un código. Esta distancia mínima es lo que se usa para determinar la capacidad de un código para detectar y corregir errores

Importancia de la Distancia Mínima

Detección de Errores: Un código con distancia mínima d_{min} puede detectar hasta $d_{min} - 1$ errores.

Corrección de Errores: Un código con distancia mínima d_{min} puede corregir hasta $(d_{min} - 1)/2$ errores.

Distancia de código 2 (Paridad)



Paridad Par

Esquema de Repetición

Paso 1: Definición del Esquema de Repetición (R3)
En un esquema de repetición (R3), cada bit de datos se transmite 3 veces. El receptor utiliza una regla de mayoría para decidir el valor del bit original. Si al menos 2 de los 3 bits recibidos son iguales, se asume que ese es el bit original.

Paso 2: Probabilidad de Error sin Redundancia
La probabilidad de error sin redundancia, P_e, es simplemente la probabilidad de que un bit transmitido sea recibido incorrectamente.

Esquema de Repetición

Paso 3: Probabilidad de Error con Redundancia (R3)

Para el esquema R3, necesitamos calcular la probabilidad de que la mayoría de los 3 bits recibidos sean incorrectos. Hay tres posibles formas en que esto puede suceder:

- Todos los 3 bits están incorrectos.
- Dos de los 3 bits están incorrectos.

La probabilidad de que todos los 3 bits estén incorrectos es f^3

Esquema de Repetición

La probabilidad de que exactamente 2 de los 3 bits estén incorrectos se calcula usando la combinación:

P(2 de 3 incorrectos) = $f^2 (1-f)$

Por lo tanto, la probabilidad total de error con redundancia, $P_e(R3)$ es:

$$P_e(R3) = f^3 + 3 f^2 (1-f)$$

Ejemplo: codificación del mensaje (BSC)

- Volvamos al desafío de detectar y corregir errores en BSC ($f = 0.1$).
- **Esquema simple:** redundancia por repetición R_3

Símbolo	Código
$s_1 = 0$	$x_1 = 000$
$s_2 = 1$	$x_2 = 111$

$$\mathbf{s} = 0\ 0\ 1\ 0\ 1\ 1\ 0$$ ← *mensaje original*

$$g(\mathbf{s}) = 000\ 000\ 111\ 000\ 111\ 111\ 000$$ ← *mensaje codificado*

Ejemplo: codificación del mensaje (BSC)

- Volvamos al desafío de detectar y corregir errores en BSC ($f=0.1$).
- **Esquema simple:** redundancia por repetición R_3

Símbolo	Código
$s_1 = 0$	$x_1 = 000$
$s_2 = 1$	$x_2 = 111$

$$\mathbf{s} = 0\ 0\ 1\ 0\ 1\ 1\ 0 \longleftarrow \textit{mensaje original}$$

$$g(\mathbf{s}) = 000\ 000\ 111\ 000\ 111\ 111\ 000 \longleftarrow \textit{mensaje codificado}$$

$$\boldsymbol{n} = 000\ 001\ 000\ 000\ 101\ 000\ 000 \longleftarrow \textit{ruido en el canal BSC}$$

Ejemplo: codificación del mensaje (BSC)

- Volvamos al desafío de detectar y corregir errores en BSC (f=0.1).
- **Esquema simple:** redundancia por repetición R_3

Símbolo	Código
$s_1 = 0$	$x_1 = 000$
$s_2 = 1$	$x_2 = 111$

$$\mathbf{s} = 0\ 0\ 1\ 0\ 1\ 1\ 0 \quad \longleftarrow \textit{mensaje original}$$

$$g(\mathbf{s}) = 000\ 000\ 111\ 000\ 111\ 111\ 000 \quad \longleftarrow \textit{mensaje codificado}$$

$$\boldsymbol{n} = 000\ 001\ 000\ 000\ 101\ 000\ 000 \quad \longleftarrow \textit{ruido en el canal BSC}$$

$$\boldsymbol{y} = 000\ 001\ 111\ 000\ 010\ 111\ 000$$

Ejemplo: codificación del mensaje (BSC)

- Volvamos al desafío de detectar y corregir errores en BSC ($f=0.1$).
- **Esquema simple:** redundancia por repetición R_3

Símbolo	Código
$s_1 = 0$	$x_1 = 000$
$s_2 = 1$	$x_2 = 111$

$$\mathbf{s} = 0\ 0\ 1\ 0\ 1\ 1\ 0 \longleftarrow \textit{mensaje original}$$

$$g(\mathbf{s}) = 000\ 000\ 111\ 000\ 111\ 111\ 000 \longleftarrow \textit{mensaje codificado}$$

$$\mathbf{n} = 000\ 001\ 000\ 000\ 101\ 000\ 000 \longleftarrow \textit{ruido en el canal BSC}$$

$$\mathbf{y} = \underbrace{000}_{0}\ \underbrace{001}_{0}\ \underbrace{111}_{1}\ \underbrace{000}_{0}\ \underbrace{010}_{0}\ \underbrace{111}_{1}\ \underbrace{000}_{0} \longleftarrow \textit{mensaje decodificado}$$

Ejemplo: codificación del mensaje (BSC)

- Volvamos al desafío de detectar y corregir errores en BSC (f=*0.1*).
- **Esquema simple:** redundancia por repetición R_3

Símbolo	Código
$s_1 = 0$	$x_1 = 000$
$s_2 = 1$	$x_2 = 111$

$$\mathbf{s} = 0\ 0\ 1\ 0\ 1\ 1\ 0 \longleftarrow \textit{mensaje original}$$

$$g(\mathbf{s}) = 000\ 000\ 111\ 000\ 111\ 111\ 000 \longleftarrow \textit{mensaje codificado}$$

$$\boldsymbol{n} = 000\ 001\ 000\ 000\ 101\ 000\ 000 \longleftarrow \textit{ruido en el canal BSC}$$

$$\boldsymbol{y} = \underbrace{000}_{0}\ \underbrace{001}_{0}\ \underbrace{111}_{1}\ \underbrace{000}_{0}\ \underbrace{010}_{0}\ \underbrace{111}_{1}\ \underbrace{000}_{0} \longleftarrow$$

errores corregidos
errores sin corregir

Ejercicio!

- Calcular la probabilidad de error en un esquema de repetición (R_3).
- Mostrar que mejora respecto al caso sin redundancia para $f = 0.1$.

Ejercicio!

- Calcular la probabilidad de error en un esquema de repetición (R_3).
  - Un error en R_3 se produce cuando se invierten dos o más bits en un bloque de 3.
  - Por lo tanto, la probabilidad de error en R_3 es la suma de dos términos:

$$f^3 \qquad\qquad 3f^2(1-f)$$

p de 3 bits invertidos — *p* de 2 bits invertidos

$$p = f^3 + 3f^2(1-f)$$

Ejercicio!

- Mostrar que mejora respecto al caso sin redundancia para f = 0.1.

$$p = f^3 + 3f^2(1 - f)$$

Para f = 0.1:

$$(0.1)^3 + 3(0.1)^2(1 - 0.1) = 0.028$$ prob. de error por bit

Redundancia por repetición

- Esquema simple: muy fácil de codificar/decodificar mensajes
- Reduce la probabilidad de error
- Podríamos seguir bajando arbitrariamente la tasa de error agregando bits de redundancia, pero..

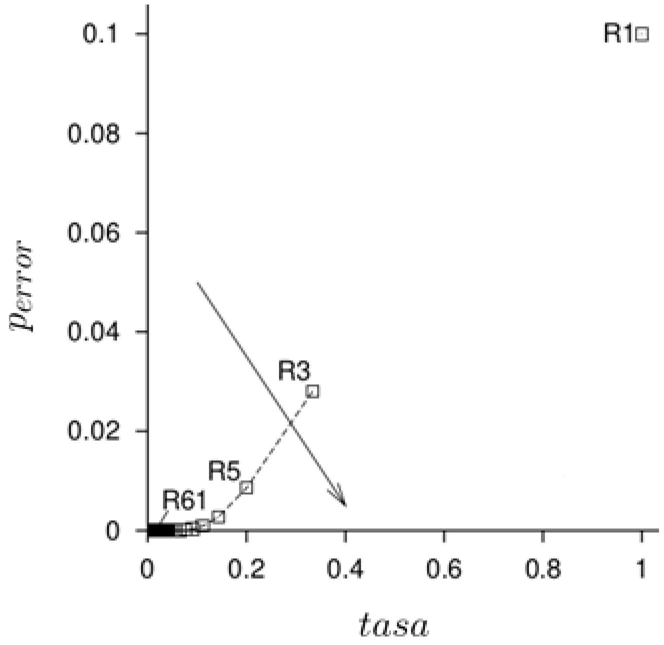
¿Qué precio pagamos?

Redundancia por repetición

- Esquema simple: muy fácil de codificar/decodificar mensajes
- Reduce la probabilidad de error
- Podríamos seguir bajando arbitrariamente la tasa de error agregando bits de redundancia, pero..

¿Qué precio pagamos?

- Tasa de transmisión cae según el factor *k* (*k = 3* en el ejemplo)

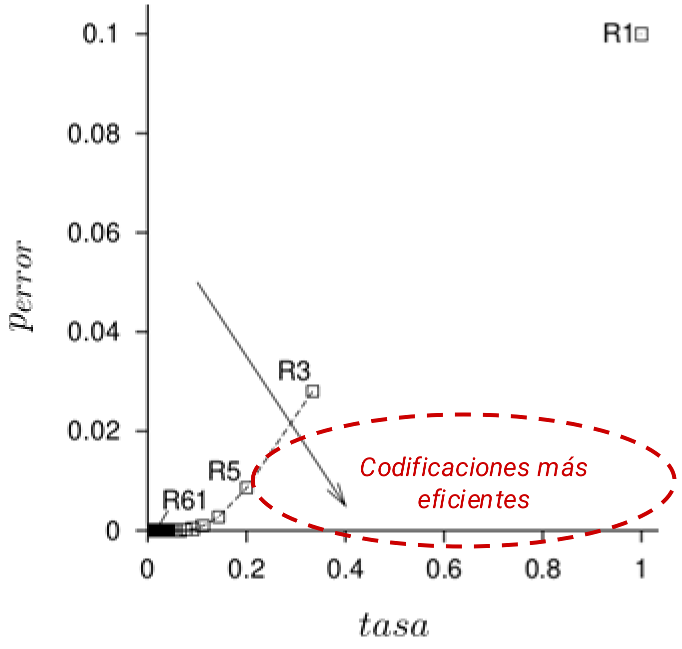


Redundancia por repetición

- Esquema simple: muy fácil de codificar/decodificar mensajes
- Reduce la probabilidad de error
- Podríamos seguir bajando arbitrariamente la tasa de error agregando bits de redundancia, pero..

¿Qué precio pagamos?

- Tasa de transmisión cae según el factor *k* (*k = 3* en el ejemplo)

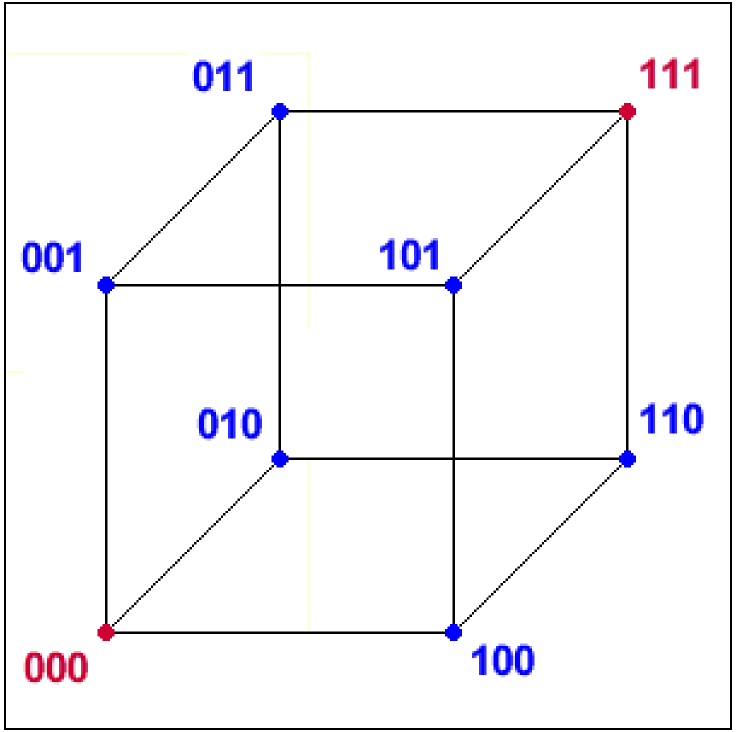


Redundancia por bloques

- Nos gustaría comunicarnos con muy baja probabilidad de error a una tasa alta. ¿Podemos mejorar la redundancia mediante repetición?

- *Idea: Agregar redundancia por bloques de datos en lugar de codificar un bit a la vez.*

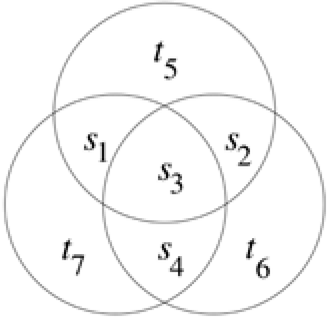
Distancia de código 3 (Hamming)



- Diagonales de un cubo

El código de Hamming

Código de **Hamming (7,4):** Cada 4 bits de mensaje se transmiten 7 (3 bits de redundancia)

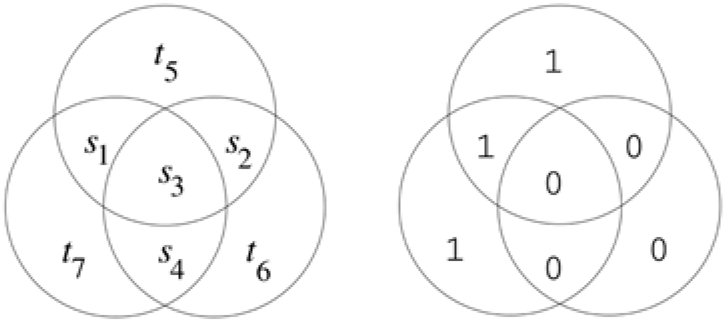


mensaje= $s_1\ s_2\ s_3\ s_4$

mensaje hamming = $s_1\ s_2\ s_3\ s_4\ t_5\ t_6\ t_7$

El código de Hamming

Código de Hamming (7,4): Cada 4 bits de mensaje se transmiten 7 (3 bits de redundancia)



$$\mathbf{s} = 1\ 0\ 0\ 0$$

$$g(\mathbf{s}) = 1\ 0\ 0\ 0\ 1\ 0\ 1$$

El código de Hamming

Código de Hamming (7,4): Cada 4 bits de mensaje se transmiten 7 (3 bits de redundancia)

Símbolo	Código	Símbolo	Código
$s_1 = 0000$	$x_1 = 0000000$	$s_9 = 1000$	$x_9 = 1000101$
$s_2 = 0001$	$x_2 = 0001011$	$s_{10} = 1001$	$x_{10} = 1001110$
$s_3 = 0010$	$x_3 = 0010111$	$s_{11} = 1010$	$x_{11} = 1010010$
$s_4 = 0011$	$x_4 = 0011100$	$s_{12} = 1011$	$x_{12} = 1011001$
$s_5 = 0100$	$x_5 = 0100110$	$s_{13} = 1100$	$x_{13} = 1100011$
$s_6 = 0101$	$x_6 = 0101101$	$s_{14} = 1101$	$x_{14} = 1101000$
$s_7 = 0110$	$x_7 = 0110001$	$s_{15} = 1110$	$x_{15} = 1110100$
$s_8 = 0111$	$x_8 = 0111010$	$s_{16} = 1111$	$x_{16} = 1111111$

El código de Hamming

- Códigos más complejos hacen más difícil el proceso de decodificación del mensaje. ¿Cómo podemos recuperar la secuencia de bits original?

  - Opción 1: Buscar en la tabla y quedarnos con la opción más cercana (distancia bit a bit). ¿Es la opción más eficiente?

  - Opción 2: ¿?

Decodificando Hamming

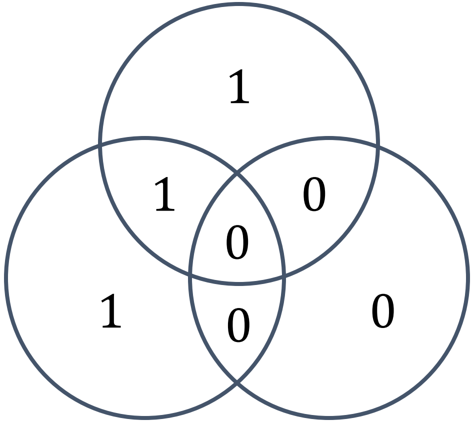
- Supongamos que se transmite **s** como:

$$\mathbf{s} = 1\ 0\ 0\ 0 \longleftarrow \textit{mensaje original}$$

$$g(\mathbf{s}) = 1\ 0\ 0\ 0\ 1\ 0\ 1 \longleftarrow \textit{Hamming (7,4)}$$

$$\boldsymbol{n} = 0\ \color{red}{1}\ 0\ 0\ 0\ 0\ 0 \longleftarrow \textit{Ruido}$$

$$\boldsymbol{y} = 1\ \color{red}{1}\ 0\ 0\ 1\ 0\ 1 \longleftarrow \textit{Mensaje recibido}$$

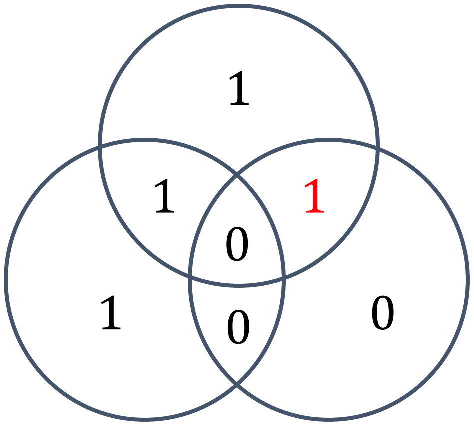


Decodificando Hamming

$\mathbf{s} = 1\ \ 0\ \ 0\ \ 0$ ⟵ *mensaje original*

$\boldsymbol{y} = 1\ \ 1\ \ 0\ \ 0\ \ 1\ \ 0\ \ 1$ ⟵ *Mensaje recibido*

- Paso 1: ubicar los elementos en los círculos

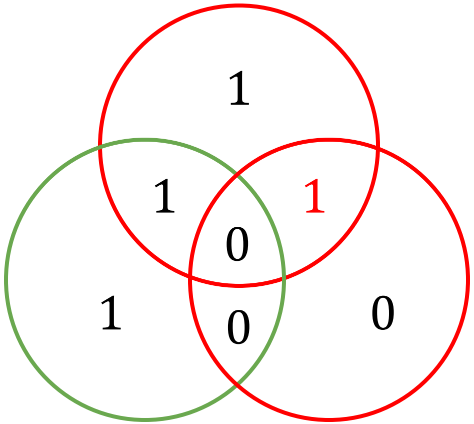


Decodificando Hamming

$$\mathbf{s} = 1\ \ 0\ \ 0\ \ 0 \longleftarrow$$ *mensaje original*

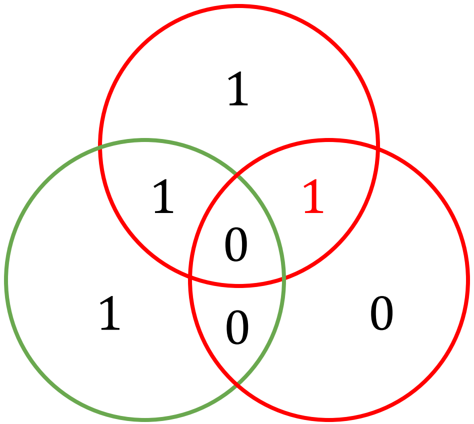
$$\boldsymbol{y} = 1\ \ {\color{red}1}\ \ 0\ \ 0\ \ 1\ \ 0\ \ 1 \longleftarrow$$ *Mensaje recibido*

- Paso 1: ubicar los elementos en los círculos
- Paso 2: marcar los círculos sin paridad par



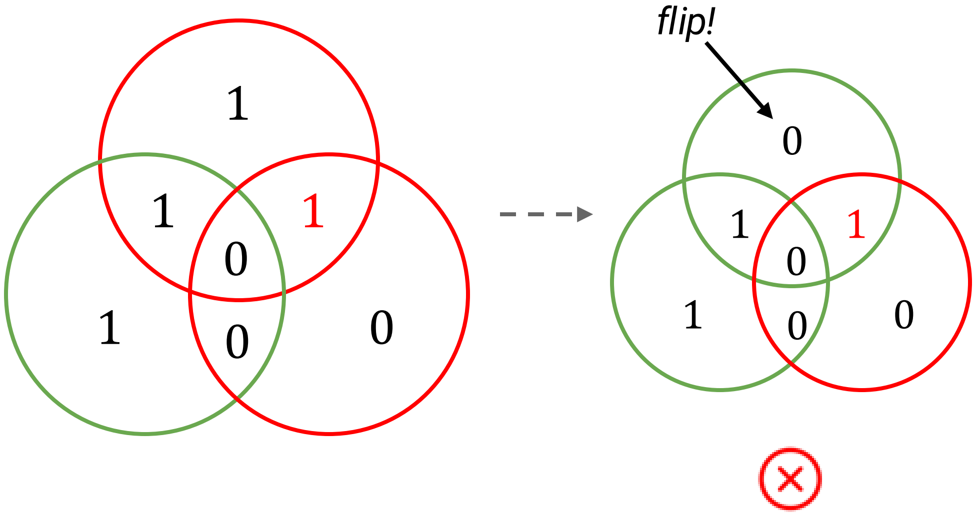
Decodificando Hamming

- Paso 3: encontrar la menor cantidad de *flips* que restauran la paridad:



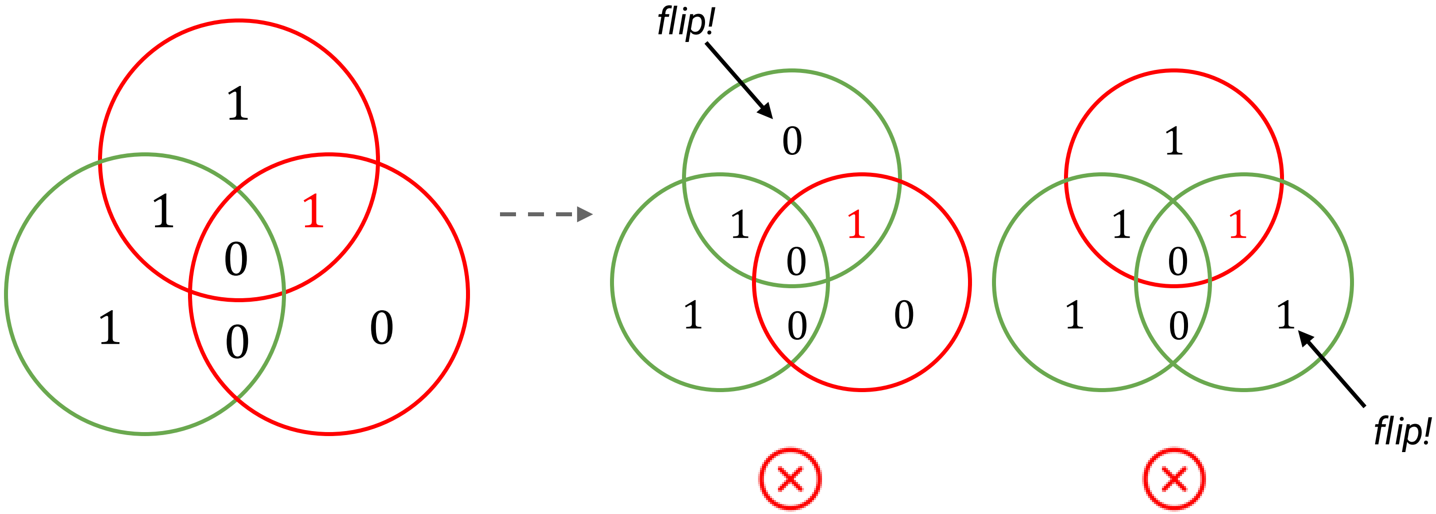
Decodificando Hamming

- Paso 3: encontrar la menor cantidad de *flips* que restauran la paridad:



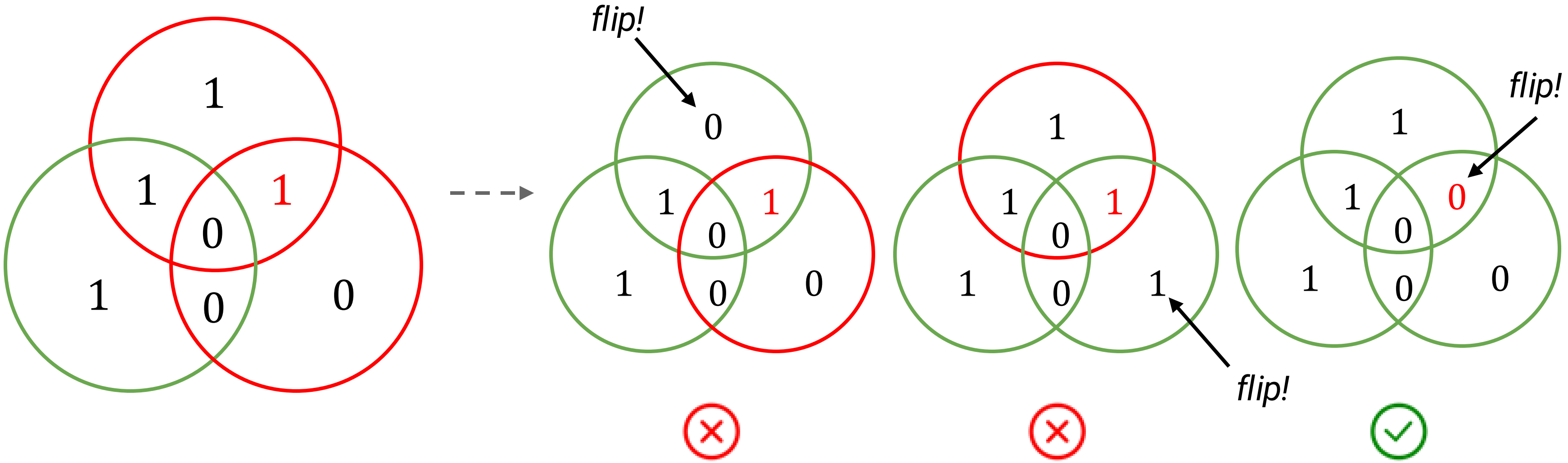
Decodificando Hamming

- Paso 3: encontrar la menor cantidad de *flips* que restauran la paridad:



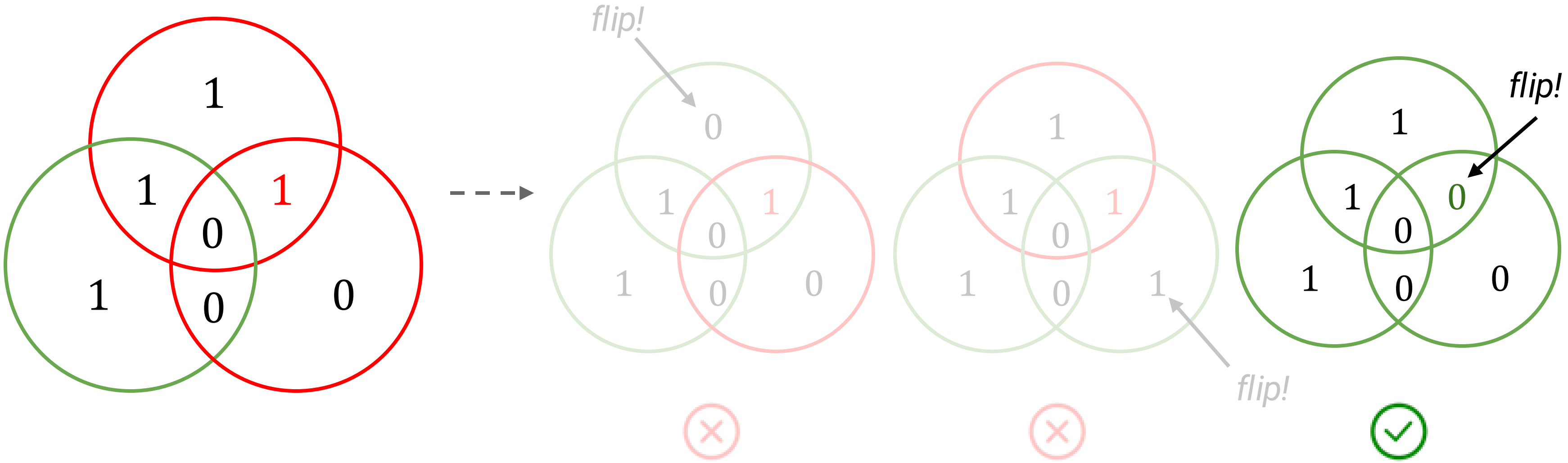
Decodificando Hamming

- Paso 3: encontrar la menor cantidad de *flips* que restauran la paridad:



Decodificando Hamming

- Paso 3: encontrar la menor cantidad de *flips* que restauran la paridad:

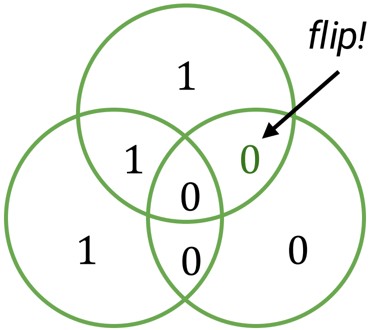


Decodificando Hamming

$\mathbf{s} = 1\ 0\ 0\ 0$ ⟵ *mensaje original*

$y = 1\ 1\ 0\ 0\ 1\ 0\ 1$ ⟵ *Mensaje recibido*

$1\ 0\ 0\ 0\ 1\ 0\ 1$ ⟵ *Mensaje corregido*



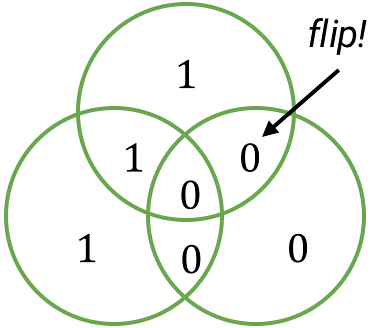
- Las violaciones a la paridad de los círculos siguen un patrón y se pueden tabular en una "*matriz de paridad*":

Decodificando Hamming

$\mathbf{s} = 1\ 0\ 0\ 0$ ⟵ *mensaje original*

$\boldsymbol{y} = 1\ \color{red}{1}\ 0\ 0\ 1\ 0\ 1$ ⟵ *Mensaje recibido*

$1\ 0\ 0\ 0\ \color{lightgray}{1\ 0\ 1}$ ⟵ *Mensaje corregido*



- Las violaciones a la paridad de los círculos siguen un patrón y se pueden tabular en una “*matriz de paridad*”:

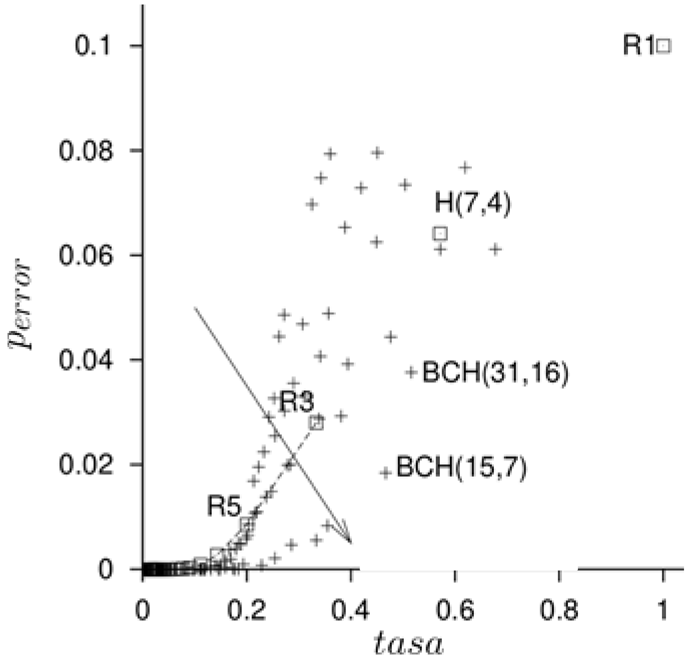
Patrón	000	001	010	011	100	101	110	111
flip	-	t_7	t_6	s_4	t_5	s_1	s_2	s_3

Código de Hamming: resumen

- Todos los bloques recibidos están, a lo sumo, a un bit de distancia de lograr la paridad.
- Solo se pueden detectar y corregir errores de hasta un bit. *Dicho de otro modo:* se producirá un error de decodificación siempre que el ruido haya invertido más de un bit en un bloque de 7.
- La probabilidad de error es similar a la redundancia por repetición (R_3), pero a una tasa de transmisión más alta (4/7 vs. 4/12).
- Existen generalizaciones a distintos tamaños de bloque (ej, [7,4],[15,7],[31,16], etc. Se conocen como *códigos BCH*.

Codificación: resumen

- Parece haber un trade-off entre la probabilidad de error (que queremos reducir) y la tasa de transmisión (que queremos mantener alta). *“No pain, no gain”*.



Codificación: resumen

- Parece haber un trade-off entre la probabilidad de error (que queremos reducir) y la tasa de transmisión (que queremos mantener alta). *“No pain, no gain”*.

- *¿Podemos obtener métodos de codificación con una performance dentro del área roja? ¿Cuál es la performance máxima que puede lograrse con el mejor algoritmo?*