Tratamiento de imágenes

Almacenamiento, Codificación y Compresión

Héctor Alejandro Montes
h.a.montes@fi.uaemex.mx
http://fi.uaemex.mx/h.a.montes

Advertencia

No use estas diapositivas como referencia única de estudio durante este curso. La información contenida aquí es una guía para las sesiones de clase y de estudio futuro. Para obtener información más completa, refiérase a la bibliografía listada en la última diapositiva.

Almacenamiento, Compresión y Codificación

- Una imagen MxN con R filtros codificada a profundidad k_i, {i=1..R}, ocupa:

$$b = \sum_{i=1}^{R} M \cdot N \cdot k_i$$

- Y si todos los filtros tienen la misma profundidad:

$$b = M \cdot N \cdot R \cdot k$$

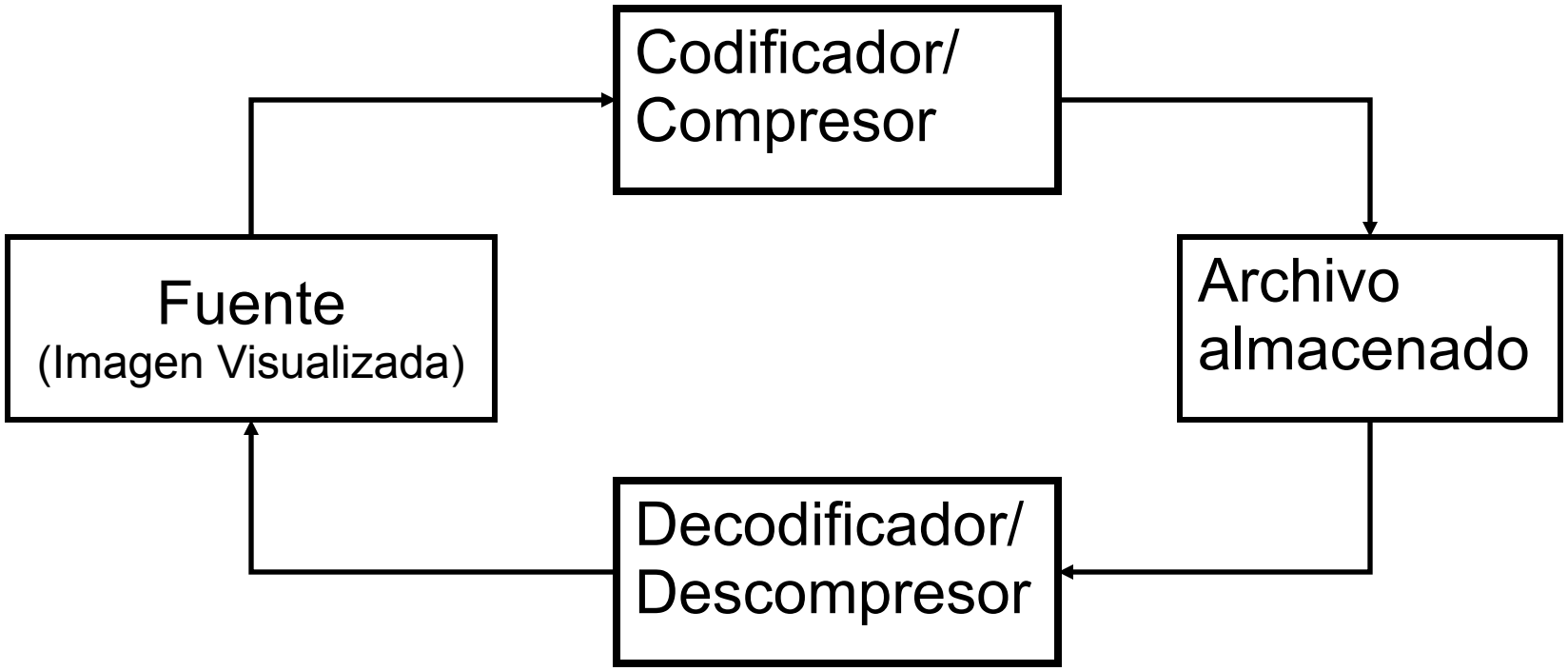
Almacenamiento, Compresión y Codificación

- Muchas aplicaciones requieren almacenar y transmitir imágenes:
  - Televisión
  - Teledetección
  - Medicina y Telemedicina
  - FAX
  - etc.
- Existe interés en tener un *almacenamiento*, *compresión* y *codificación* óptimos en términos de espacio

Almacenamiento, Compresión y Codificación

- Hay una diferencia notable entre *cómo se captura* (muestrea y digitaliza), *cómo se visualiza* y *cómo se almacena* la imagen
  - Al visualizar una imagen no sabemos como está almacenada
- Para reducir el espacio de almacenamiento, se elimina la *información redundante*

Almacenamiento, Compresión y Codificación



Formatos de Imágenes

Formatos de Imágenes

- Cosas que sabemos:
  - Son medios de organizar y almacenar imágenes
  - Una archivo de imagen se compone de valores de pixeles que se *rasterizan* cuando se despliegan
  - El valor de cada pixel representa magnitudes de brillo y color

Formatos de Imágenes

- El tamaño del archivo se incrementa según el número de pixeles que componenen la imagen y su *profundidad* en color
- Entre más grande sea el número de pixeles (MxN), más grande es la *resolución*
- Cada pixel incrementa su tamaño cuando su profundidad se incrementa:
  - 8-bit/pixel (1 byte) almacena 256 colores
  - 24-bit/pixel (3 bytes) almacena 16 millones de colores

Formatos de Imágenes

- Se utilizan algoritmos de ***compresión*** para reducir el tamaño de las imágenes
- Una imagen en color (24-bit/pixel) tomada por una cámara de 12 MP ocupa 36,000,000 bytes de memoria
- Para que este proceso sea práctico, se desarrollaron los ***formatos de imágenes***.
- Muchos de estos formatos utilizan algún algoritmo de ***compresión***

Estándares de Almacenamiento

- **JPEG** (Joint Photographic Experts Group)
  - Estándar ISO/IEC IS 10918-1 | ITU-T Recommendation T.81
  - Utilizado por la mayoría de las cámaras digitales comerciales
  - Añada ***metadatos*** a través de **EXIF** e **IPTC**
    - EXchangeable Image file Format (no en JPEG 2000, PNG o GIF).
    - International Press Telecommunications Council
  - http://www.jpeg.org/jpeg/index.html
  - www.w3.org/Graphics/JPEG/itu-t81.pdf

Estándares de Almacenamiento

- **TIFF** (Tag Image File Format)
  - Actualmente en la versión 6.0
  - Considerado el más flexible, ya que una imagen tiff puede contener imágenes en otros formatos.
  - Creado originalmente por Aldus y Microsoft y actualmente controlado por Adobe

Estándares de Almacenamiento

- **JPEG2000**
  - Aún en desarrollo
  - También está aceptado bajo el ala de ISO y de ITU
  - http://www.jpeg.org/jpeg2000/index.html

Estándares de Almacenamiento

- **GIF** (Graphics Interchange Format)
  - Desarrollado por CompuServe
  - Utiliza LZW para compresión
  - *Patentado*, aunque sus patentes terminaron en 2003-2004 y ahora puede ser utilizado sin restricciones

Estándares de Almacenamiento

- **PNG** (Portable Network Graphics)
  - ISO/IEC 15948:2003 y W3C
  - Diseñado para la Web
  - Repuesta libre de patente para GIF
  - Portable y sin pérdida
  - http://www.w3.org/TR/PNG

Estándares de Almacenamiento

- **BMP** (Windows Bitmap format)
  - Desarrollado por Microsoft
  - No existe un documento como tal que contenga la especificación formal
  - http://www.martinreddy.net/gfx/2d/BMP.txt

Estándares de Almacenamiento

- **PNM** (Portable Any Map)
  - Es un estándar abierto que contiene a otros 4:
    - PPM - Imágenes en color
    - PBM - Imágenes binarias
    - PGM - Imágenes en escala de grises
    - PAM - General, y no únicamente de imágenes
  - http://netpbm.sourceforge.net/doc/pnm.html

Estándares de Almacenamiento

- **PPM** (Portable Pixel Map)
  - Para imágenes en color
  - Posiblemente el formato más sencillo de todos.
  - Archivos en ASCII
  - Libre
  - http://netpbm.sourceforge.net/doc/ppm.html

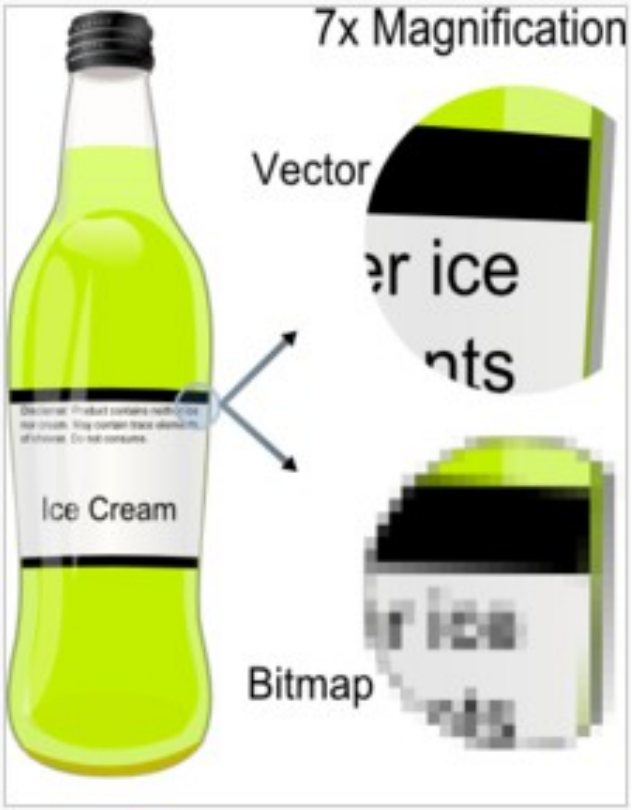
Estándares de Almacenamiento

- **PGM** (Portable Gray Map)
  - Para imágenes en escala de gris
  - Formato sencillo
  - Archivos en ASCII
  - Libre
  - http://netpbm.sourceforge.net/doc/pgm.html

Estándares de Almacenamiento

- **PBM** (Portable Bit Map)
  - Para imágenes binarias
  - Formato sencillo
  - Archivos en ASCII
  - Libre
  - http://netpbm.sourceforge.net/doc/pbm.html

Formatos en vectores



SVG: Scalable Vector Graphics

Formatos en vectores

Estándares de Almacenamiento

- Algunos otros formatos comunes:
  - TGA (Targa file format)
  - PCX
  - DCX - CCITT estándar para FAX
  - CGM (Computer Graphics Metafile)
  - EPS
  - … y muchos otros
- Una buena referencia sobre estándares de formatos de imágenes
  - http://www.martinreddy.net/gfx/2d-hi.html

I. Teoremas fundamentales de la codificación

Teoría de la Información

Teoría de la Información

- Rama de la matemáticas aplicadas
- Creada en 1948 por *Claude E. Shannon*
- Involucra la *cuantificación* de la información
- Desarrollada para encontrar límites en operaciones de *procesamiento de señales* para *comprimir*, *almacenar* y *comunicar* datos

Teoría de la Información

- Codificación de la fuente:
  - Se denota por S_i a cada símbolo (vector de intensidades) en el mensaje (la imagen original)
  - Al conjunto $S=\{S_1, S_2, \ldots, S_q\}$ de símbolos se la llama alfabeto de símbolos de entrada.

Teoría de la Información

- Codificación de fuente:
  - Cada símbolo S_i aparece con probabilidad $P(S_i)$
    - Es decir, los diferentes vectores de intensidad aparecen un número distinto de veces en la imagen
    - Los símbolos no tienen que ser equiprobables y debe cumplirse:

$$\sum_{i=1}^{q} P(S_i) = 1$$

Teoría de la Información

- Codificación de fuente:
  - Si la aparición de un símbolo no influye sobre la aparición de los demás, se dice que los símbolos son *estadísticamente independientes*
  - Si los símbolos son estadísticamente independientes se habla de *fuente de memoria nula*

Teoría de la Información

En una imagen de MxN píxeles, el vector de intensidad $S_i = v = <v_1, v_2, \ldots, v_R>$ ocurrirá n veces.

- La probabilidad de ocurrencia de v es:

$$P(v) = \frac{n}{M \times N}$$

Teoría de la Información

- Codificación:
  - Existen diferentes formas de codificar
    - Una forma de codificación trivial es:
      - Contar el número q de símbolos S_i existentes
      - Calcular el número de símbolos x_i necesarios para codificar en base r todos los símbolos de S

$$\lceil \log_r q \rceil$$

      - Asignar una combinación $X_i=x_1, \ldots, x_l$ a cada S_i

Teoría de la Información

- Codificación:
  - A las diferentes combinaciones X_i se les llama *palabras código*
  - Al conjunto $X=\{X_1, X_2, \ldots, X_q\}$ de palabras código se la llama *alfabeto código*.
  - A la función de codificación $S \Rightarrow X$ se le llama *código*

Teoría de la Información

- Podemos entender el código como el algoritmo de codificación, es decir lo que da lugar al *formato* de la imagen.
  - Formatos conocidos: TIFF, JPG, GIF, PNG, etc.
  - Cada uno es una forma de codificar la información de una imagen.

Teoría de la Información

- En una imagen, el alfabeto código son las cadenas binarias que codifican cada vector de intensidades
  - Excepto en formatos como SVG
- Aunque normalmente estas cadenas binarias tienen longitud fija, no siempre es así.

Teoría de la Información

- Longitud media:
  - Si se usan l_i bits para representar cada símbolo $S_i=v$ la longitud media de bits necesaria para codificar un bit es:

$$\overline{L}=\sum_{i=1}^{q} P(S_i)\, l_i$$

Teoría de la Información

- La definición anterior ya incluye a todos los R filtros.
- Si todos los filtros tienen la misma profundidad k la longitud media es

$$\overline{L} = P(S_i) \cdot R \cdot k$$

Teoría de la Información

- Se define la ***cantidad de información*** que aporta un símbolo como:

$$I(S_i) = \log_b \frac{1}{P(S_i)}$$

Teoría de la Información

- Se define la ***entropía de la fuente***, H(S), como la cantidad de información media que se obtiene:

$$H(S)=\sum_{i=1}^{q} P(S_i)\,I(S_i)=\sum_{i=1}^{q} P(S_i)\log_2\frac{1}{P(S_i)}$$

Teoría de la Información

- Se cumple que:

$$H_r(S) = \frac{H(S)}{\log_2 q} \leq \overline{L} \,|\, S = \{S_1, S_2, K, S_q\}$$

- Esto es, la longitud media que ocupa una imagen es más larga que la cantidad de información media contenida en ella; es decir, la imagen incorpora una cierta cantidad de ***información no necesaria o redundante.***

Teoría de la Información

- **Redundancia:**
  - Se define redundancia como la información no necesaria

$$R = 1 - \eta$$

Teoría de la Información

- La redundancia es un punto clave en la *compresión* de imágenes digitales.
- Se buscan formatos de imágenes que tengan un alto rendimiento y por ende una baja redundancia

Teoría de la Información

- Redundancia relativa:
  - Si x_1 y x_2 denotan las palabras código de dos alfabetos código X_1 y X_2 que representan la misma información S_i, se define la redundancia relativa (relative data redundancy) como:

$$R_D = 1 - \frac{1}{C_R}$$

  - Donde C_R es el ***radio de compresión*** $C_R = x_1/x_2$

Teoría de la Información

- La redundancia relativa indica cuánta información redundante contiene una representación con respecto a otra.
  - Ejemplo: Un C_R=10 significa que la primera representación tiene 10 bits por cada bit que necesita la segunda representación, o en otras palabras, la primera representación tiene un 90% de información redundante con respecto a la segunda.

Compresión y Redundancia

Compresión

- En imágenes digitales se identifican tres tipos de redundancia:
  - Redundancia de *codificación*
  - Redundancia entre píxeles o *interpixel*
    - También conocida como redundancia espacial o geométrica
  - Redundancia *psicovisual*
- Se obtiene *compresión* de la imagen cuando eliminamos alguna de estas redundancias

Redundancia de Codificación

- Depende de la codificación elegida (i.e. del formato de la imagen)
  - Ejemplo: Se pueden obtener mejores compresiones si utilizamos una codificación de longitud variable en lugar de longitud fija, asignando cadenas más cortas a los símbolos más probables y cadenas más largas a los símbolos menos probables.

Redundancia de Codificación

Rojo	1/4	00	1/8	11
Verde	1/4	01	1/4	1
Azul	1/4	10	1/8	10
Negro	1/4	11	1/2	0

Redundancia de Codificación

- Los códigos de longitud variable permiten la compresión de la información.
- Podemos saber que valores son más probables observando el histograma de la imagen

Redundancia entre píxeles

- Tiene en cuenta las propiedades geométricas entre píxeles
- Mucha de la información de un píxel individual es redundante ya que podía haberse inferido de la información de sus píxeles vecinos.

Redundancia entre píxeles

- Existen varias técnicas y su uso depende de la imagen:
  - Técnica 1
    - Hallar los coeficientes de autocorrelación
    - Calcular las modas (valor más probable)
    - Sustituir colores acorde a un umbral elegido.
  - Técnica 2 - Codificaciones por modulación Delta
    - un pixel se codifica no por su valor sino por una diferencia de valor relativa al pixel anterior

Redundancia Psicovisual

- Basada en que el sistema de visión humano no trata toda la información captada por el ojo de la misma forma.

- La idea es eliminar todo lo que para el sistema de visión humano no es relevante

Compresión

- **Compresión** consiste en reducir la cantidad de información necesaria.

- El concepto de compresión no es exclusivo de las imágenes digitales; se aplica a cualquier información; sonidos, videos, etc

Compresión

- Podemos distinguir:
  - Comprensión sin pérdidas, o libre de errores - Lossless compression
  - Compresión con pérdidas - Lossy compression
    - Por codecs de transformación - Transform coding
    - Por codecs predictivos - Predictive coding

Compresión

- La calidad de una técnica de compresión se mide por el **radio de compresión** y se expresa normalmente como:

  n:1

  donde n es la cantidad de veces que se ha reducido la información:

  – Ejemplo: Un radio 10:1 es que la información se ha reducido 10 veces, o el 90%

III. Compresión libre de errores

Compresión libre de errores

- Tipos de código:
  - Bloque: Cada símbolo se codifica con una palabra de longitud fija
  - No singular: Un código es no singular si todas sus palabras código son distintas
  - Unívoco: Un código no singular es unívoco si y sólo si su extensión de orden *n* es no singular para cualquier valor finito de *n*, es decir no hay dudas o ambigüedades en la decodificación.

Compresión libre de errores

- Tipos de código:
  - **Compacto**: Código unívoco cuya longitud es menor o igual a la longitud media de todos los códigos unívocos que pueden aplicarse a la misma fuente y al mismo alfabeto.
    - Este es el código que vamos buscando, ya que es la mejor compresión posible en un entorno libre de errores.

Compresión libre de errores

- El **código Huffman** es un algoritmo para construir códigos compactos en una compresión libre de errores.

- Permite eliminar la redundancia de codificación

Código Huffman

- Sea una imagen cuyo conjunto de vectores de intensidades es $S=\{S_1, S_2, \ldots, S_q\}$ que aparecen con probabilidad $P(S_1), P(S_2), \ldots, P(S_q)$

Código Huffman

- Repetir hasta que sólo queden dos grupos de símbolos
  - Ordenar los símbolos acorde a sus probabilidades de mayor a menor
  - Agrupar los dos últimos con probabilidades más bajas
    - Nota: El hecho de agrupar de dos en dos es por que estamos usando una base binaria de 0 y 1. De ser una base diferente agrupamos acorde a la base.
  - Recalcular las probabilidades (sumando)

- De forma inversa, deshacer las agrupaciones asignando un 0 o un 1 a los grupos y teniendo en cuenta que cada símbolo es el prefijo del anterior.

Código Huffman

S1(0.4)	S1(0.4)	S1(0.4)	S1(0.4)	S1(0.4)	S2,S3,S4,S5,S6,S7 (0.6)
S2(0.2)	S2(0.2)	S2(0.2)	S2(0.2)	S3,S4,S5,S6,S7 (0.4)	S1(0.4)
S3(0.2)	S3(0.2)	S3(0.2)	S3(0.2)	S2(0.2)	
S4(0.09)	S4(0.09)	S5,S6,S7 (0.11)	S4,S5,S6,S7 (0.2)		
S5(0.06)	S5(0.06)	S4(0.09)			
S6(0.03)	S6,S7 (0.05)				
S7(0.02)					

Código Huffman

					S2,S3,S4,S5,S6,S7 (0)
				S3,S4,S5,S6,S7 (00)	S1(**1**)
			S3(**000**)	S2(**01**)	
		S5,S6,S7 (0010)	S4,S5,S6,S7 (001)		
	S5 (**00100**)	S4(**0011**)			
S6 (**001010**)	S6,S7 (00101)				
S7 (**001011**)					

Compresión libre de errores

- Otros algoritmos usados en compresión libre de errores para eliminar la redundancia de la codificación son:
  - Huffman truncado: Se codifican con longitud variable sólo los símbolos más probables, y el resto se codifican con longitud fija.
  - Aritmético: Se codifica todo el mensaje (y no símbolo a símbolo) como un número originado a partir de enlazar las probabilidades de los símbolos

Compresión libre de errores

- LZW (Lempel-Ziv-Welch) es una técnica para eliminar la redundancia entre píxeles.

- Es la usada en TIFF, GIF o PDF.

- Publicada en:

WELCH, T A "*A technique for high-performance data compression*" Computer. Vol. 17, pp. 8-19. June 1984

LZW

- Se adapta de forma dinámica a la redundancia de la información a comprimir.
- Es reversible
- No tiene pérdida de información
- La compresión es transparente, *i.e.* La computadora no tiene conocimiento de que existe compresión

LZW

- La idea es relativamente simple: sustituir cadenas de información por *tokens* de menor longitud

- Cada vez que hay una substitución por algo de menor longitud se da una compresión

LZW

- Ejemplo:

  “La mantis verde se come la hoja verde del árbol verde”

  Se sustituye por

  “La mantis t1 se come la hoja t1 del árbol t1” siendo t1=[verde]

LZW

- LZW requiere de un diccionario de datos para ir guardando la información de las cadenas substituidas

- Este diccionario no es necesario enviarlo ya que se puede reconstruir en el destino.

LZW
Algoritmo de compresión

- STRING = get input character
- WHILE there are still input characters DO
  - CHARACTER = get input character
  - IF STRING+CHARACTER is in the string table then
    - STRING = STRING+character
  - ELSE
    - output the code for STRING
    - add STRING+CHARACTER to the string table
    - STRING = CHARACTER
  - END of IF
- END of WHILE
- output the code for STRING

Compresión libre de errores

- Otra técnica para eliminar la redundancia entre píxeles es la descomposición o codificación por plano de bit.

- Usada en JPEG y FAX

- **[Andra2001] "Efficient VLSI Implementation of Bit Plane Coder of JPEG2000"**
  K. Andra, T. Acharya and C. Chakrabarti
  Proc. of SPIE Applications of Digital Image Processing, 2001

Descomposición por plano de bit

- La imagen se descompone en planos de bit
  - Ejemplo: Si tenemos profundidad de 8 bits, obtenemos 8 planos, uno para cada bit.
- Los distintos planos son imágenes binarias que se comprimen de forma separada.

Descomposición por plano de bit

- Existen diferentes variantes para codificar/ comprimir los diferentes planos de bits:
  - Codificación de área constante
  - White block skipping
  - 1D run length coding - Standard para FAX
  - 2D run length coding
  - Block Truncation
  - Otras…

IV. Compresión con pérdidas

Compresión con pérdidas

- Se pierde precisión en la información para ganar en compresión.
- Una vez que se aplica un algoritmo de compresión con pérdidas ya no es posible recuperar la información original intacta.

Compresión con pérdidas

- Es quizás la más habitual
- La pérdida debe ser tolerable en el sentido que la información aún sea “entendible”
- Normalmente atacan a la redundancia psicovisual.

Compresión con pérdidas

- Se consideran 2 formas básicas de compresión con pérdidas:
  - Por codecs de transformación
    - La información de transforma para simplificarla y luego se comprime sin pérdidas
  - Por codecs predictivos
    - La información se estudia para predecir su comportamiento.
    - Se compara la predicción con la realidad y se codifica el error