

# Tema 13.

# Convolucionales

**Razonamiento y Representación del Conocimiento**

# Introducción

- Partimos de redes neuronales (NN) para realizar el aprendizaje automático
- Aprender es sinónimo de encontrar una frontera en un espacio  $k$ -dimensional que separe los elementos de clases diferentes
- Con un número de dimensiones bajo, las NN funcionan satisfactoriamente

# Introducción

- Si la entrada a una NN es una imagen, la dimensión del espacio de soluciones es el número de píxeles de la imagen:
  - En  $64 \times 64 \rightarrow 4096$  dimensiones
  - En  $100 \times 100 \rightarrow 10000$  dimensiones
  - En  $1024 \times 768 \rightarrow 7.86 \times 10^6$

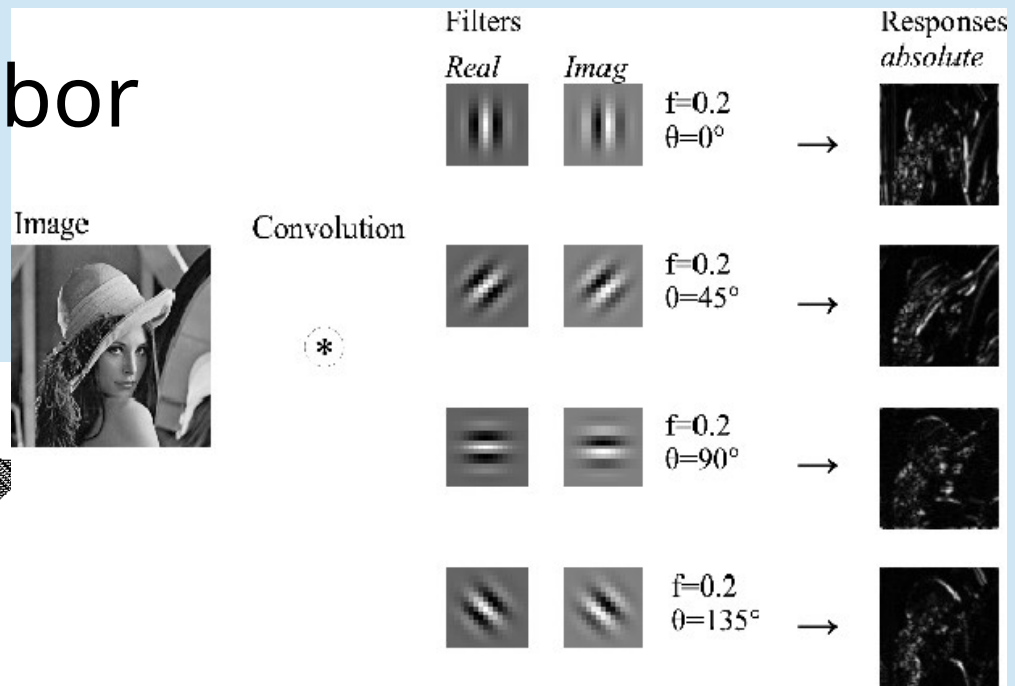
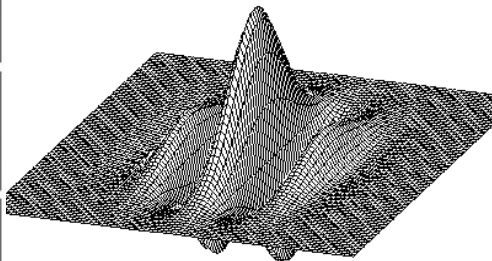
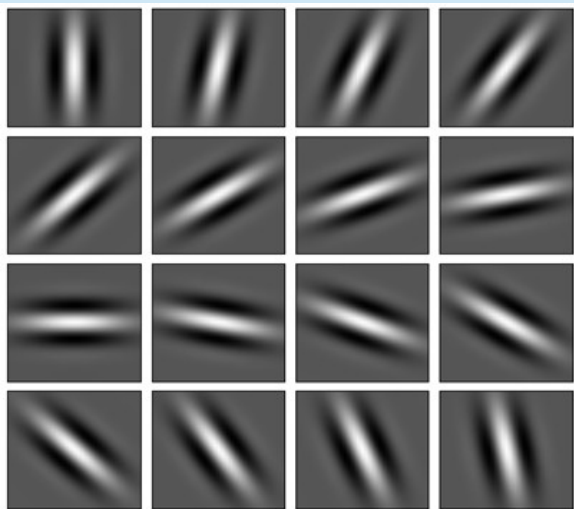
Inmanejable!  $\rightarrow$  la maldición de la dimensionalidad

# Filtrado de imagen

- Necesitamos reducir la información que aparece en las imágenes
- Idea: filtrar las imágenes para destacar en ellas las características que nos permitan reconocer los objetos que aparecen en ellas

# Filtrado de imagen

- Ejemplo: filtros de Gabor



Ejemplo: filtro de Canny



# Filtrado de imagen

- Primeros filtros de imagen
  - Extraen características que nos parecen relevantes:
    - Bordes, puntos esquina, cambios de textura, etc.
  - Problema: no se consigue mejorar de forma significativa los resultados del aprendizaje al utilizar estos filtros

# Filtrado de imagen

- Los filtros: ¿Por qué no funcionan?
- ¿Seguro que no funcionan?
- ¿Qué filtros funcionan mejor?
- ¿Y si... en vez de elegir los filtros 'a mano' dejamos que sea el algoritmo el que decida qué filtros son los mejores?

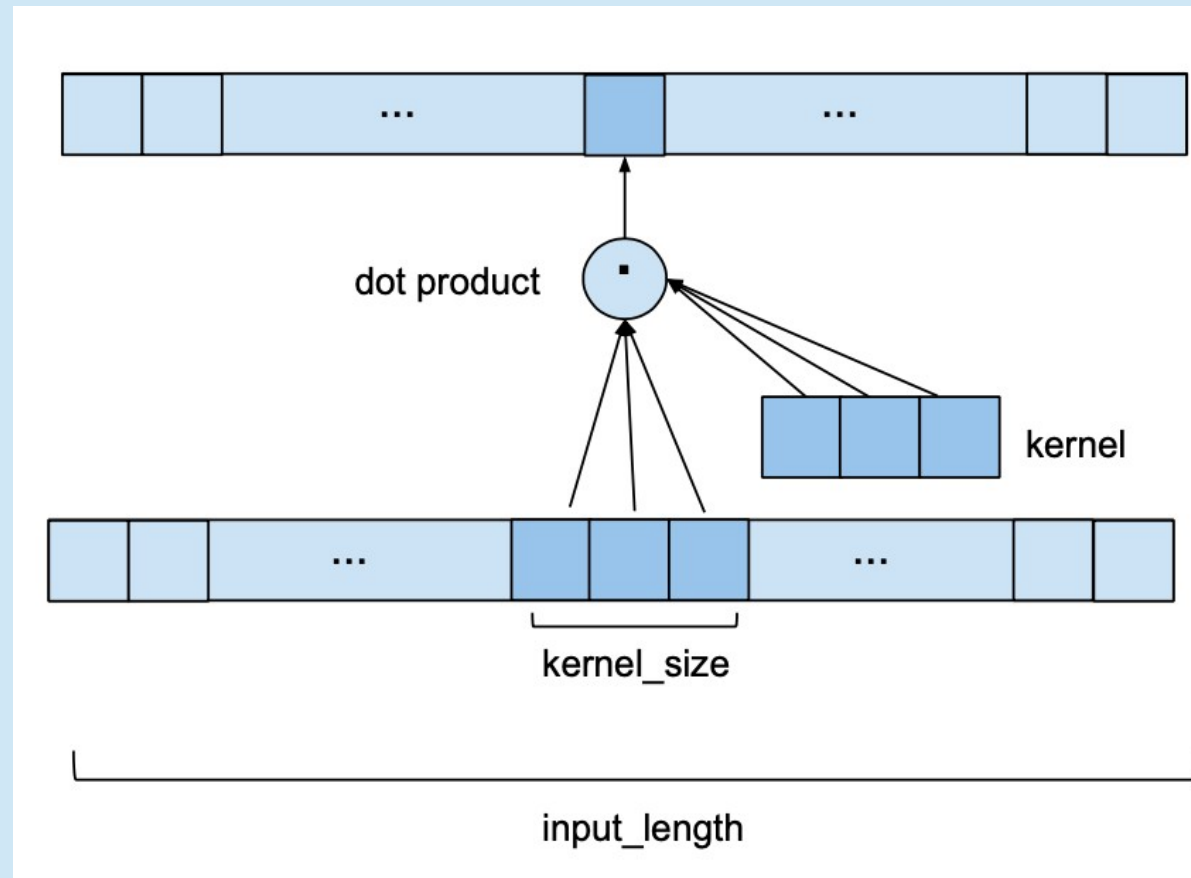
# Convoluciones

- Convoluciones en imágenes:
  - Convoluciones 2D
- Convolución: combinación lineal de los píxeles de una imagen ponderados por los valores de un filtro de convolución (kernel)
- ¿Nos suena?



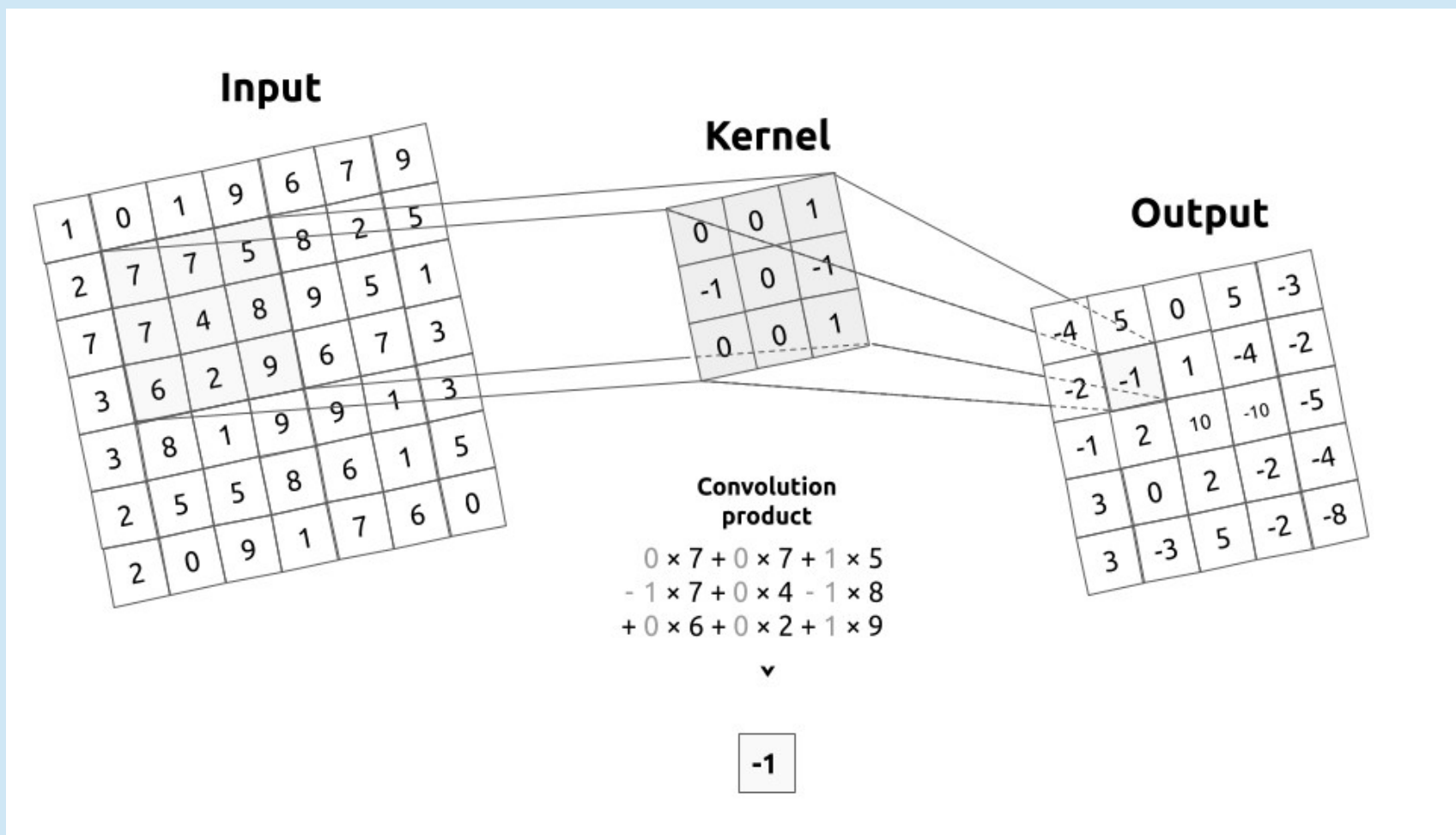
# Convoluciones

- ¿Cómo funcionan?
  - Ejemplo 1D
  - Entrada y kernel son vectores
  - Resultado: otro vector



# Convoluciones

- 2D



# Capas Convolucionales

- Partimos del concepto de una fully connected
  - Una imagen es una matriz
  - Necesitamos 'aplanarla'
    - Tendremos un vector ( $w \times h$ ) en el que la imagen está almacenada por filas
    - Un elemento del vector a  $+w$  posiciones de otro es el pixel inmediatamente inferior en la imagen

# Capas Convolucionales

- Capa Fully connected – Ejemplo 4x4

$$\mathbf{x} = [x_1, x_2, x_3, x_4]$$

Capa con 4 neuronas:

$$\begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \end{bmatrix} = \begin{bmatrix} W_{11} & W_{12} & W_{13} & W_{14} \\ W_{21} & W_{22} & W_{23} & W_{24} \\ W_{31} & W_{32} & W_{33} & W_{34} \\ W_{41} & W_{42} & W_{43} & W_{44} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

# Capas Convolucionales

- Concepto de capa local
  - Parche: dado un pixel de la imagen  $(i,j)$ , el conjunto de pixeles que están a una distancia menor o igual a  $k$
  - Capa local  $\rightarrow$  capas en las que un pixel en la salida solo se ve afectado por un parche de un tamaño determinado

# Capas Convolucionales

- Concepto de capa local:
  - Hacemos 0 los pesos de la capa que no están dentro del área de influencia de un pixel

$$\begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \end{bmatrix} = \begin{bmatrix} W_{12} & W_{13} & 0 & 0 \\ W_{21} & W_{22} & W_{23} & 0 \\ 0 & W_{31} & W_{32} & W_{33} \\ 0 & 0 & W_{41} & W_{42} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

# Capas Convolucionales

- Operamos también con los bordes

$$\begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \end{bmatrix} = \begin{bmatrix} W_{11} & W_{12} & W_{13} & 0 & 0 & 0 \\ 0 & W_{21} & W_{22} & W_{23} & 0 & 0 \\ 0 & 0 & W_{31} & W_{32} & W_{33} & 0 \\ 0 & 0 & 0 & W_{41} & W_{42} & W_{43} \end{bmatrix} \begin{bmatrix} 0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ 0 \end{bmatrix}$$

# Capas Convolucionales

- Concepto de equivarianza a la traslación
  - Una convolución debe dar el mismo resultado si se aplica en dos zonas de la imagen con los mismos píxeles

$$\begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \end{bmatrix} = \begin{bmatrix} W_2 & W_3 & 0 & 0 \\ W_1 & W_2 & W_3 & 0 \\ 0 & W_1 & W_2 & W_3 \\ 0 & 0 & W_1 & W_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

Matriz toeplitz



# Capas Convolucionales

- Matrices Toeplitz
  - Permite realizar la operación de convolución de una imagen con un kernel de forma eficiente
  - La función que se aplica sigue siendo una combinación lineal de las entradas con los pesos de la capa (kernel)

# Capas Convolucionales

- El efecto de usar dos filtros convolucionales consecutivos  $\rightarrow$  igual que uno de mayor tamaño de kernel
- Si el número de filtros consecutivos es grande  $\rightarrow$  capa fully connected
- Solución: incluir una función de activación como las de las neuronas tras la convolución

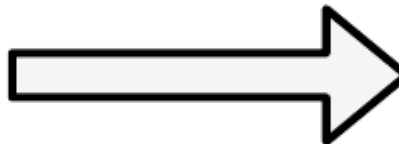
# Reducción de tamaño

- Las convoluciones no cambian el tamaño de la entrada
  - Suele convenir ir reduciendo el tamaño antes de llegar a la capa fully-connected
  - Tratamos de respetar la distribución espacial
    - Suma de los valores de una región
    - Máximo valor de una región ←

# Capas Max-Pooling

- Reducen el tamaño de la entrada a la mitad

3.2	-1.5	2.7	0.5
0.2	0.7	-1.8	3.0
0.4	1.3	1.25	-0.6
-2.0	0.1	-0.8	1.0

Max-pooling  


3.2	3.0
1.3	1.25

# Arquitectura de una CNN

- Generalmente alternaremos
  - Varios filtros convolucionales
  - Una capa max-pooling
- Finalmente: una capa fully-connected

# Arquitectura de una CNN

- Ejemplo: Clasificador de 10 clases

