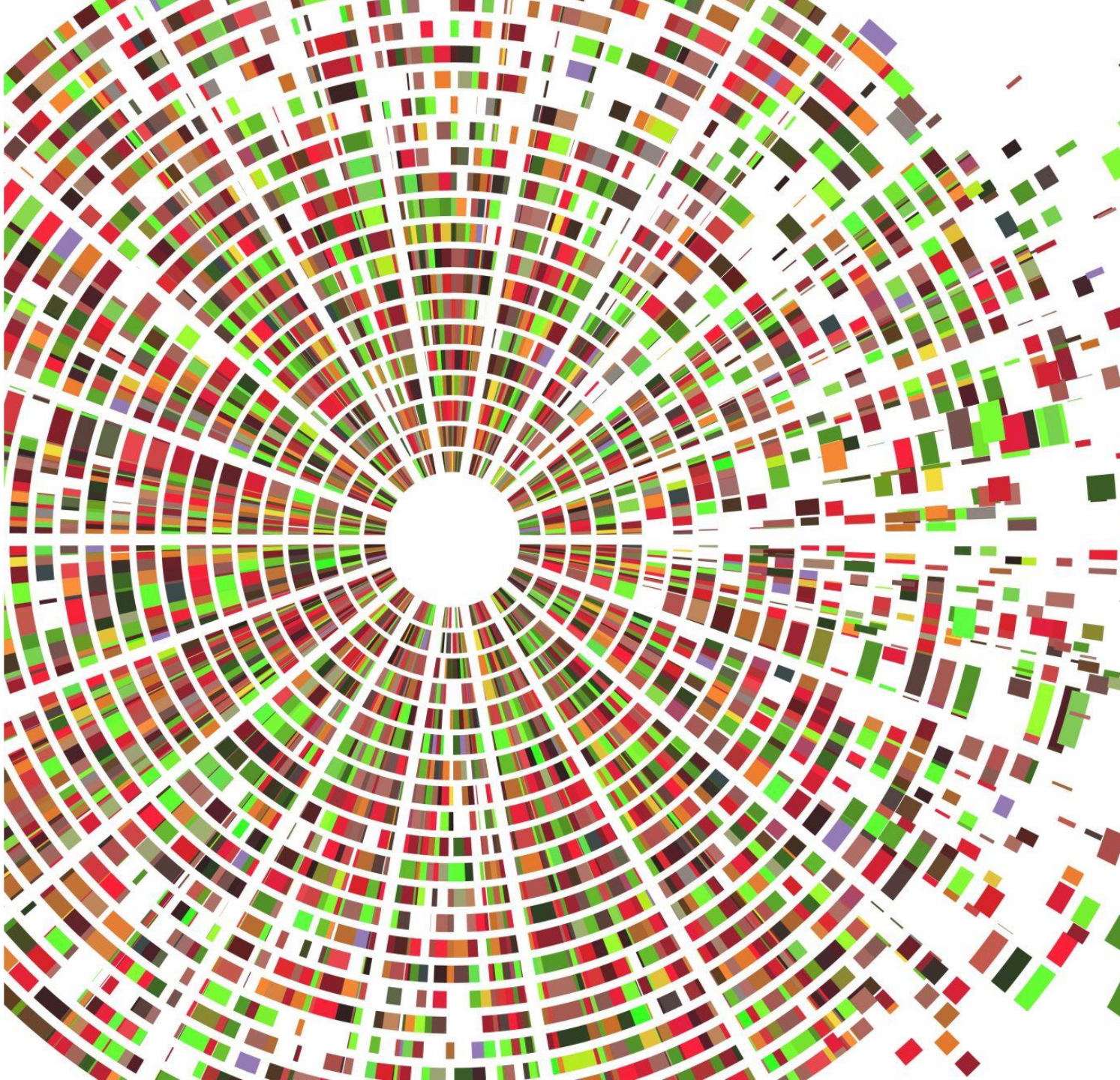
Métodos avanzados de ciencia de datos

Prof. Emily Díaz



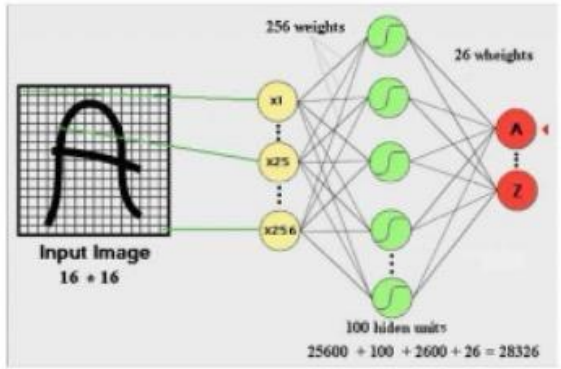
Contenido

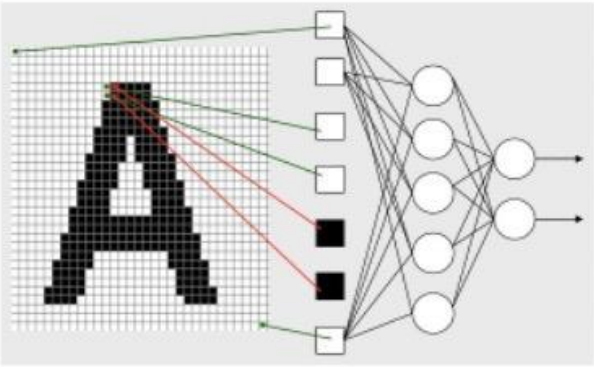
Conceptos básicos de redes neuronales convolucionales

Arquitecturas populares de CNN

Aumento de datos

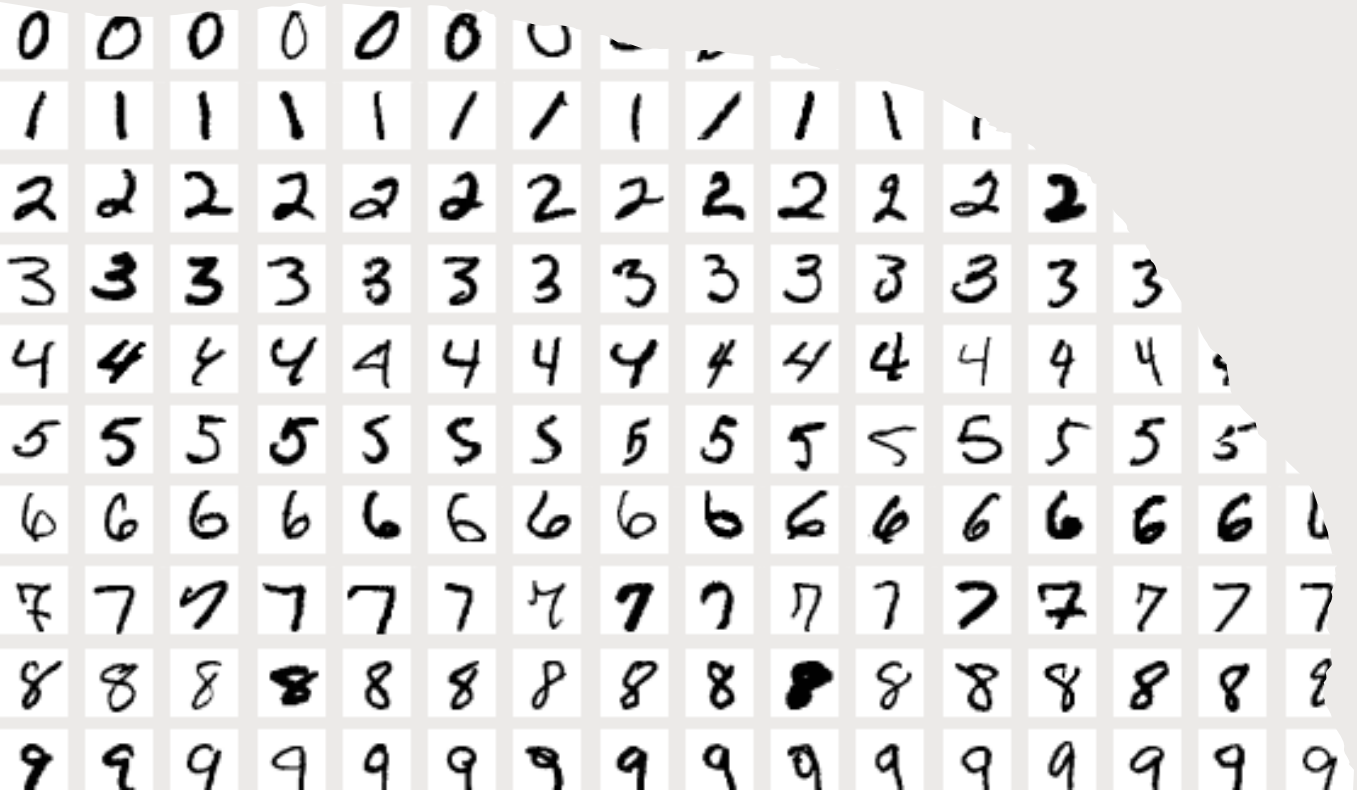
Motivación: Reconocimiento de caracteres





- Utilización de MLP es inapropiada por:
  - Alto número de parámetros de entrada
  - Alto número de coeficientes
  - No son robustas a cambios como distorsiones, re-escalamiento
  - No aprovechan la estructura espacial de las imágenes. Por ejemplo, en las imágenes, los píxeles cercanos suelen estar relacionados, pero las MLP no tienen en cuenta estas relaciones espaciales
- Para el caso de reconocimiento de caracteres, se empezó a plantear cómo solucionar esto de mejor manera

LeNet-5, el inicio de Redes Neuronales Convolucionales (CNN)



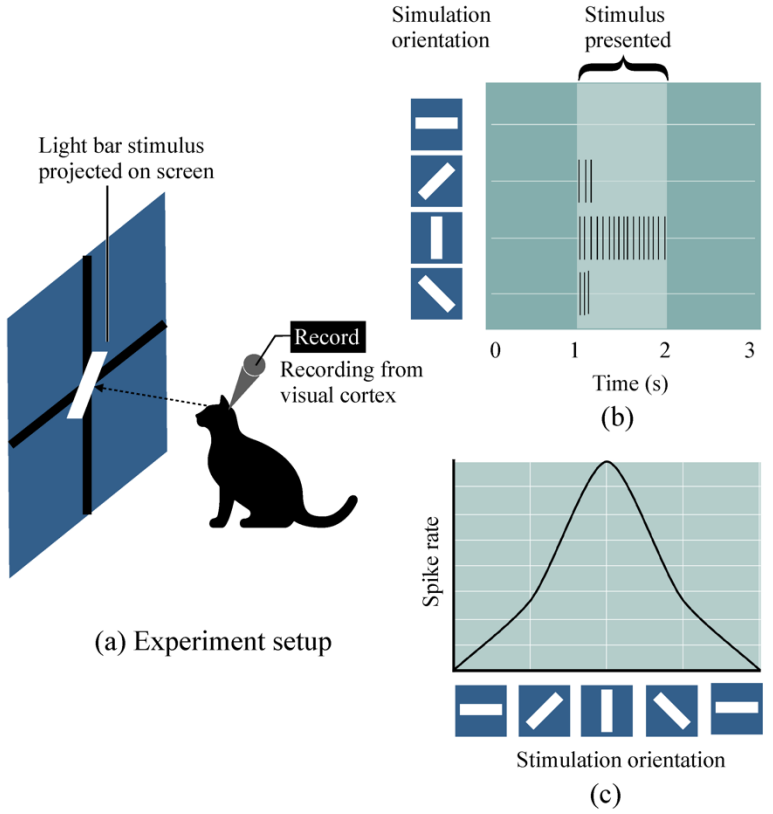
- A finales de los años 1980 y principios de los años 1990, Yann LeCun desarrolló la arquitectura LeNet-5, una de las primeras CNN.
- Fue diseñada para abordar el problema del reconocimiento de dígitos escritos a mano.
- El objetivo era clasificar los dígitos en el famoso conjunto de datos MNIST (una colección de números escritos a mano del 0 al 9) que se utiliza en tareas como el reconocimiento de códigos postales.

Ventajas de usar CNN

- La arquitectura de las redes convolucionales brindan 3 ventajas clave para el análisis de imágenes:
  1. **Conectividad local:** Con capas convolucionales en donde los filtros se mueven a través de **pequeñas secciones de la imagen**, se capturan características locales (**bordes, curvas, texturas, etc)**
  2. **Uso compartido de parámetros:** El **mismo filtro se aplica a toda** la imagen, lo cual **reduca la cantidad de parámetros** y hace la red más eficiente
  3. **Invariancia de la traslación:** Al enfocarse en **patrones locales**, son resistentes a **cambios o distorsiones pequeñas en los caracteres o objetos de interés en la imagen**. Esto es muy útil cuando por ejemplo existen variaciones en estilos de escritura a mano

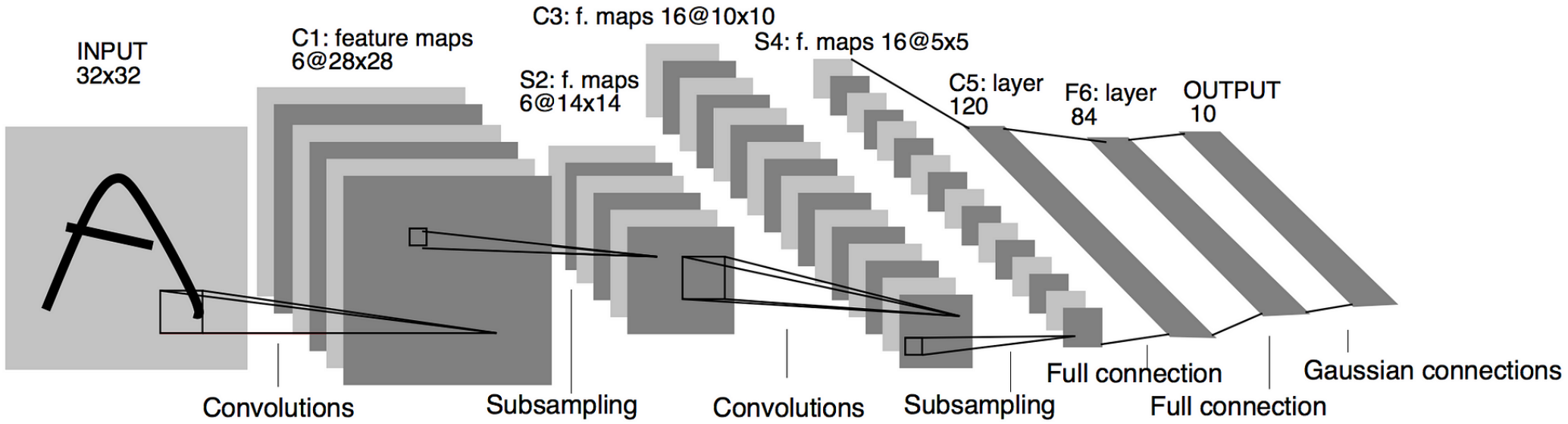
Qué son los CNN?

- Yann LeCun (considerado el padre de las CNN) **desarrolló LeNet, la primera CNN.**
- Fue inspirado por los descubrimientos de David Hubel y Torsten Wiesel en **cómo el cerebro procesa la información visual, particularmente en su investigación sobre la corteza visual de los gatos**
- Algunos de los conceptos que más impactaron las redes neuronales artificiales son:
  - **Estructura jerárquica:** las neuronas de la corteza visual están organizadas jerárquicamente, donde **las neuronas de nivel inferior detectan características simples como bordes y orientaciones, y las neuronas de nivel superior combinan estas características** para detectar patrones más **complejos**
  - **Campos receptivos:** las neuronas de la corteza visual responden a estímulos en regiones específicas del campo visual, conocidas como campos receptivos. **Algunas neuronas responden a estímulos simples como bordes, mientras que otras responden a formas o movimientos más complejos.**
  - **Invariancia de localidad y desplazamiento:** se descubrió que las neuronas eran sensibles a características en **áreas locales del campo visual** y podían detectar patrones (como bordes) **incluso si el patrón estaba ligeramente desplazado o distorsionado.**



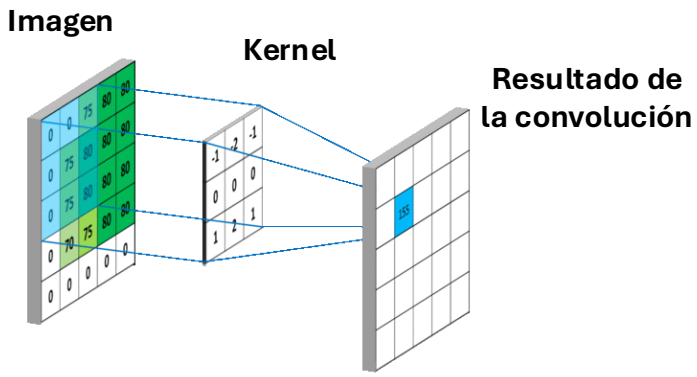
LeNet

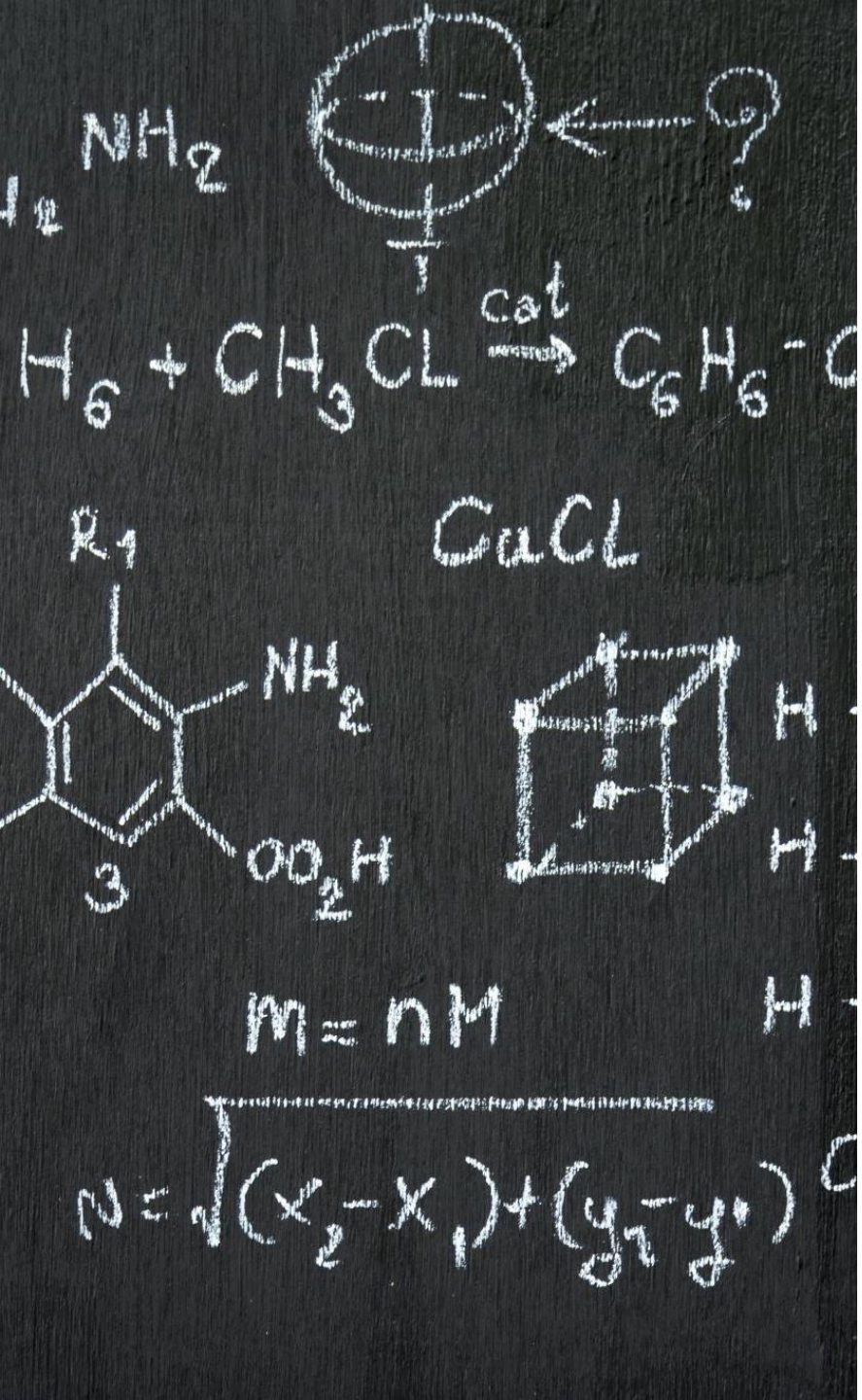
- LeNet-5, reflejó estos principios biológicos en su arquitectura:
  - Las capas **convolucionales** actúan como las células simples de la corteza visual, **detectando patrones básicos como los bordes**.
  - Las capas de agrupación (***pooling***) realizan un **muestreo descendente**, de forma similar a cómo las células complejas del cerebro integran información de células simples, lo que hace que el sistema sea **más robusto a los cambios de posición y escala**.
  - **Las capas completamente conectadas** al final **combinan las características aprendidas** para hacer predicciones finales, de **forma similar a las regiones corticales de nivel superior** que combinan las características visuales en un todo coherente.



Qué es una operación de convolución?

- Es una operación matemática para extraer **variables/características** de datos como imágenes.
- Se aplica como un filtro pequeño, llamado *kernel*, deslizándolo sobre la imagen y calculando una suma ponderada del filtro y la porción de la imagen que superpone.





Qué es una operación de convolución?

- Pasos:
  1. **Filtro o kernel:** Se define una pequeña matriz (3x3, 5x5, etc) que se aplicará a la imagen. Este kernel es un set de coeficientes que aprenderá la red y se inicializan de manera aleatoria. Al igual que MLP, en cada epoch/periodo de entrenamiento, se actualizan los valores de los coeficientes en pos de la minimización de la función de pérdida.
  2. **Deslizamiento sobre la imagen:** El filtro se aplica a la imagen deslizándolo por cada píxel de la imagen, moviéndose un píxel a la vez (paso/***stride***), y en cada paso, calcula el producto escalar de los valores del filtro y los valores de los píxeles correspondientes en la imagen.
  3. **Valor resultante o mapa de características (*feature map*):** El resultado del producto escalar para cada posición se almacena como un único número en una nueva matriz (llamada mapa de características). Este proceso **se repite para toda la imagen y el mapa de características captura el grado de coincidencia de determinados patrones** (como bordes, texturas, etc.) **con la imagen en diferentes ubicaciones**.

Ejemplo matemático

Imagen

1	**2**	**3**
4	**5**	**6**
7	**8**	**9**

Kernel

1	**0**
0	**-1**

1 De qué tamaño será la matriz resultante? Depende del paso y relleno que decidamos

Asumamos que en este caso no usamos relleno y usamos un paso de 0

La fórmula para calcular el tamaño del mapa de características resultantes es:

$$Mapa = \left(\frac{Tamaño\ original\ - Tamaño\ del\ kernel + 2 * relleno}{paso}\right) + 1$$

En este caso:

$$Mapa = \left(\frac{3 - 2 + 2 * 0}{1}\right) + 1 = 2$$

Entonces el mapa de característica final es de 2x2

Stride/Paso: Número de pixels que se mueve en un paso. Es 1 si se mueve al siguiente pixel en cada paso

Padding/Relleno: A veces, para conservar el tamaño de la entrada, se aplica un relleno. El relleno agrega filas y columnas adicionales de ceros alrededor del borde de la imagen. Sin relleno, el mapa de características de salida será más pequeño que el de entrada, pero con relleno, las dimensiones pueden permanecer iguales.

Ejemplo matemático

Imagen

1	2	3
4	5	6
7	8	9

Kernel

1	0
0	-1

2 Cálculo del primer valor

1*1 + 2*0 + 4*0 + 5*-1 = -4	

Ejemplo matemático

Imagen

1	2	3
4	5	6
7	8	9

Kernel

1	0
0	-1

3

Cálculo del segundo valor

-4	2*1 + 3*0 + 5*0 + 6*-1 = -4

Ejemplo matemático

Imagen

1	2	3
4	5	6
7	8	9

Kernel

1	0
0	-1

4 Cálculo del tercer valor

-4	-4
4*1 + 5*0 + 7*0 + 8*-1 = -4	

Ejemplo matemático

Imagen

1	2	3
4	5	6
7	8	9

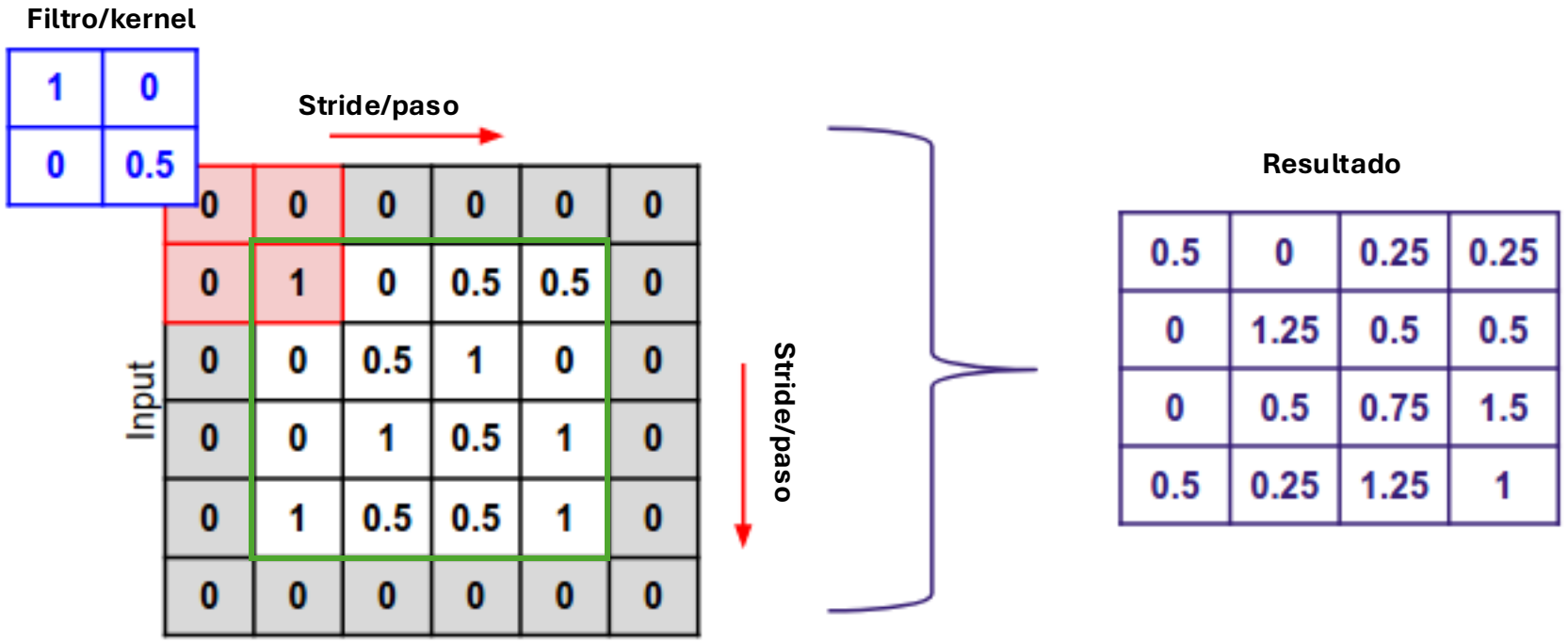
Kernel

1	0
0	-1

4 Cálculo del cuarto valor

-4	-4
-4	5*1 + 6*0 + 8*0 + 9*-1 = -4

Con padding/relleno



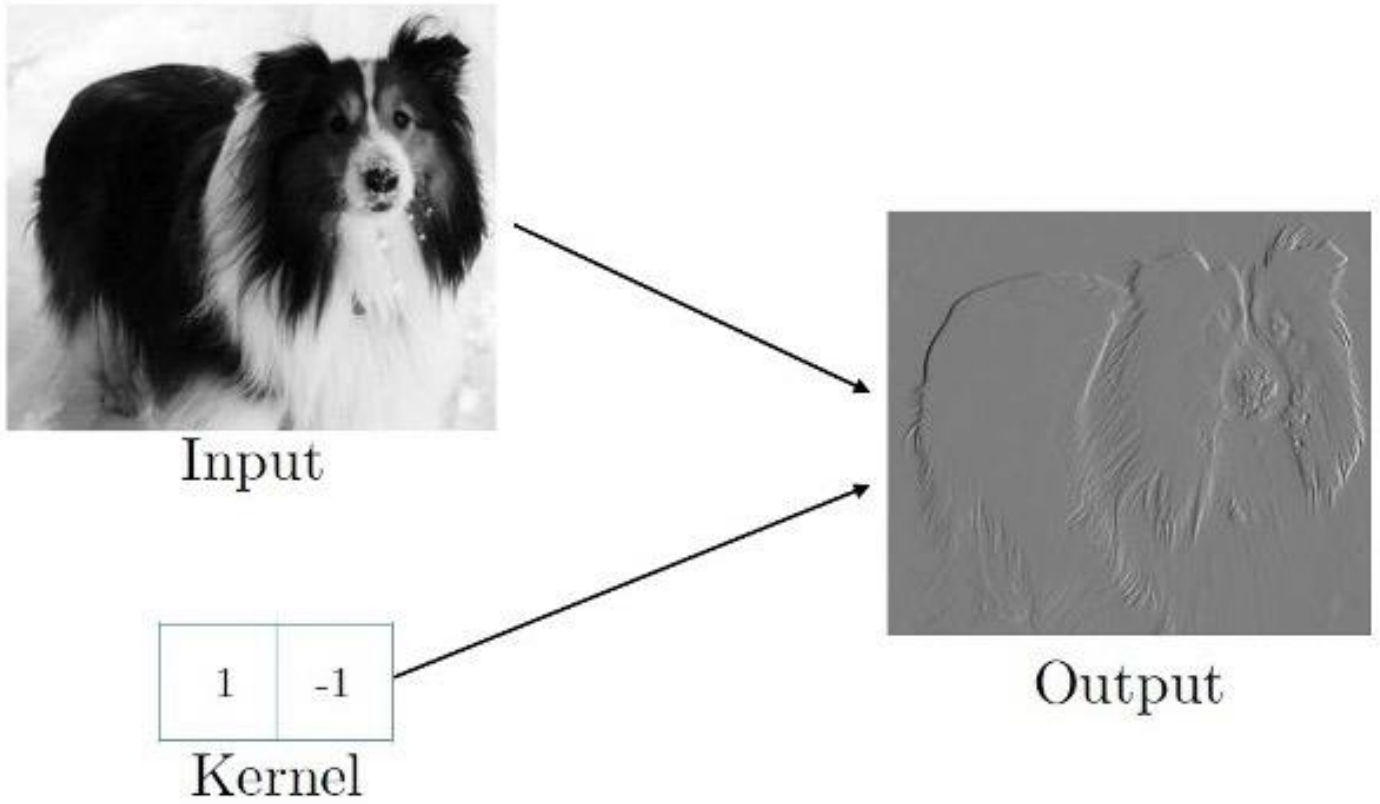
- Añadimos bordes de 0 como relleno para preservar el tamaño de la imagen y no perder información de los bordes.
- El tamaño del relleno depende del tamaño del kernel y del stride/paso. Para que quede el mapa del mismo tamaño que el original, la fórmula es:

$$P = \frac{S * (H - 1) - H + F}{2}$$

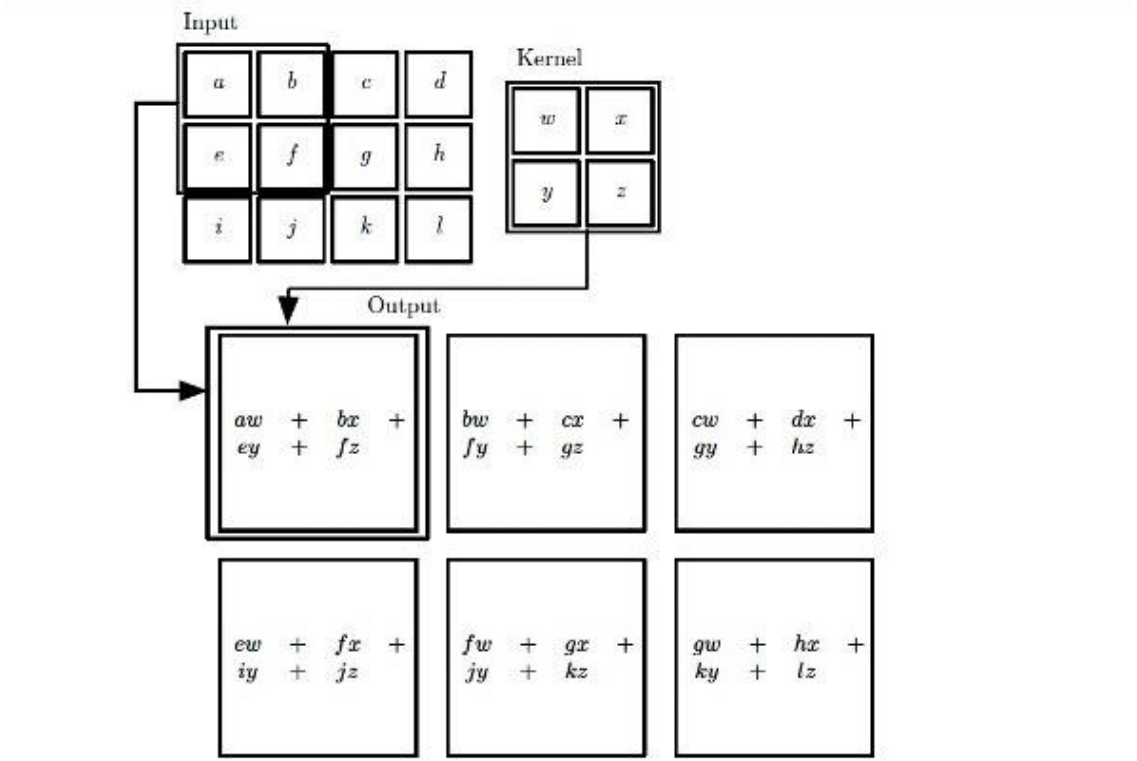
- Donde H es la altura (asumiendo que la imagen es cuadrada), F el tamaño del filtro y S el stride

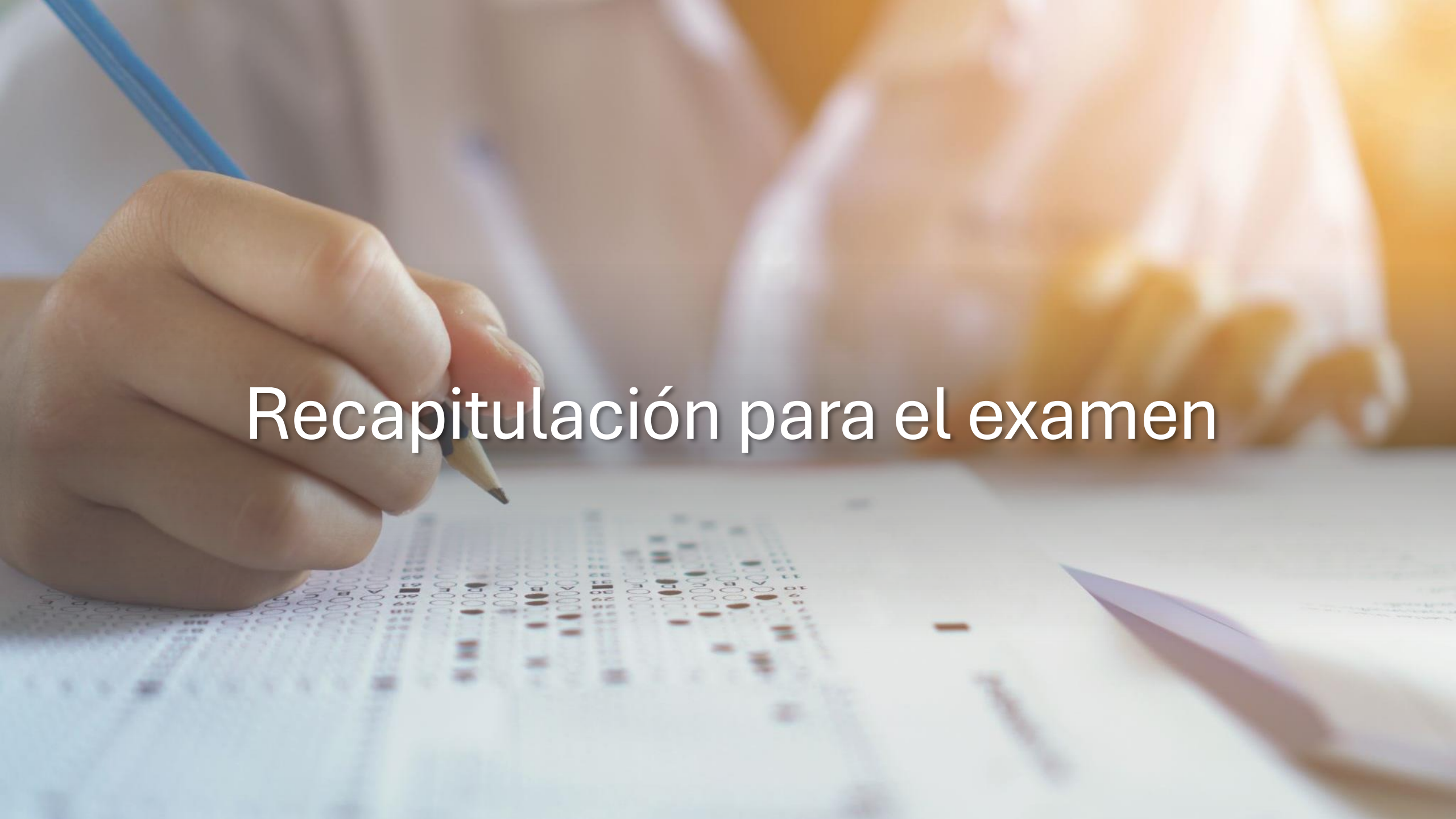
Imagen original

Los filtros sirven como detectors de distintas características dependiendo de sus coeficientes



Convolución en 2D- general

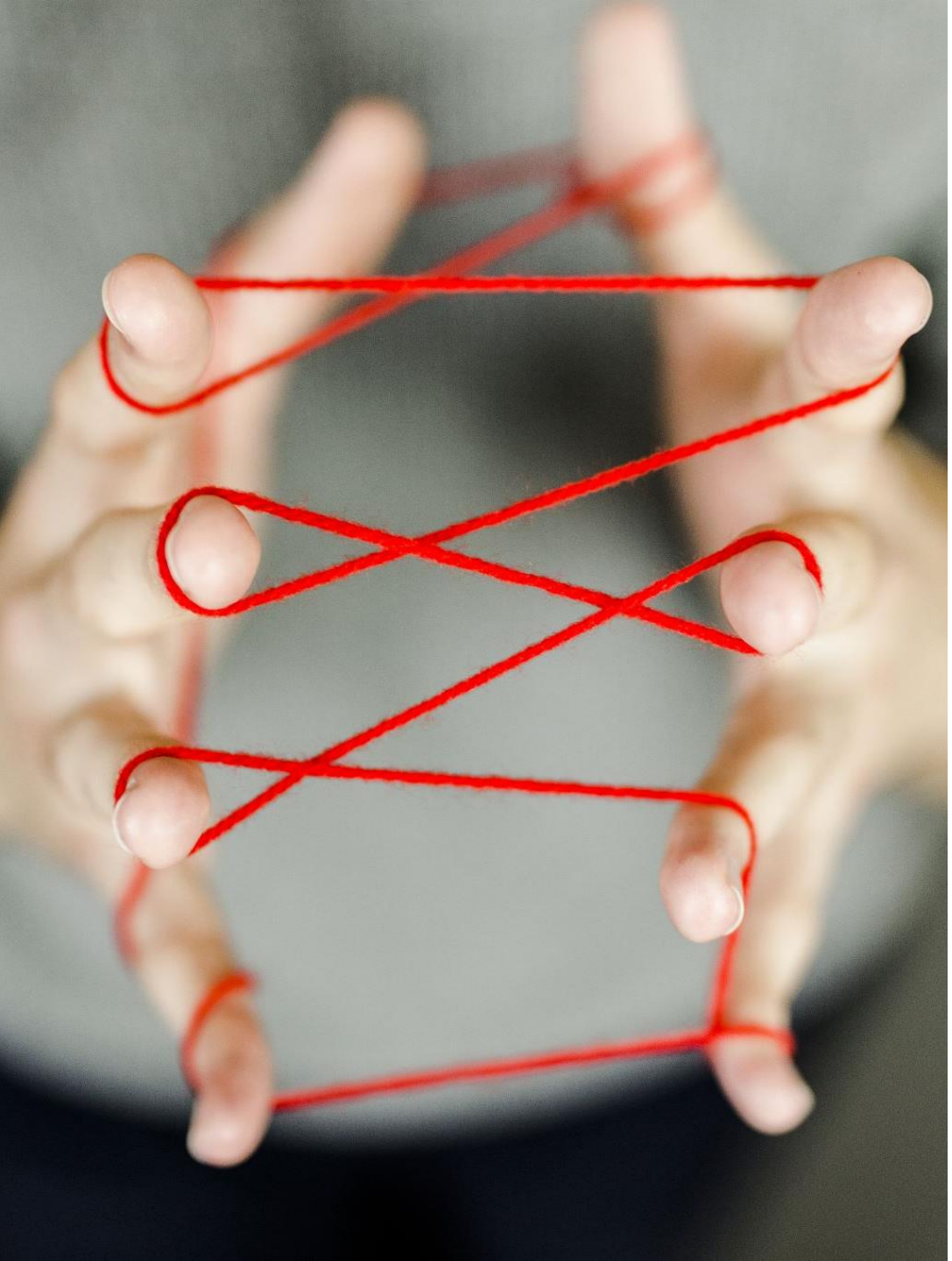




Recapitulación para el examen

Temas que entran

- Todo lo visto antes de entrar a la presentación de la semana 6 de imágenes.
  - Introducción a ciencia de datos, aprendizaje supervisado, aprendizaje no supervisado y aprendizaje reforzado
  - Introducción a redes neuronales: Historia, motivación y casos de uso
  - Conceptos básicos de redes neuronales: neuronas, capas, tipos de redes neuronales
  - Propagación hacia delante
  - Funciones de activación
  - Funciones de pérdida.
  - Optimización y propagación hacia atrás
  - Metricas de rendimiento
  - Sobreajuste y balance entre sesgo y varianza
  - Métodos de regularización
  - Visualización e interpretabilidad
  - Paralelización y poder computacional necesario
  - Modelos pre-entrenados



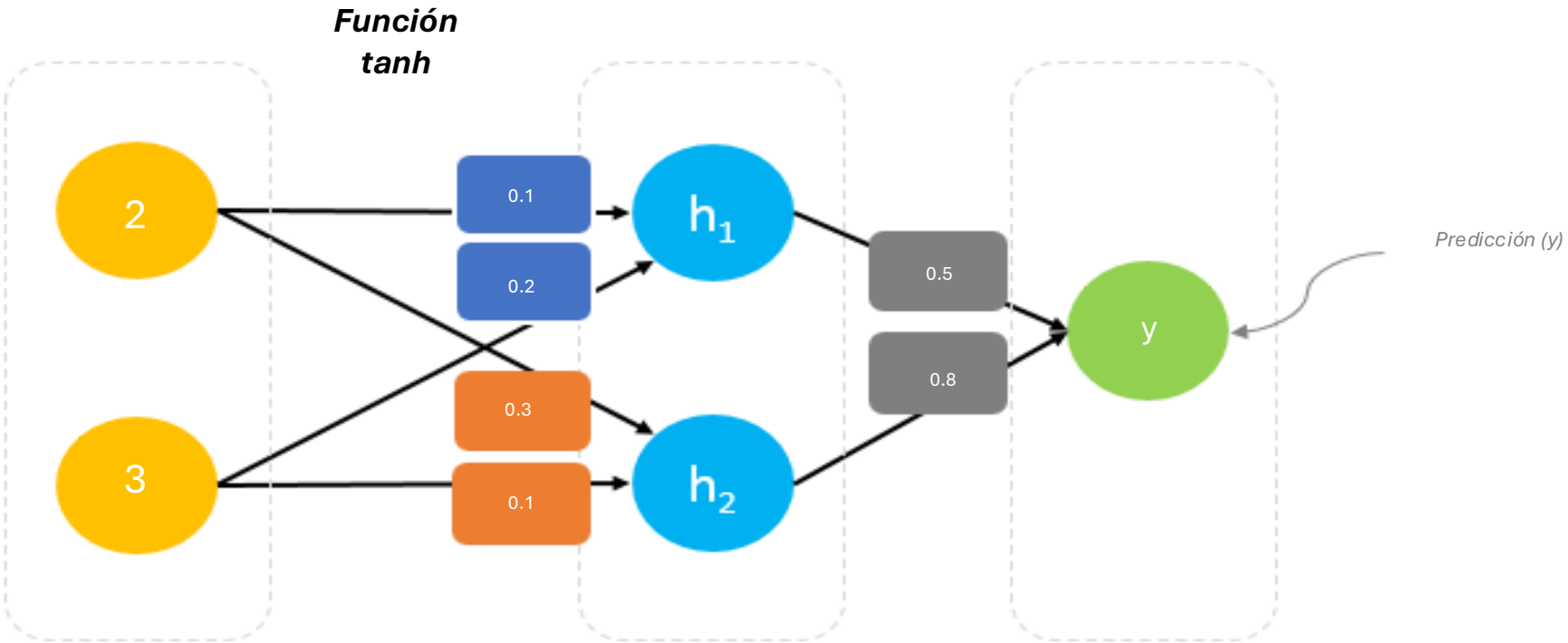
Pasos de construcción de una red neuronal

La red se inicializa con coeficientes de valor aleatorio. Para entrenarla se siguen los siguientes pasos:

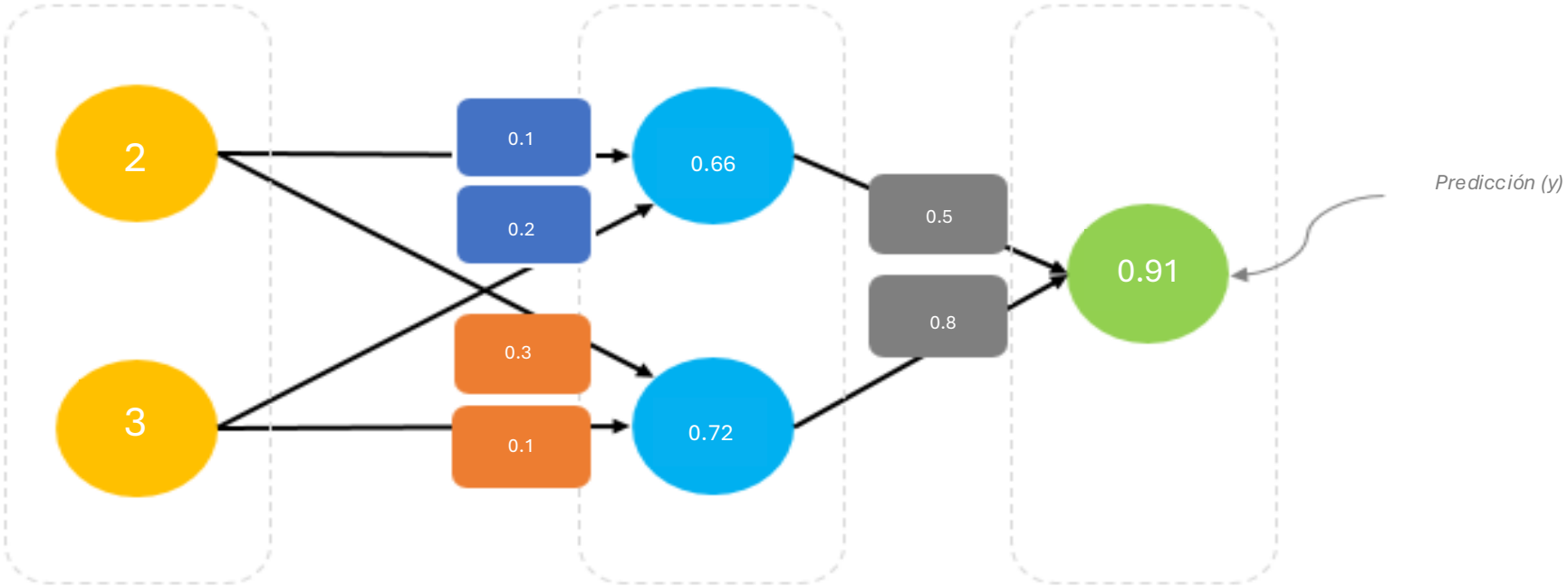
1. **Propagación hacia delante (Forward pass):** Los datos de entrada se pasan a través de la red neuronal, capa por capa, hasta obtener el resultado final (predicción)
2. **Cálculo de pérdida:** La pérdida o error se calcula comparando la salida de la red neuronal con el valor real (target). Esta pérdida cuantifica cuán lejos está la predicción de la red del valor esperado.
3. **Propagación hacia atrás (Backward propagation)**: Se calcula el gradiente de la pérdida respecto a cada peso en la red neuronal, utilizando el algoritmo de retropropagación. Este gradiente se usa para actualizar los pesos.
4. **Optimización:** Los pesos de la red se ajustan en función de los gradientes calculados. Este paso generalmente implica el uso de un optimizador, como el descenso de gradiente, que ajusta los pesos para minimizar la pérdida.

*Cada iteración de estos pasos se llama un **epoch**, y el proceso se repite muchas veces hasta que la red converja a una solución óptima (o suficientemente buena).*

Ejemplo



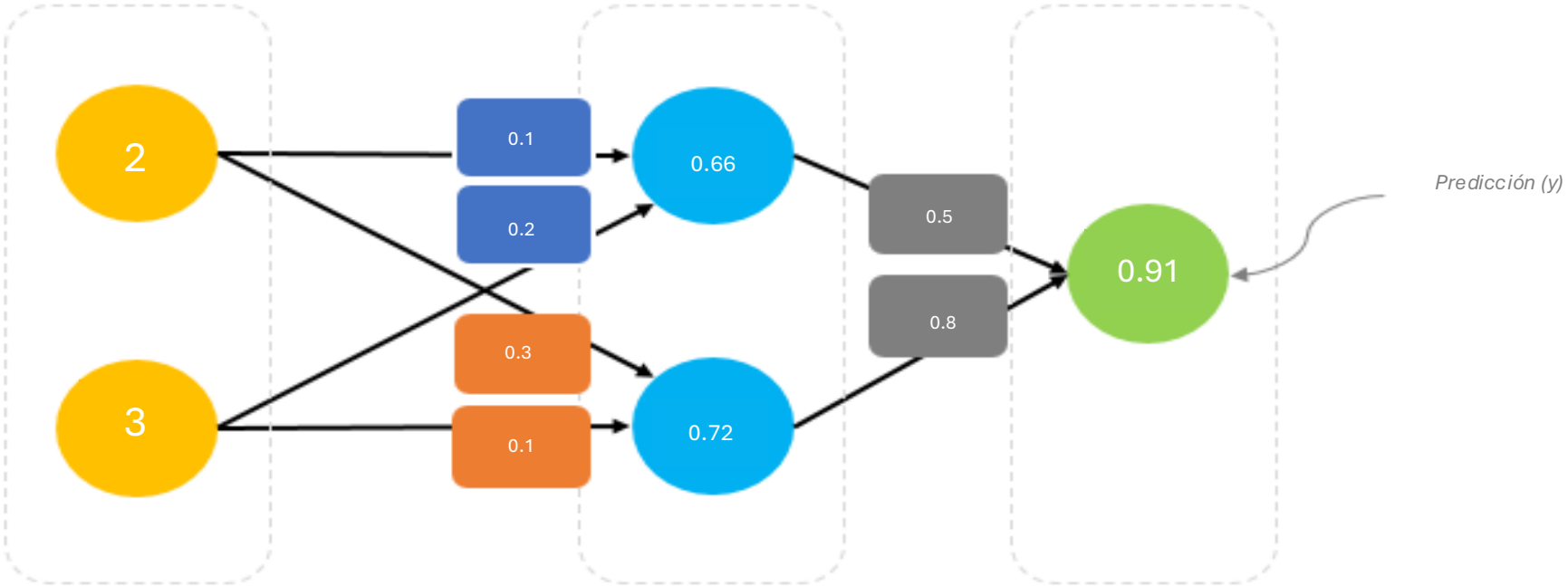
Propagación hacia delante



- h_1 = tanh(2*0.1 + 3*0.2) = tanh(0.8) = 0.66
- h_2 = tanh(2*0.3 + 3*0.1) = tanh(0.9) = 0.72
- y = 0.5*0.66 + 0.8 * 0.72 = 0.91

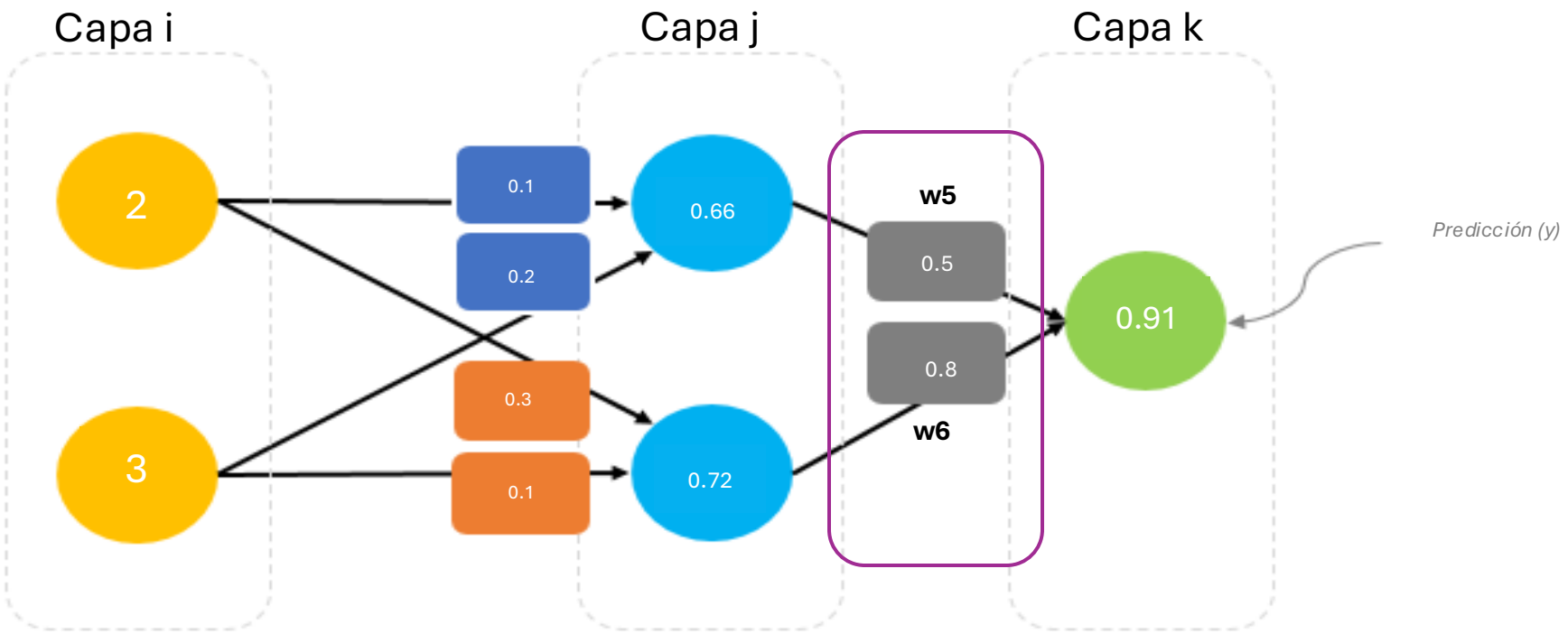
Cálculo del error

$$Error = \frac{1}{2}(\hat{y} - y)^2$$



- $Y = 1$
- $\hat{Y} = 0.91$
- $\frac{1}{2}(0.91 - 1)^2 = 0.00045$

Propagación hacia atrás



Cálculos en cada paso

$$h_j = \tanh\left(\sum_{j=1}^{p} w_{ij} * x_i\right)$$

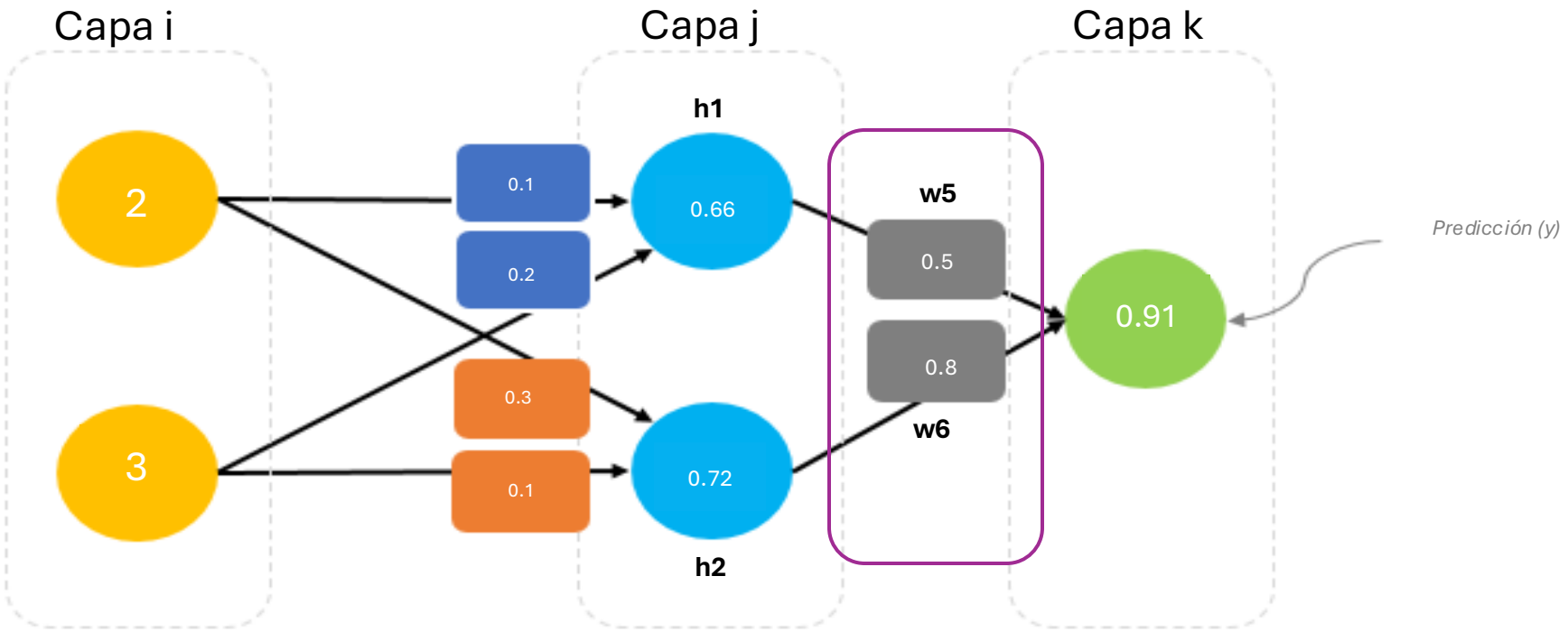
$$\hat{y} = \sum_{k=1}^{c} w_{jk} * h_j$$

$$\textbf{Error} = \frac{1}{2}(\hat{y} - y)^2$$

1

$$\frac{\partial Error}{\partial w5} = \frac{\partial Error}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial w5} = ?$$

Propagación hacia atrás



Cálculos en cada paso

$$h_j = \tanh\left(\sum_{j=1}^{p} w_{ij} * x_i\right)$$

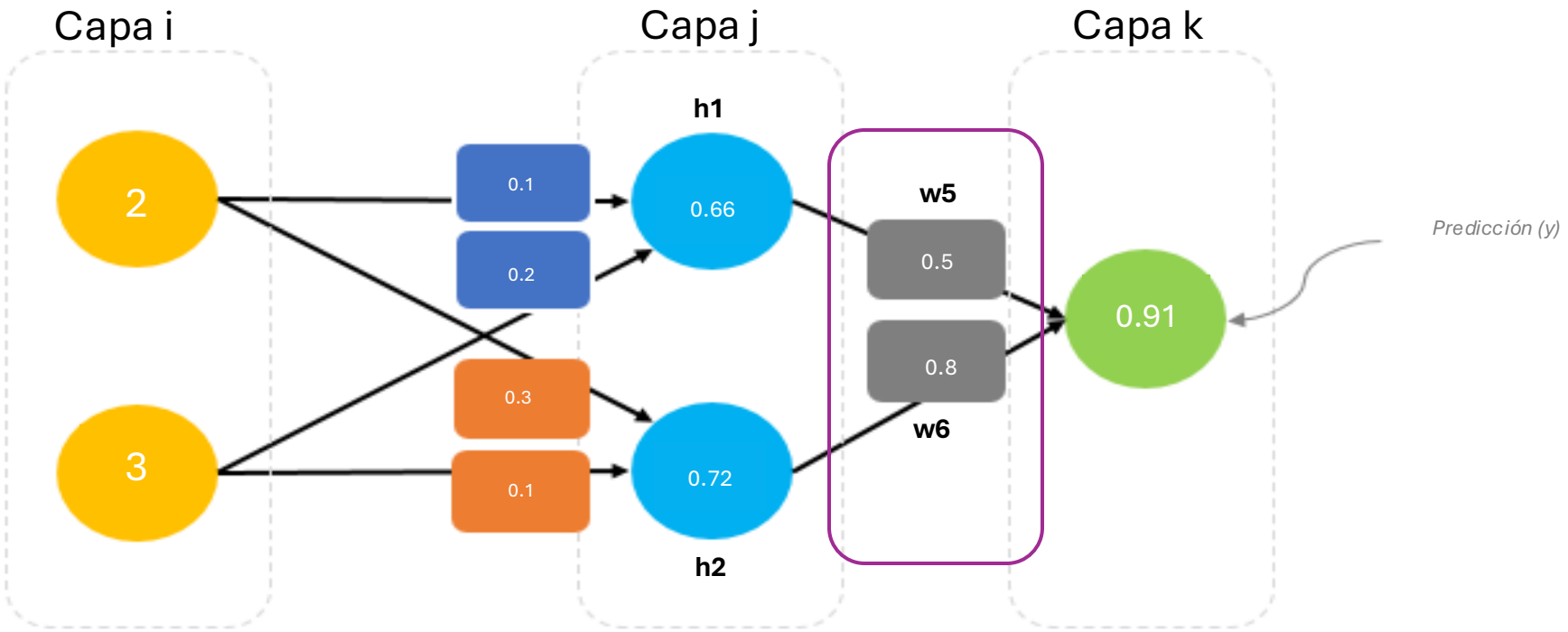
$$\hat{y} = \sum_{j=1}^{c} w_{jk} * h_j$$

$$\boldsymbol{Error} = \frac{1}{2}(\hat{\boldsymbol{y}} - \boldsymbol{y})^2$$

1

$$\frac{\partial Error}{\partial w5} = \frac{\partial Error}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial w5} = \left[2 * \left(\frac{1}{2}\right) * (\hat{\boldsymbol{y}} - \boldsymbol{y}) * \boldsymbol{1}\right][\boldsymbol{h_1}] = (0.91 - 1) * 0.66 = -0.0594$$

Propagación hacia atrás



Cálculos en cada paso

$$h_j = \tanh\left(\sum_{j=1}^{p} w_{ij} * x_i\right)$$

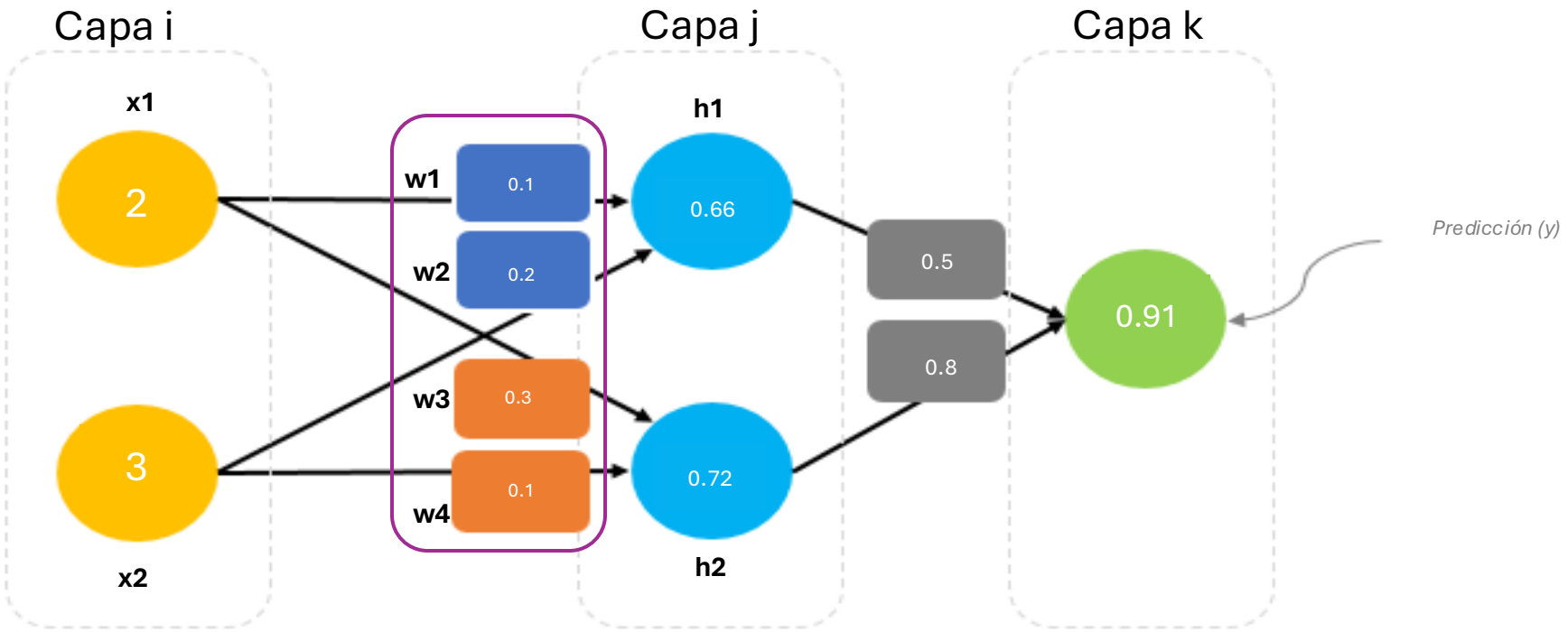
$$\hat{y} = \sum_{j=1}^{c} w_{jk} * h_j$$

$$\textbf{Error} = \frac{1}{2}(\hat{y} - y)^2$$

2

$$\frac{\partial Error}{\partial w6} = \frac{\partial Error}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial w6} = \left[2 * \left(\frac{1}{2}\right) * (\hat{y} - y) * \mathbf{1}\right][h_2] = (0.91 - 1) * 0.72 = -0.0648$$

Propagación hacia atrás



Cálculos en cada paso

$$h_j = \tanh(a_j) = \tanh\left(\sum_{j=1}^{p} w_{ij} * x_i\right)$$

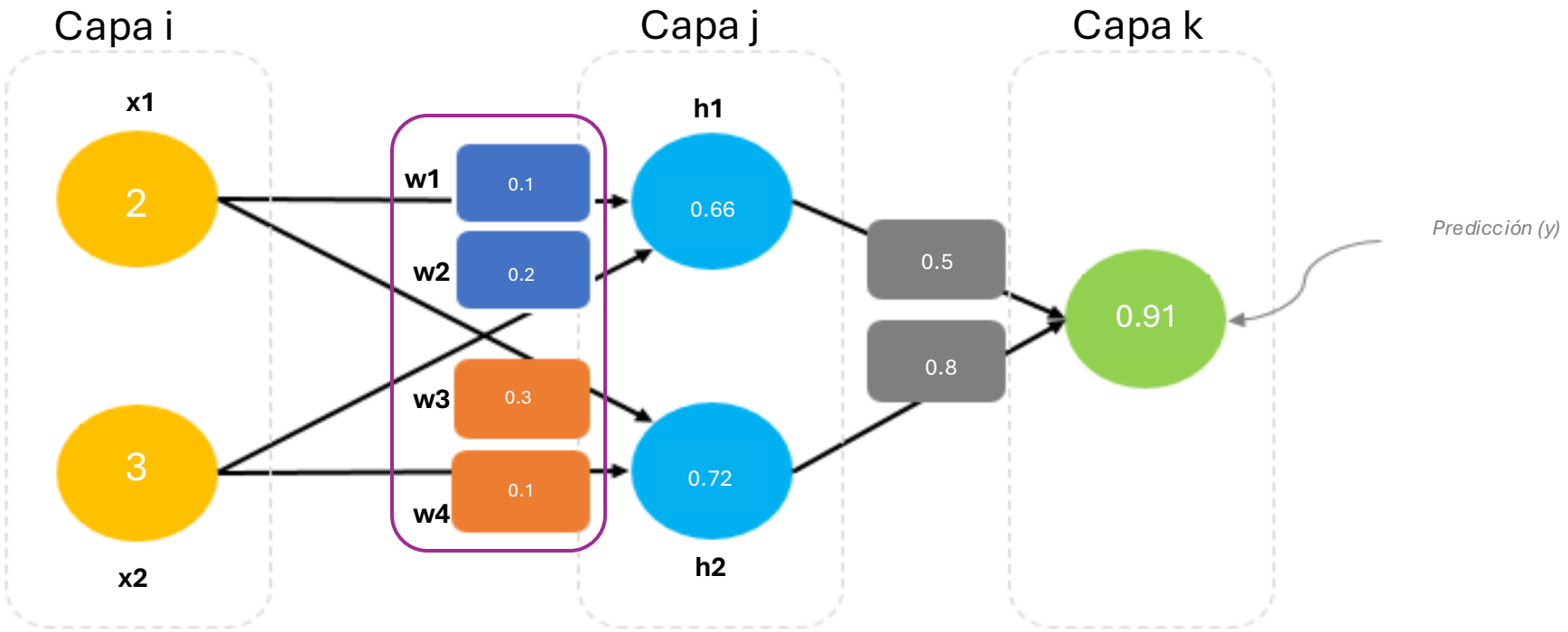
$$\hat{y} = \sum_{j=1}^{c} w_{jk} * h_j$$

$$\textbf{Error} = \frac{1}{2}(\hat{\boldsymbol{y}} - \boldsymbol{y})^2$$

3

$$\frac{\partial Error}{\partial w1} = \frac{\partial Error}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial h_j} \frac{\partial h_j}{\partial w_1} = [(\hat{\boldsymbol{y}} - \boldsymbol{y})][\boldsymbol{w_5}][1 - \tanh(a_1)^2][\boldsymbol{x_1}] = ?$$

Propagación hacia atrás



Cálculos en cada paso

$$h_j = \tanh(a_j) = \tanh\left(\sum_{j=1}^{p} w_{ij} * x_i\right)$$

$$\hat{y} = \sum_{j=1}^{c} w_{jk} * h_j$$

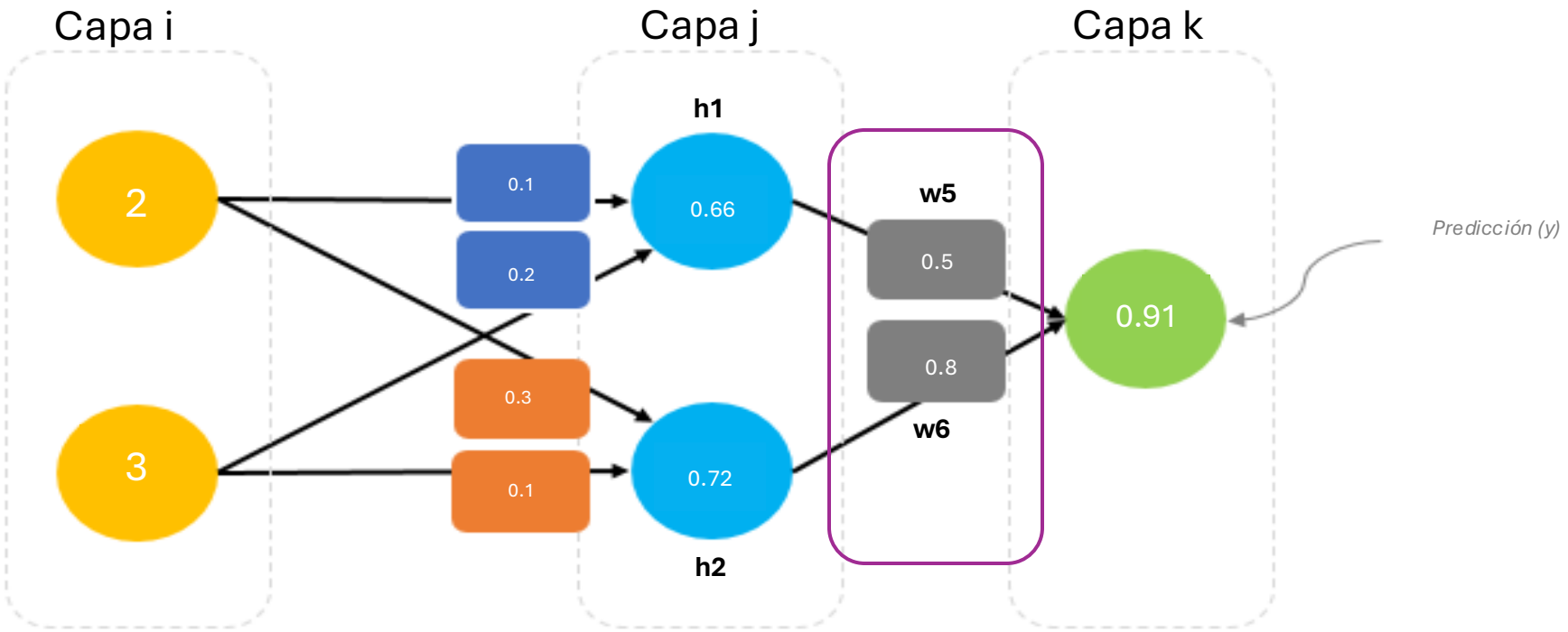
$$\boldsymbol{Error} = \frac{\mathbf{1}}{\mathbf{2}}(\hat{\boldsymbol{y}} - \boldsymbol{y})^{\mathbf{2}}$$

3

$$\frac{\partial Error}{\partial w1} = \frac{\partial Error}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial h_j} \frac{\partial h_j}{\partial w_1} = [(\hat{\boldsymbol{y}} - \boldsymbol{y})][\boldsymbol{w_5}][1 - \tanh(a_1)^2][\boldsymbol{x_1}]$$

$$= -0.09 * 0.5 * (1 - 0.66^2) * 2 = \mathbf{-0.0508}$$

Propagación hacia atrás



Cálculos en cada paso

$$h_j = \tanh\left(\sum_{j=1}^{p} w_{ij} * x_i\right)$$

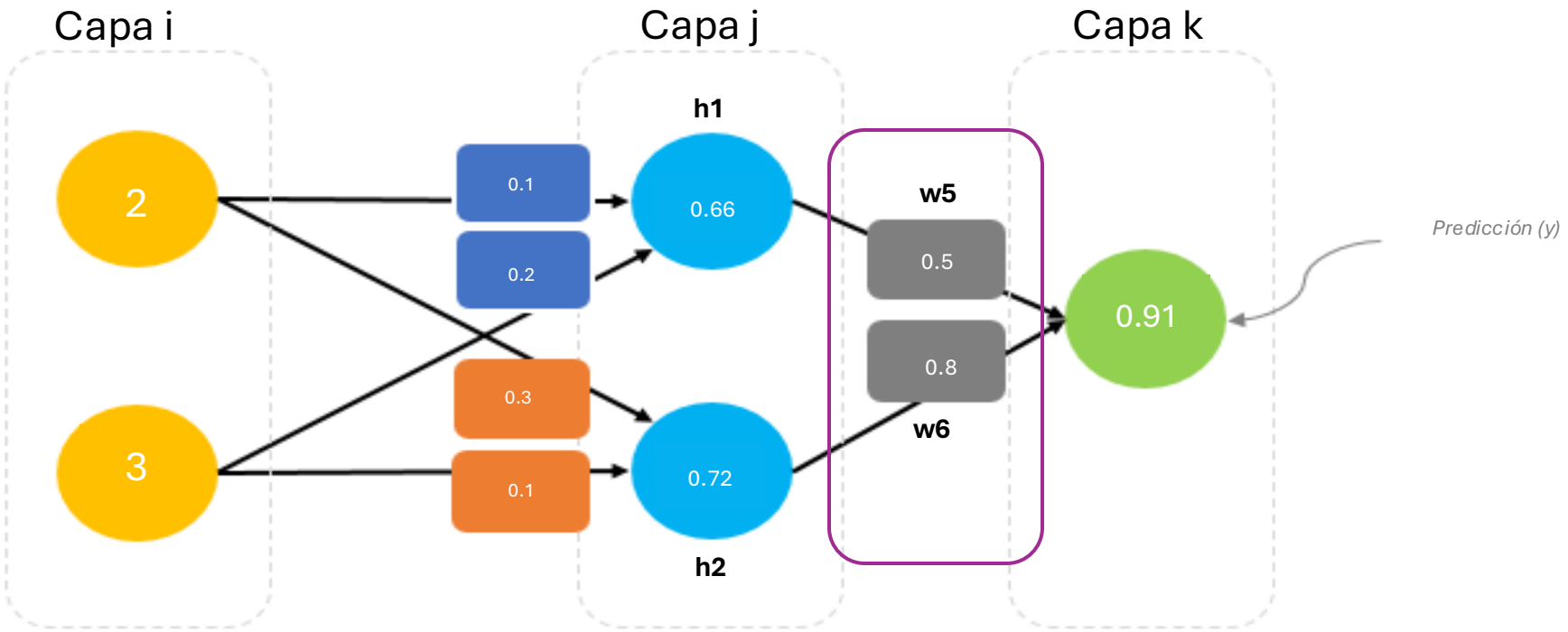
$$\hat{y} = \sum_{j=1}^{c} w_{jk} * h_j$$

$$\textbf{Error} = \frac{1}{2}(\hat{y} - y)^2$$

4

$$\frac{\partial Error}{\partial w4} = \frac{\partial Error}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial h_j} \frac{\partial h_j}{\partial w_4} = [(\hat{y} - y)][w_6][1 - \tanh(a_2)^2][x_2] = ?$$

Propagación hacia atrás



Cálculos en cada paso

$$h_j = \tanh\left(\sum_{j=1}^{p} w_{ij} * x_i\right)$$

$$\hat{y} = \sum_{j=1}^{c} w_{jk} * h_j$$

$$\boldsymbol{Error} = \frac{\mathbf{1}}{\mathbf{2}}(\hat{\boldsymbol{y}} - \boldsymbol{y})^{\mathbf{2}}$$

4

$$\frac{\partial Error}{\partial w4} = \frac{\partial Error}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial h_j} \frac{\partial h_j}{\partial w_4} = [(\hat{\boldsymbol{y}} - \boldsymbol{y})][\boldsymbol{w_6}][1 - \tanh(a_2)^2][\boldsymbol{x_2}]$$

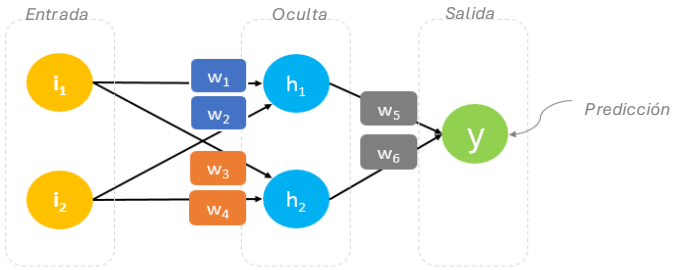
$$= -0.09 * 0.8 * (1 - 0.72^2) * 3 = \mathbf{-0.1040}$$

Actualización de los parámetros

- Ejemplo anterior, con una tasa de aprendizaje de 0.5

$$w(\tau + 1) = w(\tau) - \eta \nabla_w E$$

w	w(t)	$\nabla_w E$	w(t+1)
w_1	0.1	-0.0508	0.1 + 0.5*0.0508 = 0.1254
w_2	0.2	-0.0762	0.2 + 0.5*0.0762 = 0.2381
w_3	0.3	-0.0693	0.3 + 0.5*0.0693 = 0.3346
w_4	0.1	-0.1040	0.1 + 0.5*0.1040 = 0.1520
w_5	0.5	-0.0594	0.5 + 0.5*0.0594 = 0.5297
w_6	0.8	-0.0648	0.8 + 0.5*0.0648 = 0.8324



Esto se repite por muchos epochs/periodos hasta conseguir un resultado satisfactorio en términos de error

Cuáles funciones de activación vimos..

Nombre	Función	Derivada
ReLU	$\max(0, x)$	$\mathrm{ReLU}'(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x < 0 \\ \text{undefined} & \text{if } x = 0 \end{cases}$
Sigmoid/logística	$\sigma(x) = \frac{1}{1 + e^{x}}$	$\sigma'(x) = \sigma(x) \cdot (1 - \sigma(x))$
Tanh (hiperbólica tangente)	$\frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$	$\tanh'(x) = 1 - \tanh^2(x)$
Softmax	$\frac{e^{z_i}}{\sum_j^k e^{z_j}}$	$\mathbf{J}_{ij} = \begin{cases} \sigma(z_i) \cdot (1 - \sigma(z_i)) & \text{if } i = j \\ -\sigma(z_i) \cdot \sigma(z_j) & \text{if } i \neq j \end{cases}$

Funciones de pérdida que hemos visto...

Entropía cruzada binary

$$\text{Loss}(y, \hat{y}) = -\left[y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})\right]$$

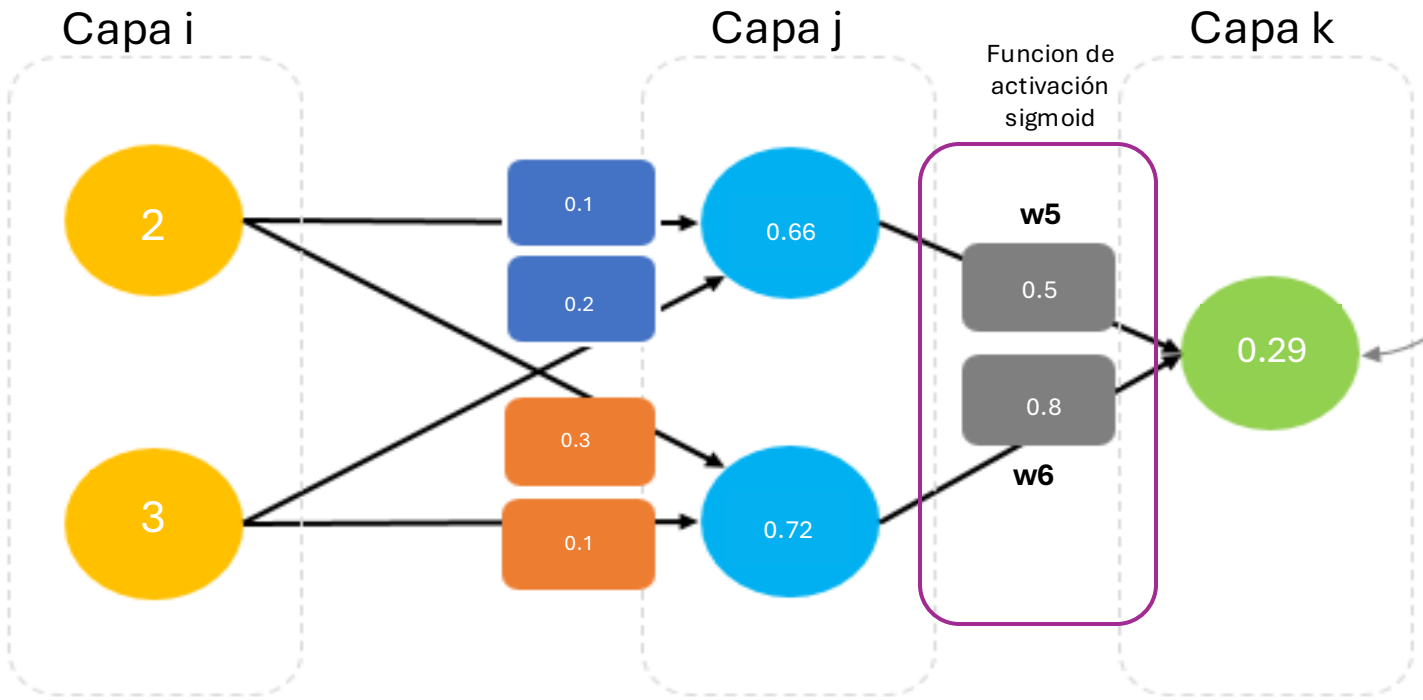
Entropía cruzada categórica

$$\text{Loss}(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{i=1}^{C} y_i \log(\hat{y}_i)$$

Error cuadrático medio

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

Propagación hacia atrás – sigmoid/logistica



$$\frac{1}{1+e^{x}}$$

Predicción (y)

= 0.29

Error: -log(0.29) = xxx

Cálculos en cada paso

$$h_j = \tanh\left(\sum_{j=1}^{p} w_{ij} * x_i\right)$$

$$\hat{y} = \sigma\left(\sum^{c} w_{jk} * h_j\right)$$

$$\text{Loss}(y,\hat{y}) = -\left[y\log(\hat{y}) + (1-y)\log(1-\hat{y})\right]$$

$$\frac{\partial Error}{\partial w5} = \frac{\partial Error}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial w5} = dy * sigmoid(a1) * \left(1 - sigmoid(a1)\right) * h1$$

La derivada del error con respecto al valor predicho depende de cuál es el valor real (y):

$$\frac{\partial Error}{\partial \hat{y}} = -\frac{1}{\hat{y}}\ si\ y = 1\ y\ \frac{\partial Error}{\partial \hat{y}} = \frac{1}{1-\hat{y}}\ si\ y = 0$$