Redes Neuronales

Visión por Computador II

Flujo de entrenamiento

A. Definir hiper-parámetros iniciales (LR, EPOCHS, L2…)

B. Preparación de dataset: lectura de datos, normalización, aumento de datos

C. Entrenamiento

    Loop por cada Epoch:

        Loop por cada batch del dataset:

            1. Predecir con parámetros actuales ($h_\theta(x)$)
            2. Calcular el costo de modelo ($\hat{y} \stackrel{?}{=} y$)
            3. Calcular la gradiente del costo por parámetro ($\nabla_\theta J$)
            4. Actualizar parámetros ($\theta := \theta - \alpha \nabla_\theta J$)

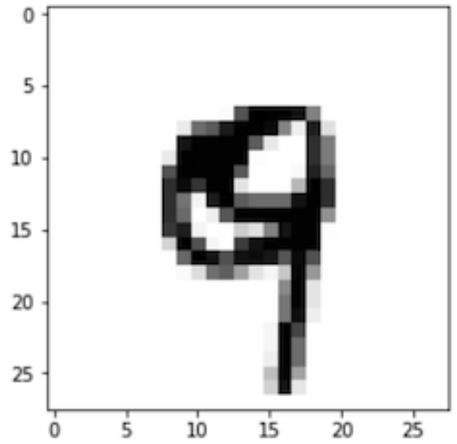
Redes Neuronales

Motivación

Supongamos que tenemos que escribir un programa que clasifique dígitos escritos a mano

MNIST dataset

Imagen 28x28px

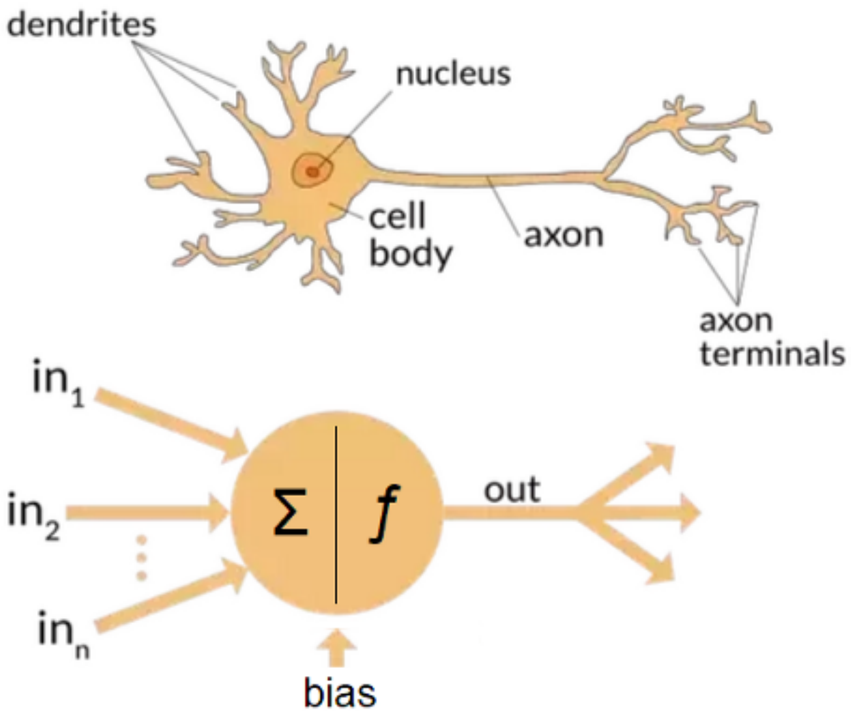


¿Cómo podríamos clasificar los 10 dígitos?

Perceptrón

Inspirada en las redes neuronales biológicas (50s)

- Cerebro compuesto por 10 billones de neuronas
- Cada una conectada a 10.000 Neuronas
- Señal activa ciertas neuronas
- Activada si la señal supera un límite



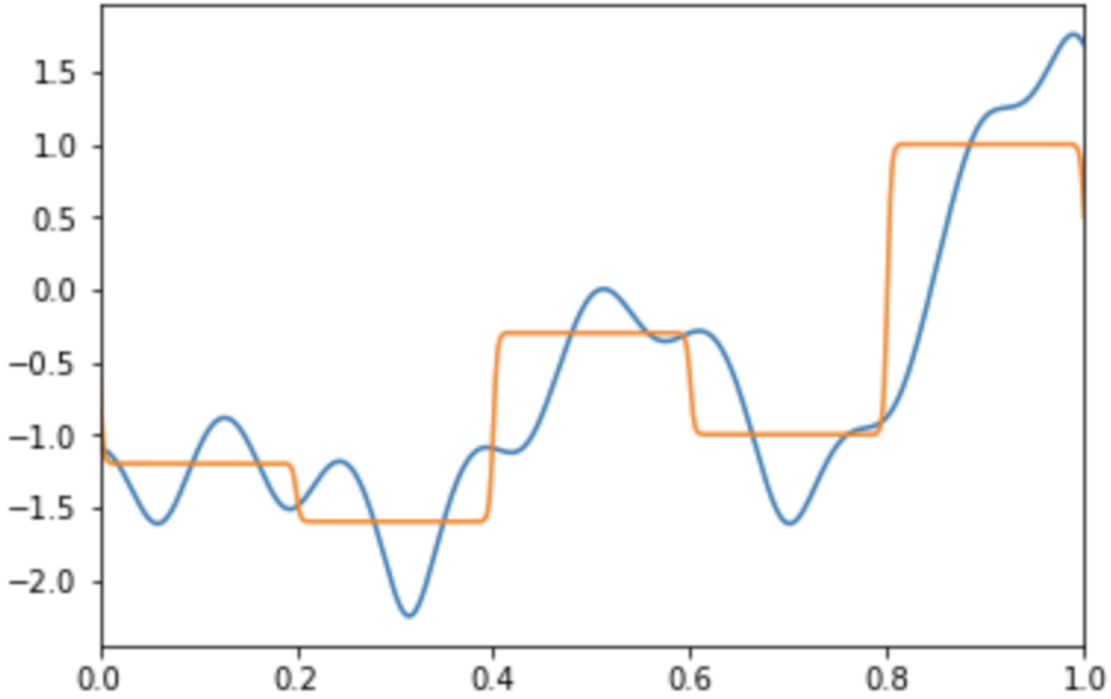
towardsdatascience.com

Redes Neuronales

- Las *redes profundas feedforward*, también llamadas *Redes Neuronales*, o *Perceptrones multicapa (MLP)*, son los modelos de ML por defecto

- El objetivo de una red es aproximar alguna función f^*.

- Define un mapeo $y = f(x; \theta)$ y aprende el valor de los parámetros θ que dan como resultado la mejor aproximación de la función



A visual proof that neural nets can compute any function

http://neuralnetworksanddeeplearning.com/chap4.html

Redes Neuronales

- Las redes feedforward son de extrema importancia para ML. Constituyen la base de muchas aplicaciones comerciales importantes.

- Por ejemplo, las redes CNN que se utilizan para el reconocimiento de objetos a partir de fotografías son un tipo especializado de red feedforward.

- Las redes neuronales feedforward se llaman redes porque suelen representarse componiendo muchas funciones distintas.

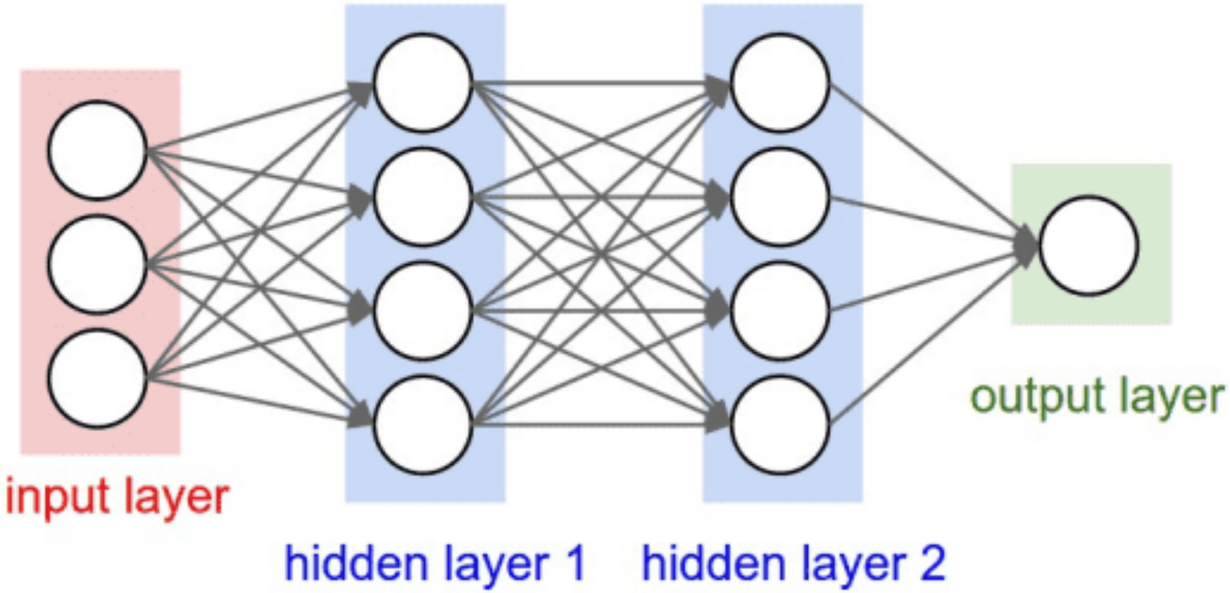
https://serious-science.org/artificial-neural-networks-10459

Redes Neuronales

Estos modelos se denominan *feedforward* porque la información fluye a través de la función que se evalúa de x,

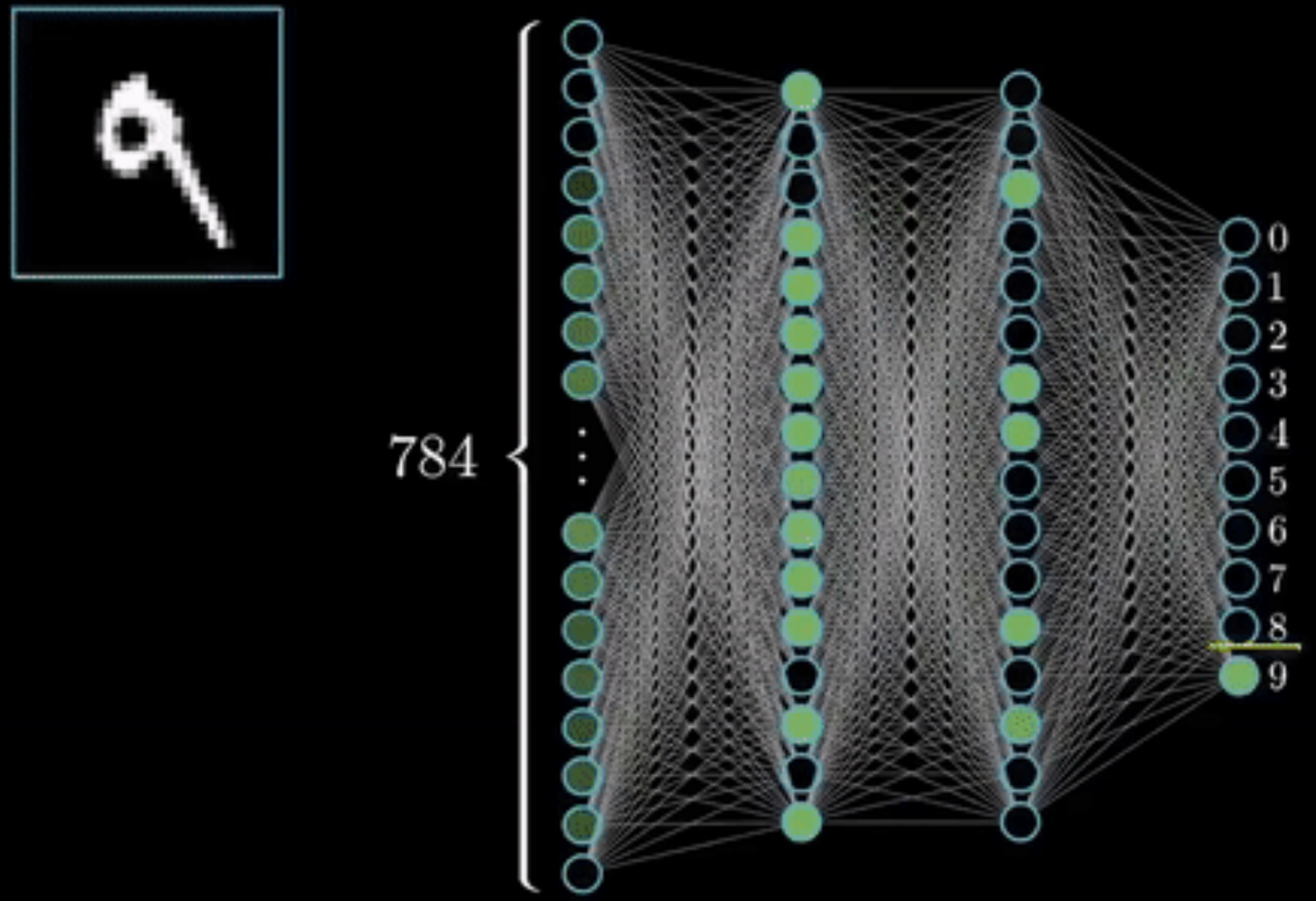
a través de los cálculos intermedios utilizados para define f,

y finalmente a la salida y.



https://mukulrathi.com/demystifying-deep-learning/feed-forward-neural-network/

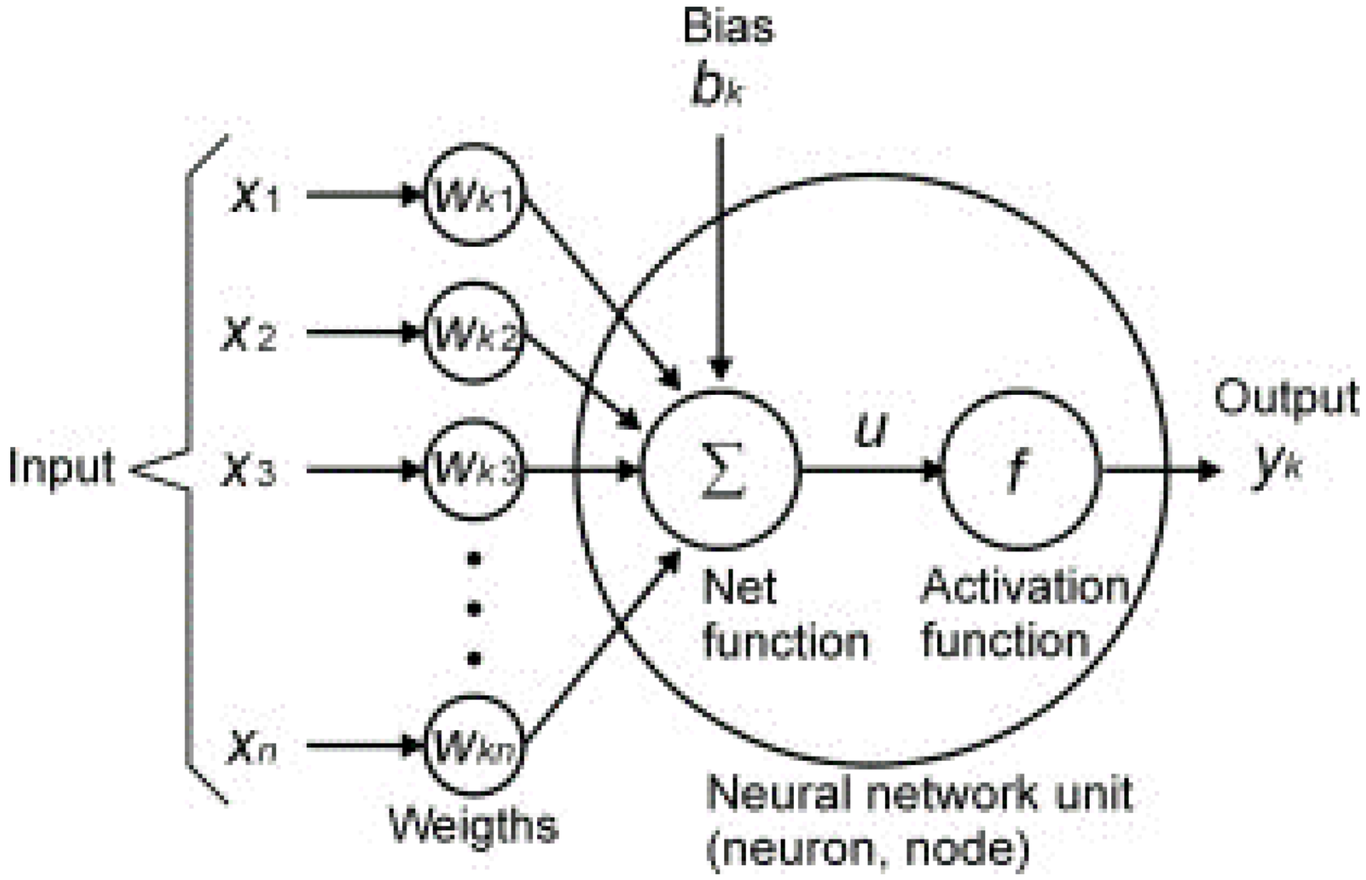
Feedforward



Patrón de activación de una capa causa una patrón específico en la siguiente

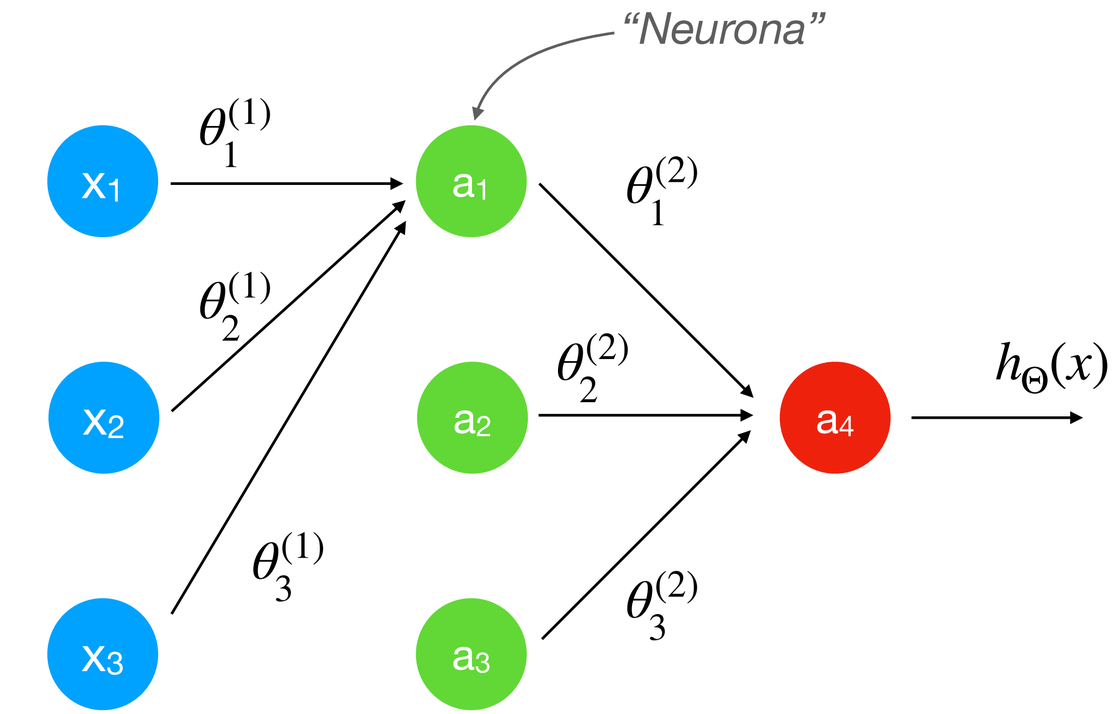
youtube.com/watch?v=aircAruvnKk

Unidad de Red Neuronal



2020 - Comparative Analysis of Recurrent Neural Network Architectures for Reservoir Inflow Forecasting

Red Neuronal



Layer 1

Input layer

Layer 2

Hidden layer

Layer 3

Output layer

$a_i^{(j)}$ *Activación de la unidad* i *de la capa* j

$\Theta^{(j)}$*Matriz de los pesos de la capa j*

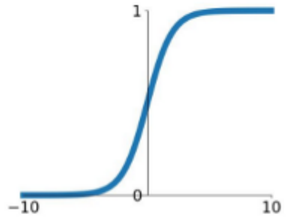
$$a_1 = \phi(\sum_i^n x\theta^T)$$

Funciones de activación

Activation Functions

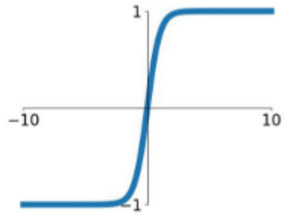
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



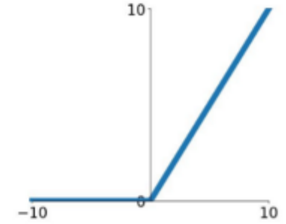
tanh

$$\tanh(x)$$



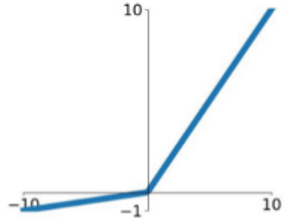
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

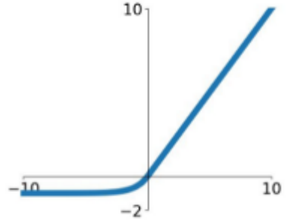


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



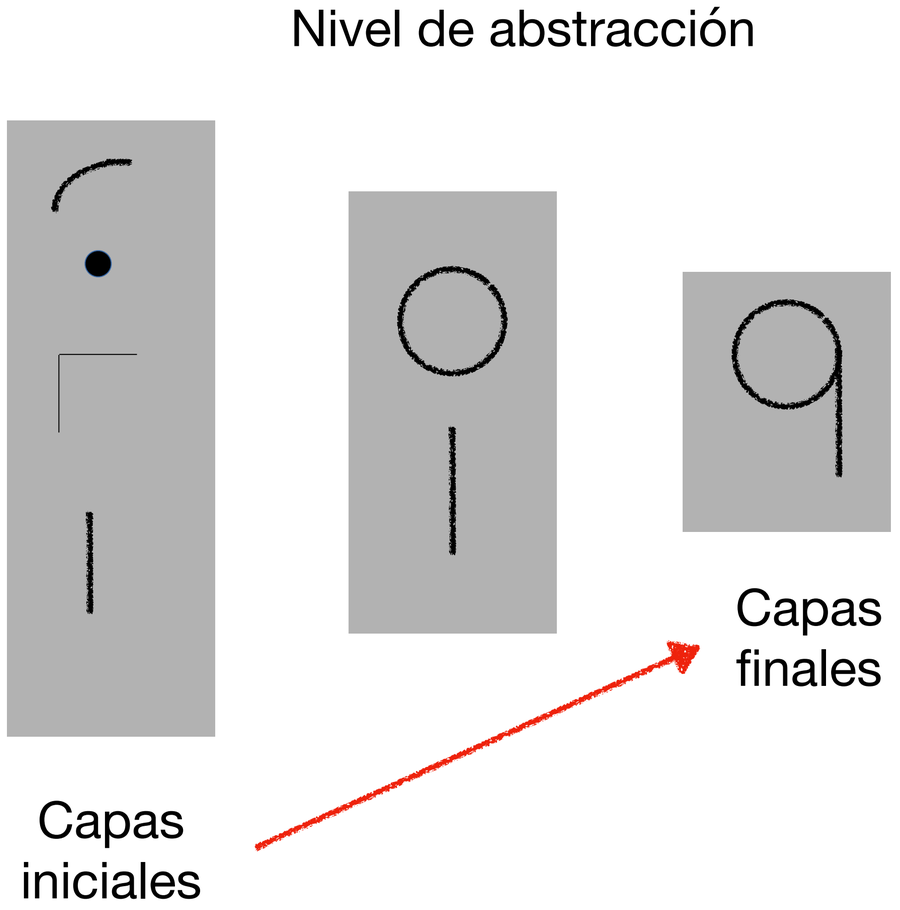
https://medium.com/@shrutijadon/survey-on-activation-functions-for-deep-learning-9689331ba092

Estructura en Capas

Comportamiento hipotético

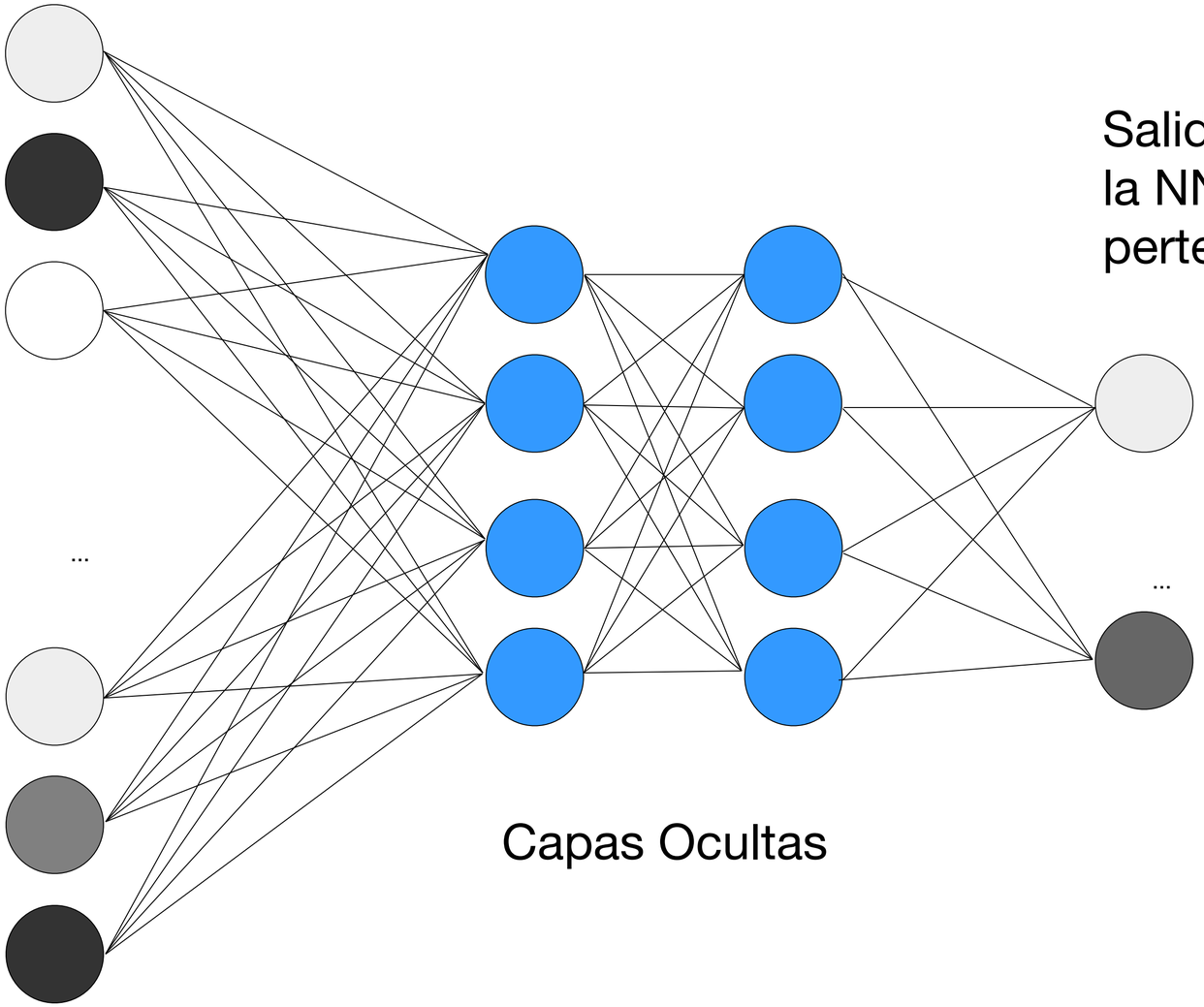
Cada unidad se activa cuándo encuentra elementos específicos en la capa anterior.

1. **Capas iniciales** Elementos básico
2. **Capas intermedias** Sub-componentes
3. **Capas finales** Respuesta



Arquitectura Neurona Artificial

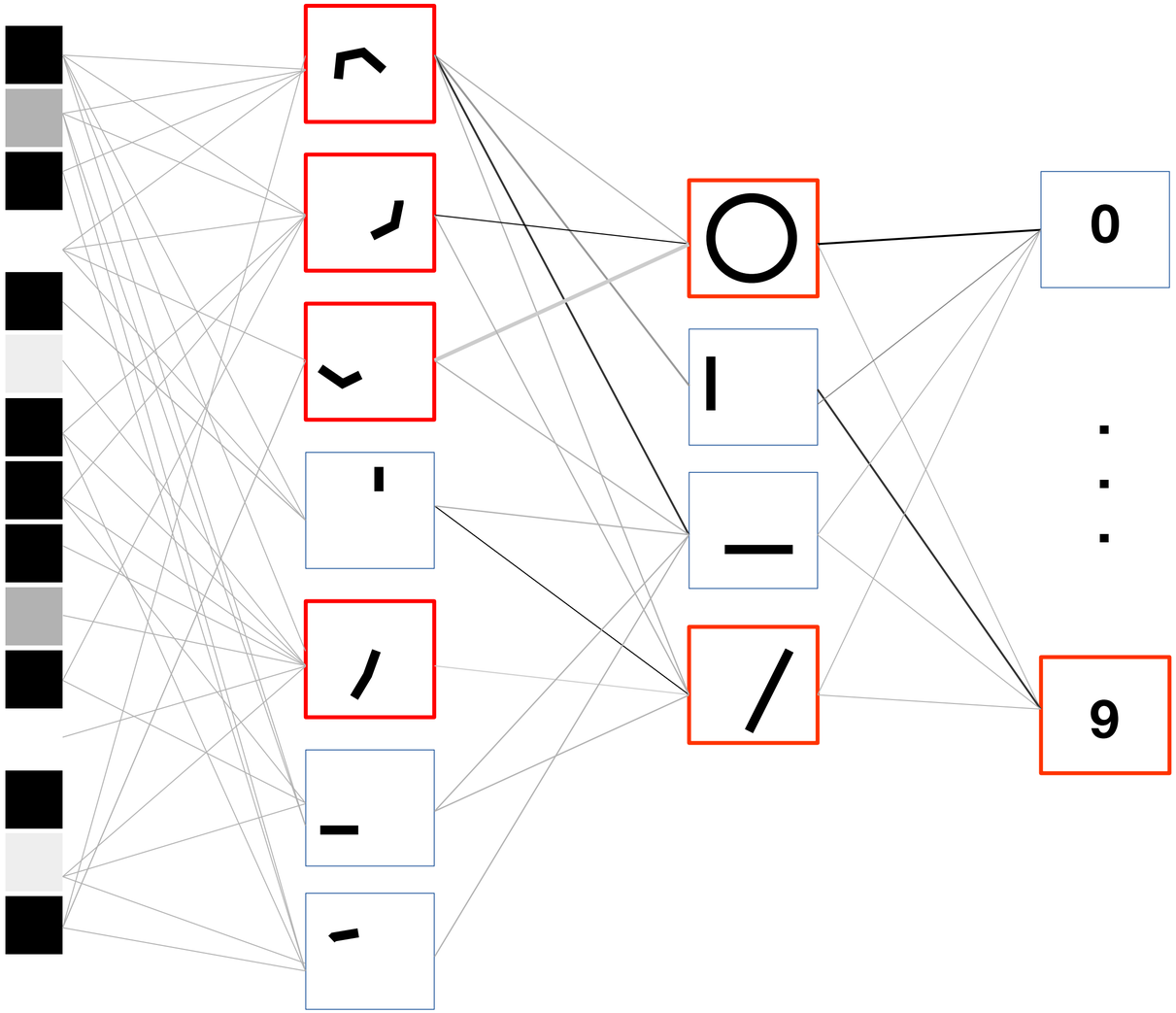
Valores de Imagen



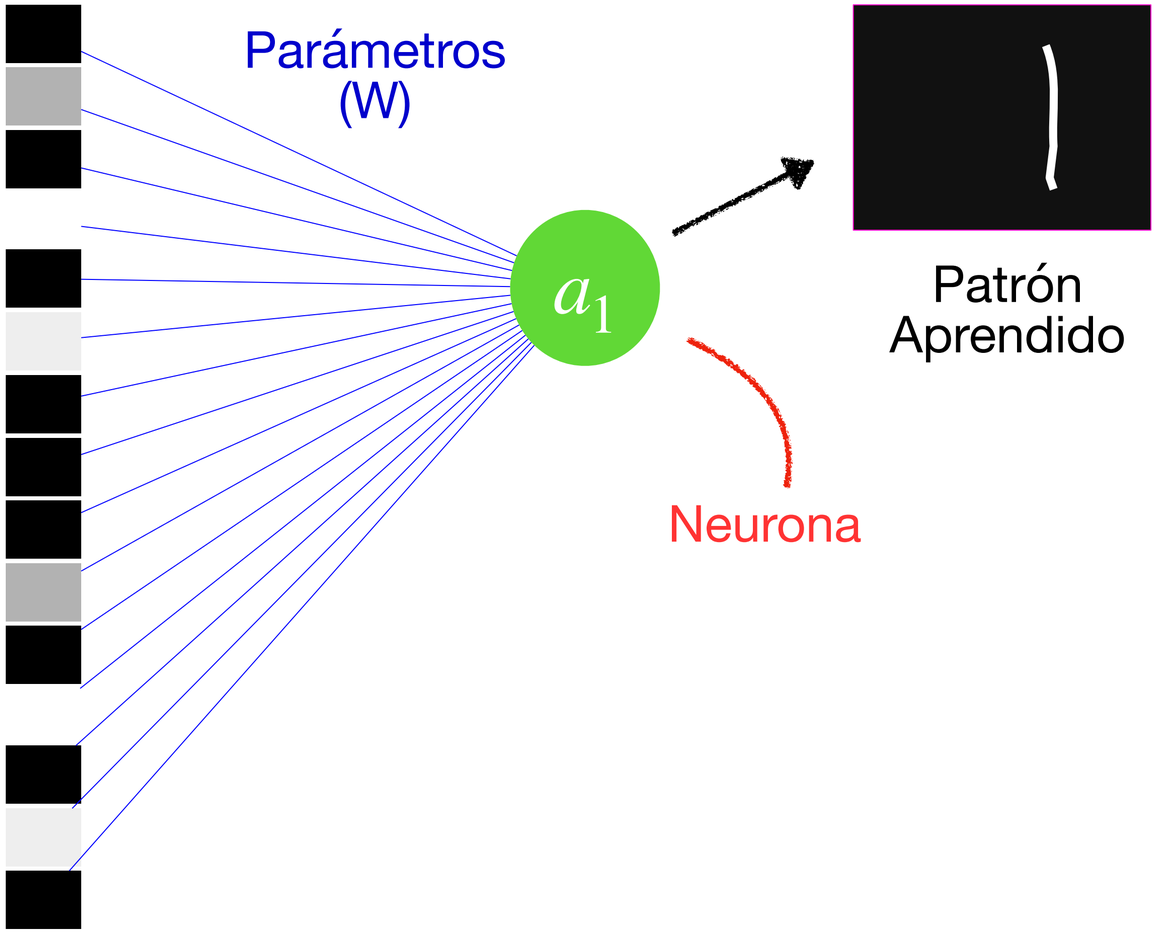
Capas Ocultas

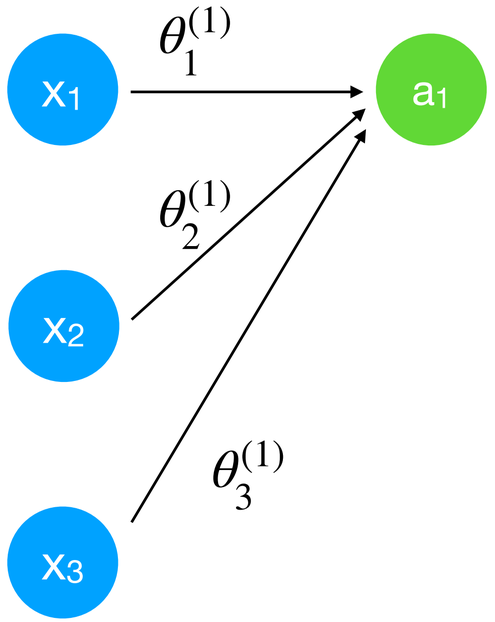
Salida: Que tanto cree la NN que la imagen pertenece a una clase

Comportamiento Hipotético



Comportamiento Hipotético



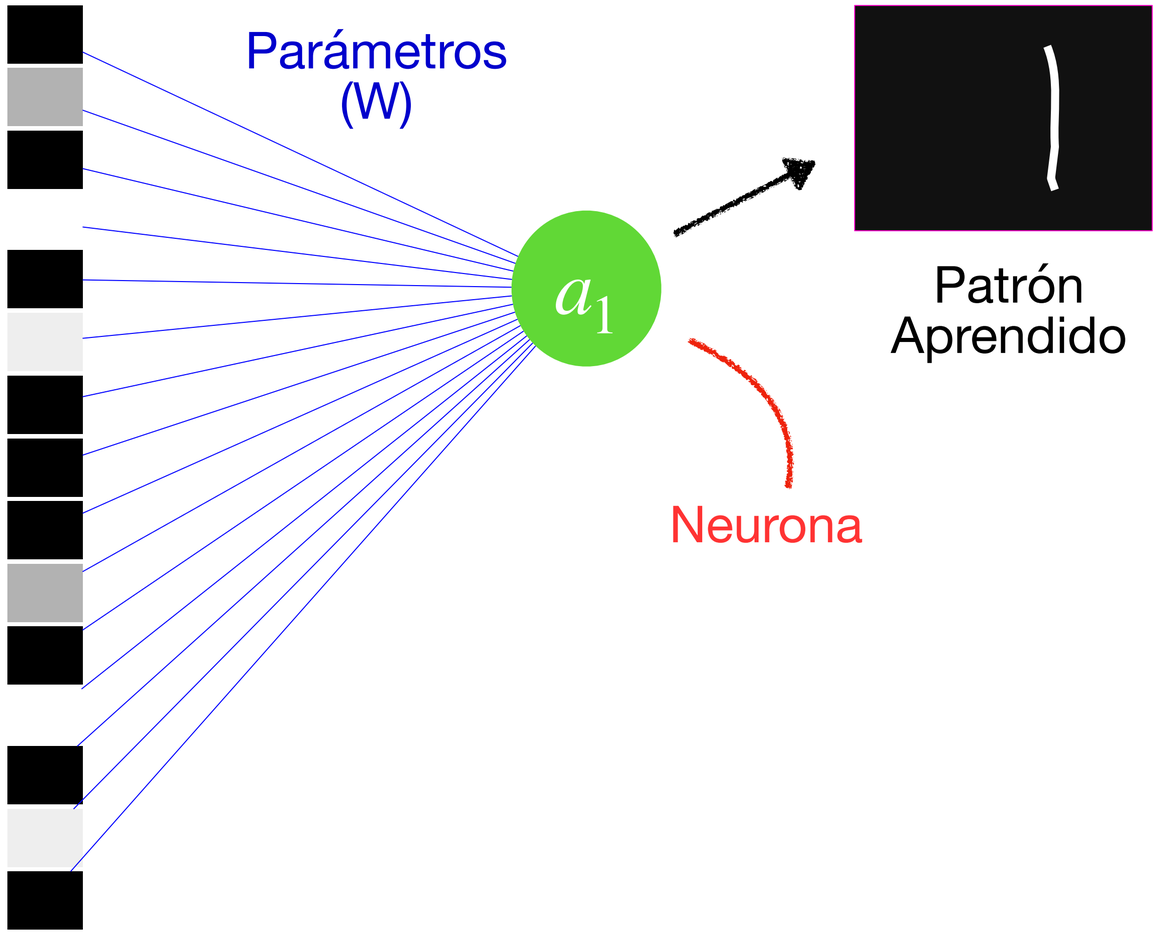


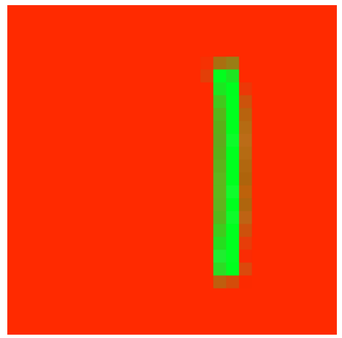
$$\phi(\theta_1 a1 + \theta_2 a_2 + \theta_3 a_3 \ldots + \theta_n a_n + b)$$

Reconocimiento de elementos básicos

Comportamiento Hipotético

Reconocimiento de elementos básicos





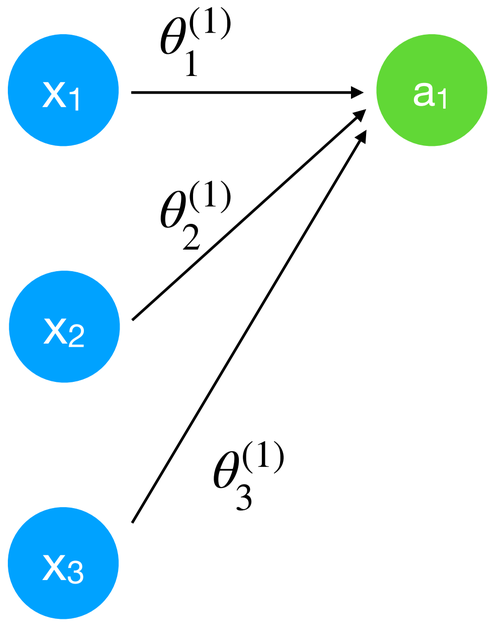
$$\Theta$$

$$\phi(\theta_1 a1 + \theta_2 a_2 + \theta_3 a_3 \ldots + \theta_n a_n + b)$$

Cómputo de Predicciones

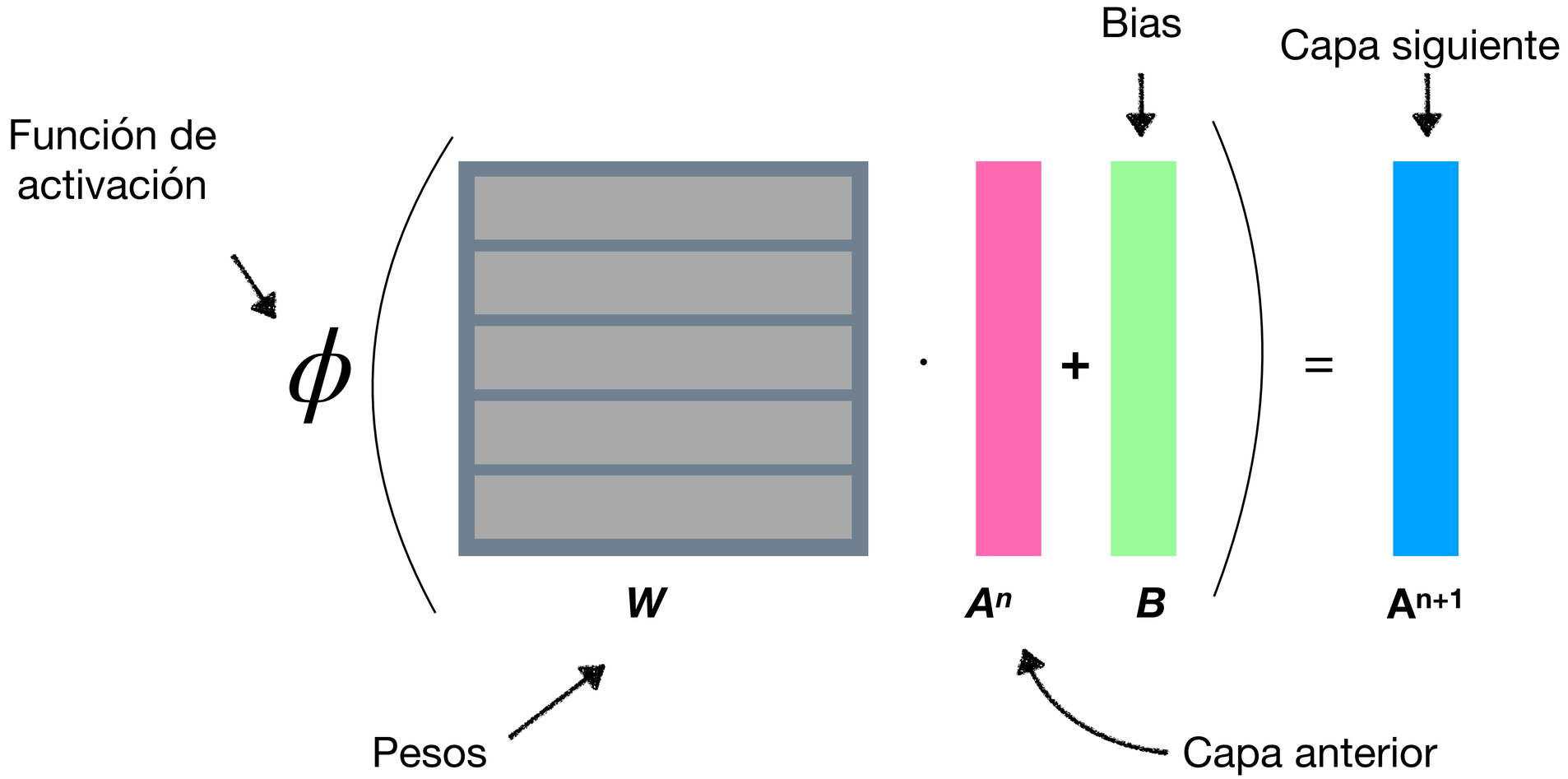
Bias Term

- Controla que tan difícil es activar una neurona
- Activar señales significativas
- También es entrenado



$$\phi(\theta_1 a1 + \theta_2 a_2 + \theta_3 a_3 \ldots + \theta_n a_n + b)$$

Cómputo de Predicciones



$$Z = W \cdot A_n + B$$

$$A_{n+1} = \phi(Z)$$

Entrenamiento

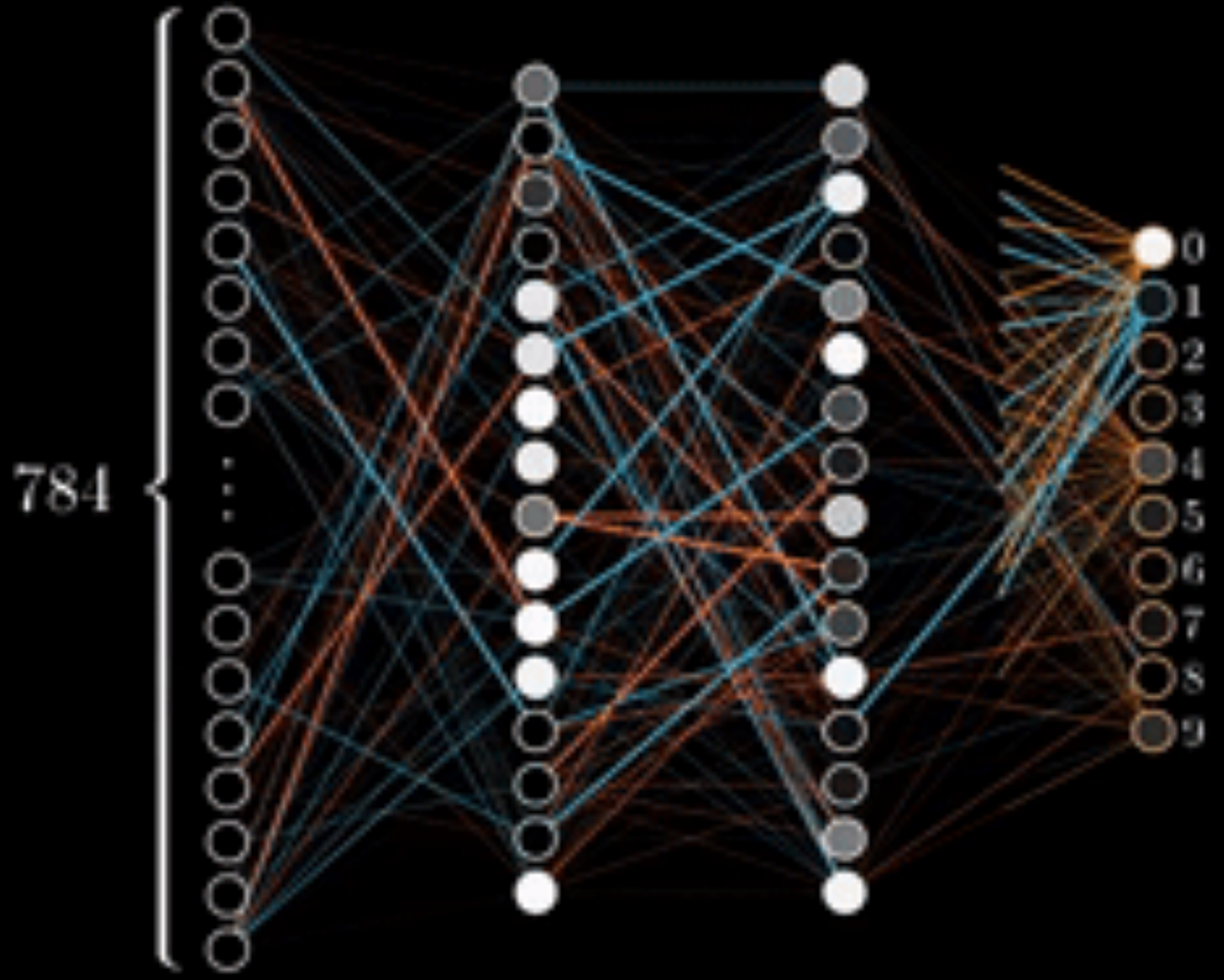
Etapas de entrenamiento

i. **Feedforward**
   Predicción con los pesos actuales

ii. **Back Propagation**
   Ajustar las ponderaciones en función del gradiente de la función de costes

https://www.furrytips.com/bad-dog-behaviors-encouraged-by-owners/

Backpropagation



https://developer.nvidia.com/blog/a-data-scientists-guide-to-gradient-descent-and-backpropagation-algorithms/

Backpropagation

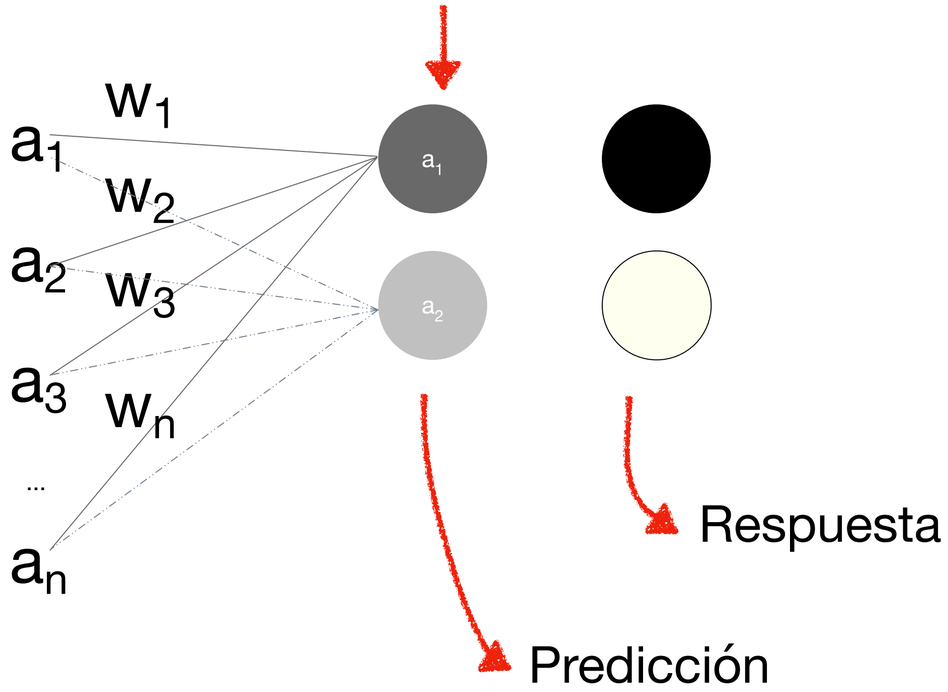
Modificar los parámetros

- Proporcional a que tan lejos están del valor deseado
- Que tanto influencian la respuesta final
- Modificar las activaciones anteriores

Propagar a la capa anterior

- Teniendo en cuenta las otras activaciones
- Propagar coherente

$$\phi(\theta_1 a1 + \theta_2 a_2 + \theta_3 a_3 \ldots + \theta_n a_n + b)$$

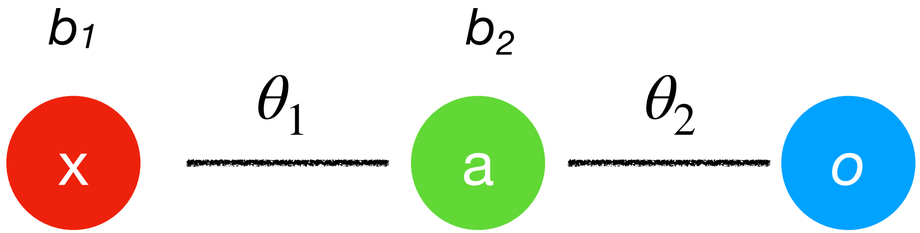


Backpropagation

- Ejemplo con una red simple
- Objetivo es calcular $\nabla \theta_j$
- Regla de la cadena

$$F(x) = f(g(x))$$

$$F'(x) = f'(g(x))g'(x)$$



NN de dos capas:

$$z = \theta_1^T x + b$$

$$a = \phi(z)$$

$$o = \phi(\theta_2^T a + b_2)$$

Backpropagation

Error del modelo (costo):

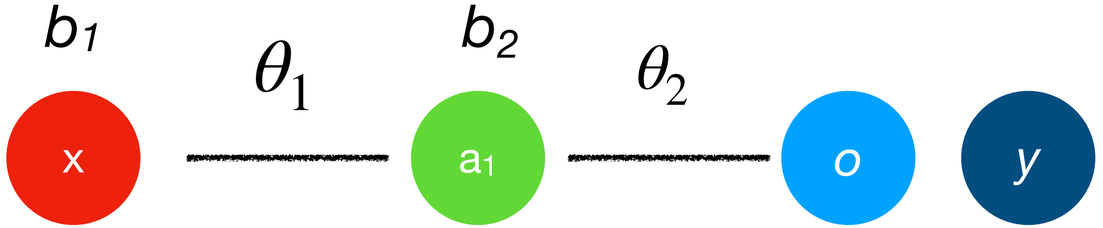
$$J = \frac{(o-y)^2}{2}$$

Necesitamos calcular las derivadas del costo con respecto a cada parámetro

$$\nabla\theta = \begin{bmatrix} \frac{\partial J}{\partial \theta_1} \\ \frac{\partial J}{\partial \theta_2} \end{bmatrix}$$

$$\frac{\partial J}{\partial \theta_2} = \frac{\partial J}{\partial o}\frac{\partial o}{\partial \theta_2} = (o-y)a$$

$$\frac{\partial J}{\partial \theta_1} = \frac{\partial J}{\partial o}\frac{\partial o}{\partial \theta_1} = (o-y)\theta_2\phi'(z)x$$



NN de dos capas:

$$o = \theta_2^T a + b_2$$

$$a = \phi(z)$$

$$z = \theta_1^T x + b$$

Ver detalle de derivadas en Notas de ML Cap 7.3 (Backpropagation)

Gradient descent

Algoritmo (Forma vectorial)

1. Hacer feedforward y calcular (z, a, o)
2. Calcular:
    1. $\delta_2 = o - y$
    2. $\delta_1 = (o - y) \cdot \theta_2^T \odot \phi'(z)$

3. Calcular gradientes

$$\frac{\partial}{\partial \theta_2} = \delta_2 a^T, \quad \frac{\partial}{\partial b_2} = \delta_2$$

$$\frac{\partial}{\partial \theta_1} = \delta_1 x^T, \quad \frac{\partial}{\partial b_1} = \delta_1$$

Derivadas de la función de activación

- Calcular la derivada de la función de activación de acuerdo a la activación usada en cada capa
- Diferentes capas pueden tener diferentes activaciones

Function Type	Equation	Derivative
Linear	$f(x) = ax + c$	$f'(x) = a$
Sigmoid	$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)\,(1 - f(x))$
TanH	$f(x) = tanh(x) = \frac{2}{1+e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ReLU	$f(x) = \begin{cases} 0 \text{ for } x<0 \\ x \text{ for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 \text{ for } x<0 \\ 1 \text{ for } x \geq 0 \end{cases}$
Parametric ReLU	$f(x) = \begin{cases} \alpha x \text{ for } x<0 \\ x \text{ for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha \text{ for } x<0 \\ 1 \text{ for } x \geq 0 \end{cases}$
ELU	$f(x) = \begin{cases} \alpha(e^x - 1) \text{ for } x<0 \\ x \text{ for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha \text{ for } x<0 \\ 1 \text{ for } x \geq 0 \end{cases}$

Derivadas de funciones de activación

https://www.analyticsvidhya.com/blog/2021/04/activation-functions-and-their-derivatives-a-quick-complete-guide/

Ejercicio

Crear una red neuronal de una capas para la clasificación de dígitos del dataset MNIST