#AprendeDesdeCasa
#AprendeConLosPioneros

Online | DMC Perú

CURSOS ANALYTICS

Machine Learning Advanced

- Redes Neuronales -

Docente: Manuel Montoya

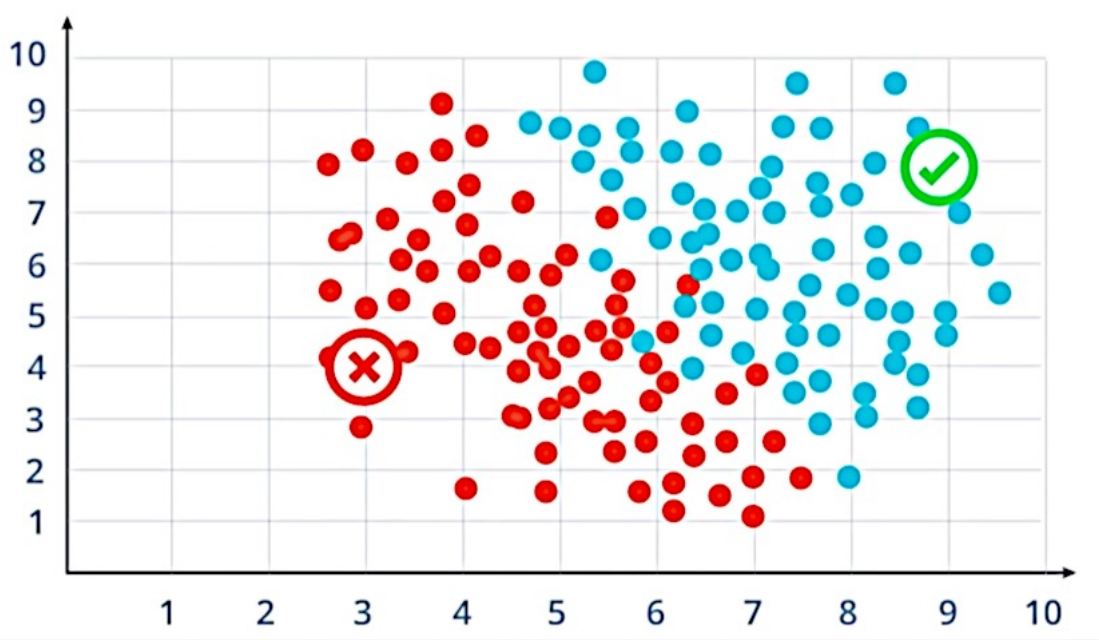
Agenda

1. Clasificación con Perceptrón
2. Redes Neuronales
3. Feedforward y Backpropagation

Perceptrón

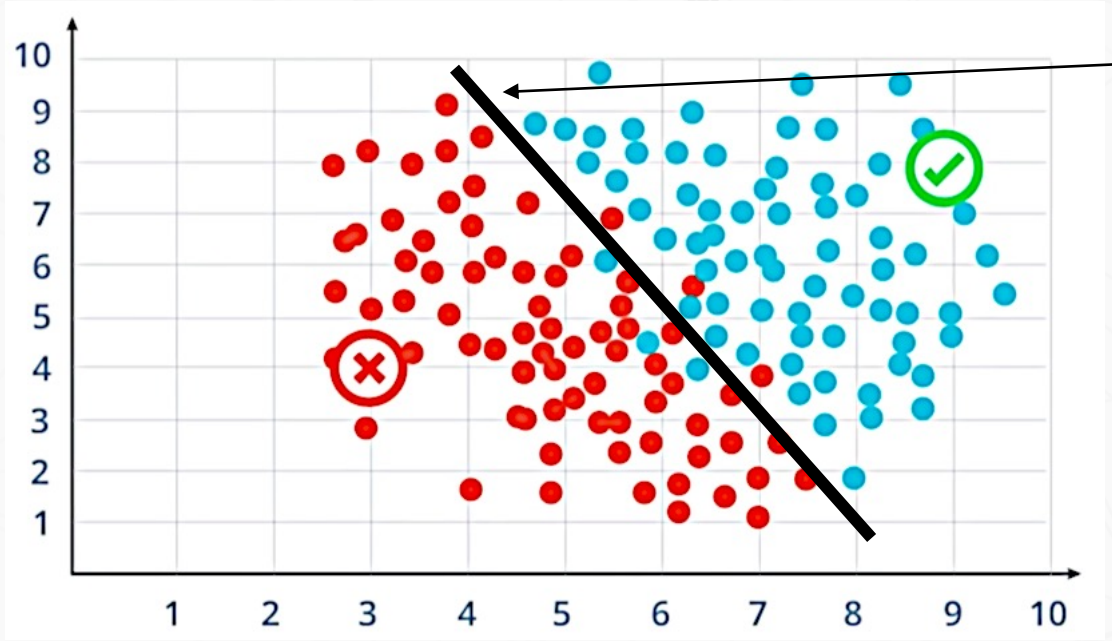
Clasificación: Admisión a universidad

notas



ensayo

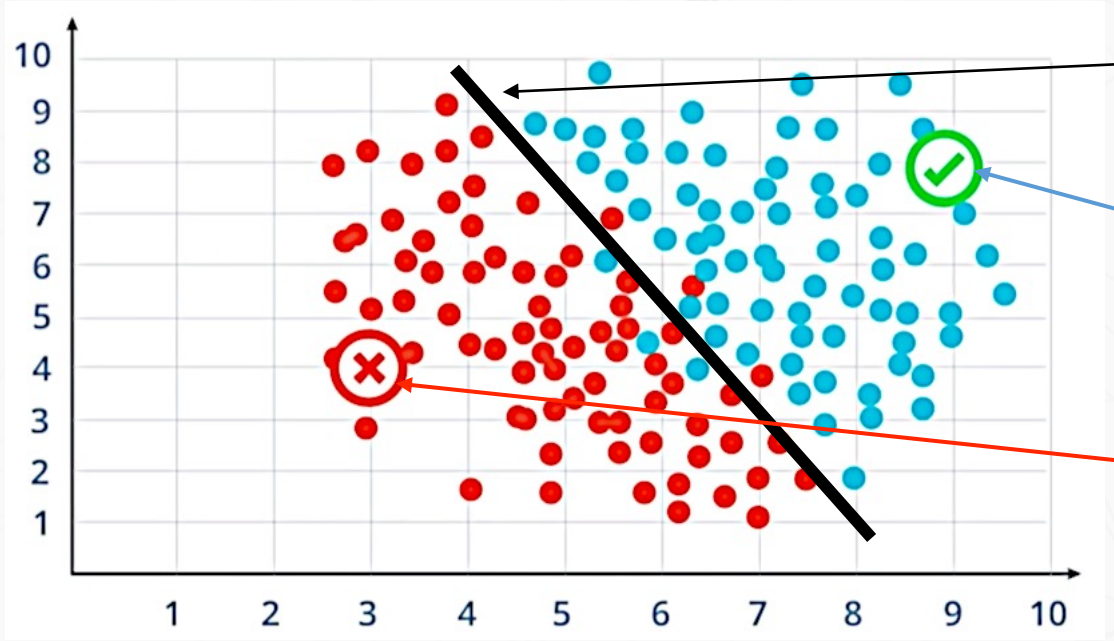
Clasificación: Admisión a universidad



Límite de decisión

2 * notas + 1 * ensayo – 18 = 0

Clasificación: Admisión a universidad

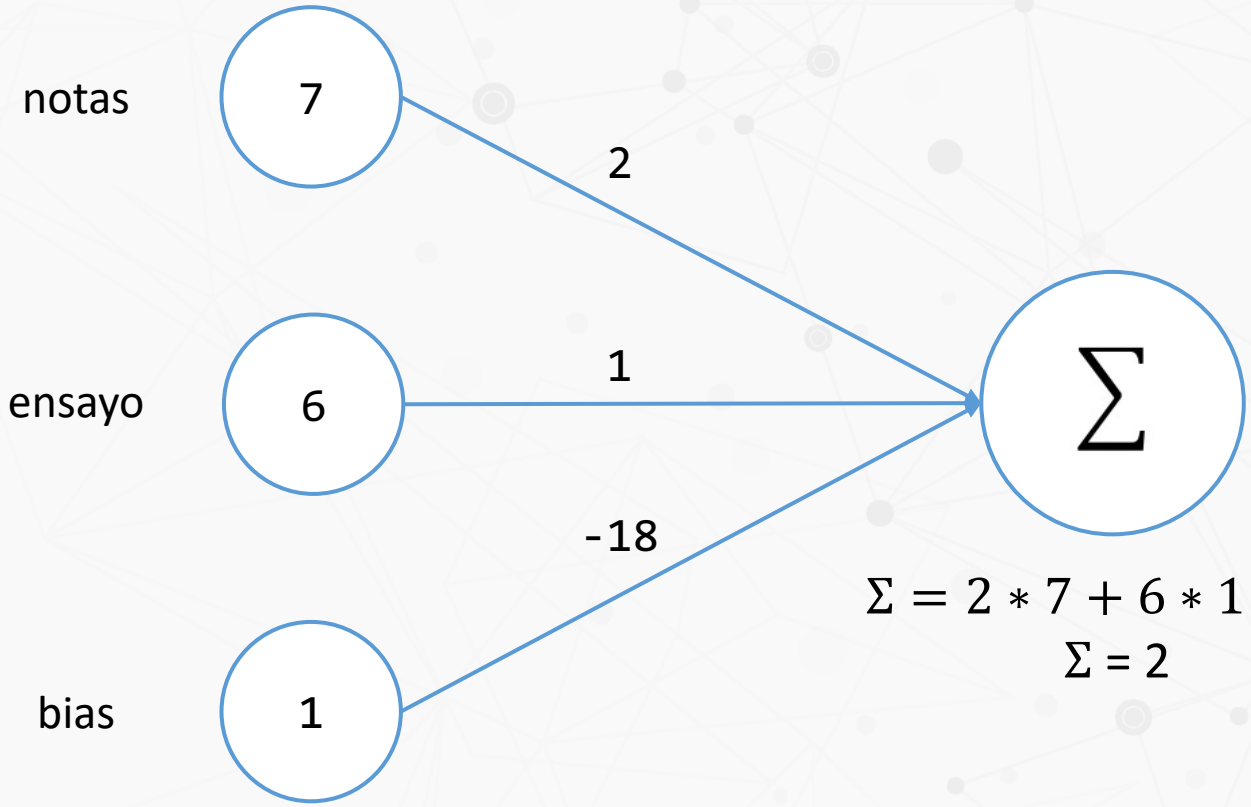


Límite de decisión
2 * notas + 1 * ensayo – 18 = 0

Admitido
2 * notas + 1 * ensayo - 18 >= 0

No admitido
2 * notas + 1 * ensayo - 18 < 0

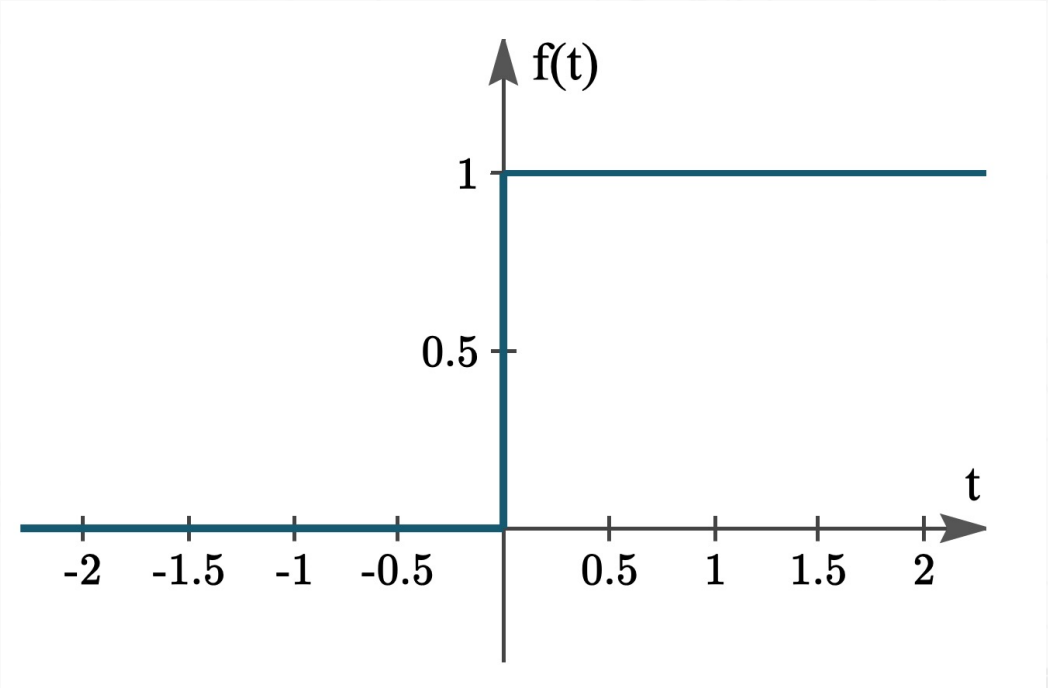
Clasificación: Perceptrón



$$\Sigma = 2 * 7 + 6 * 1 - 18$$

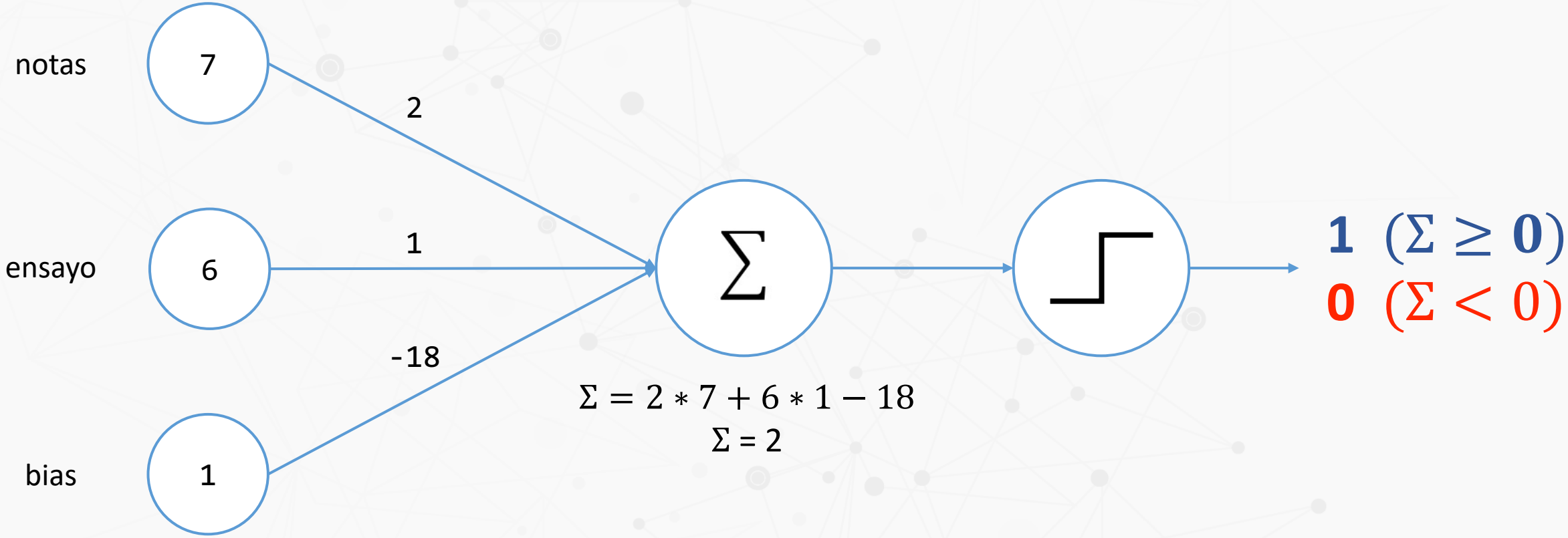
$$\Sigma = 2$$

Step function



$$u(t) = \begin{cases} 0 & t < 0 \\ 1 & t > 0 \end{cases}$$

Clasificación: Perceptrón

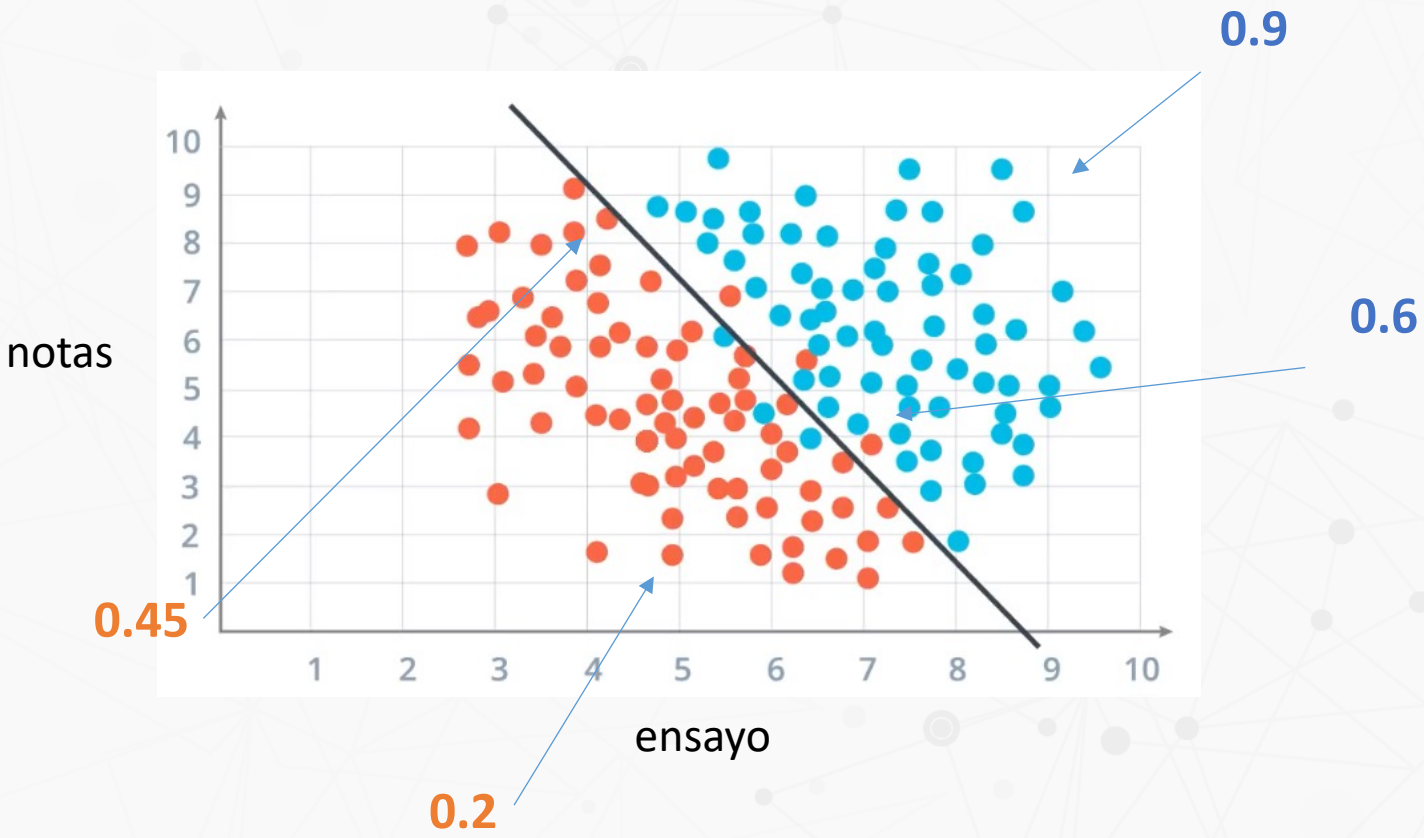


Predicción de probabilidades

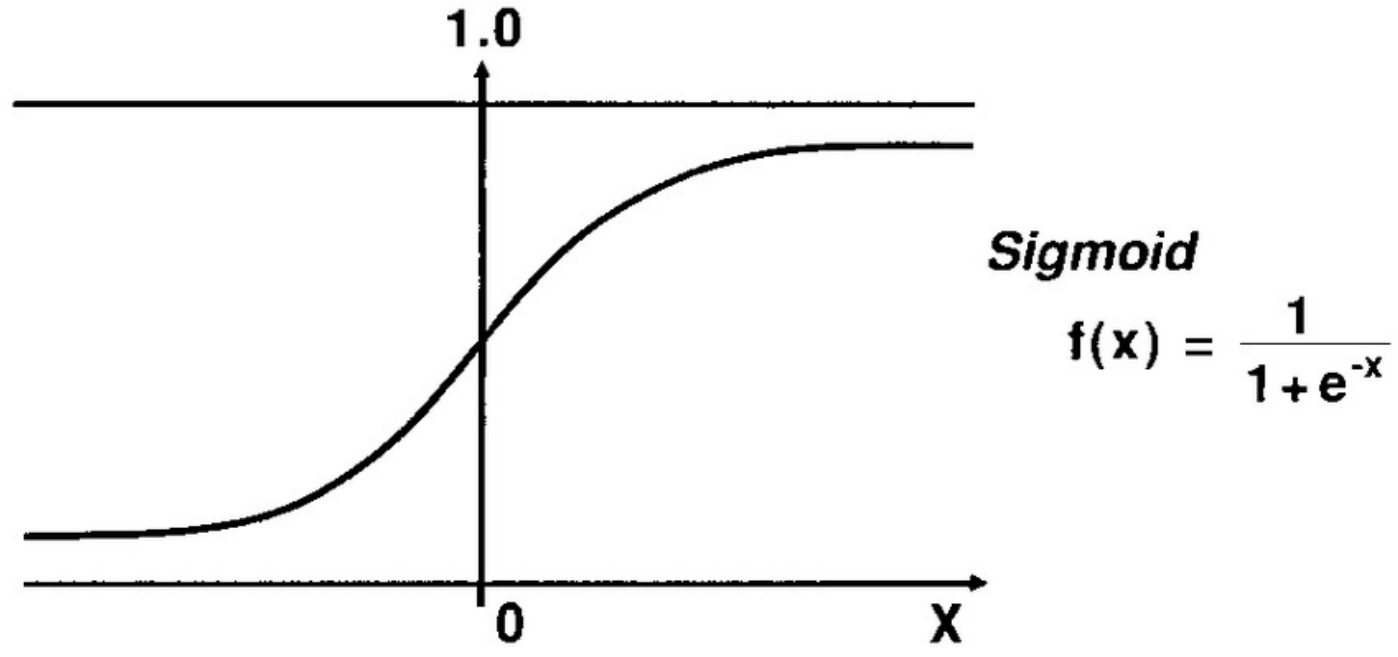
notas

ensayo

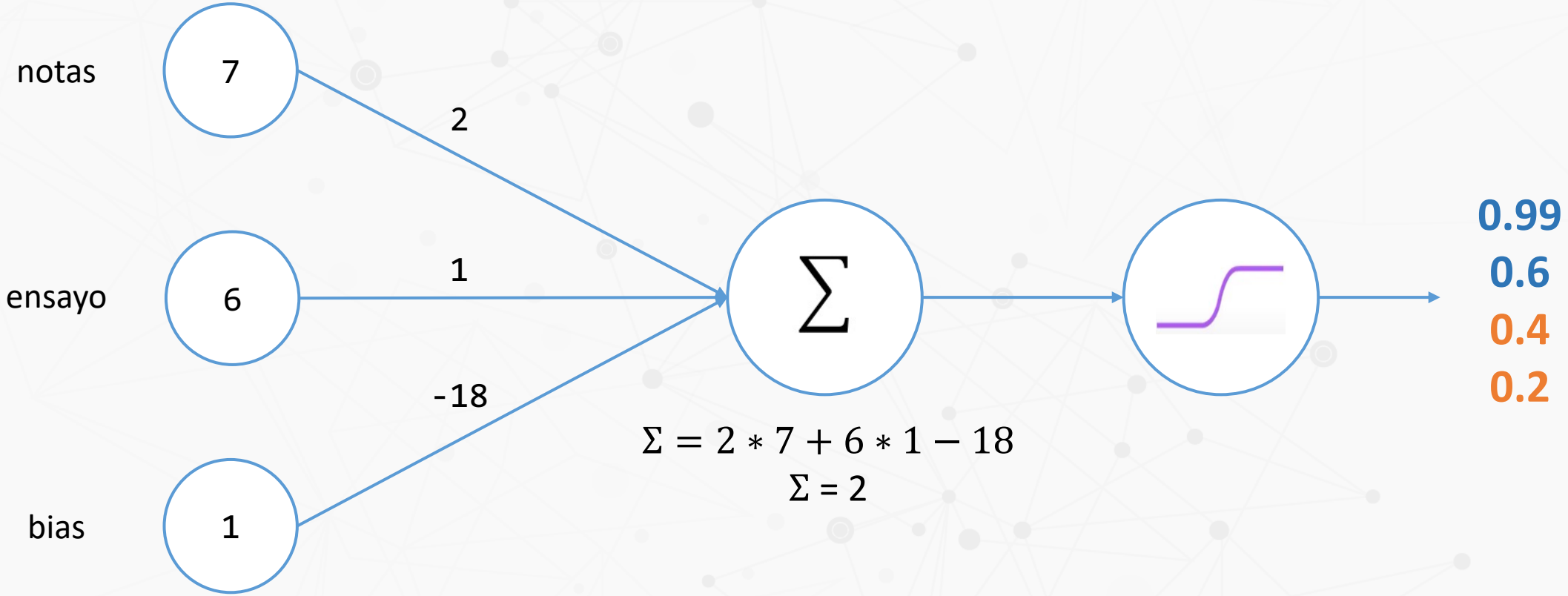
Predicción de probabilidades



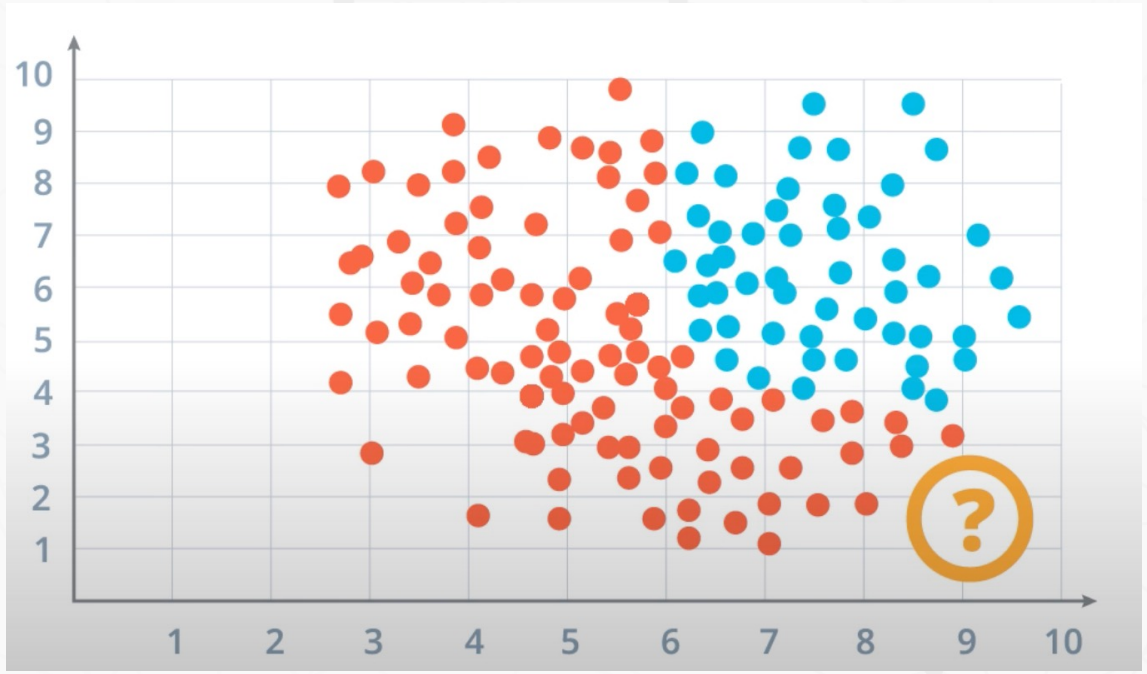
Función Sigmoidal



Predicción de probabilidades

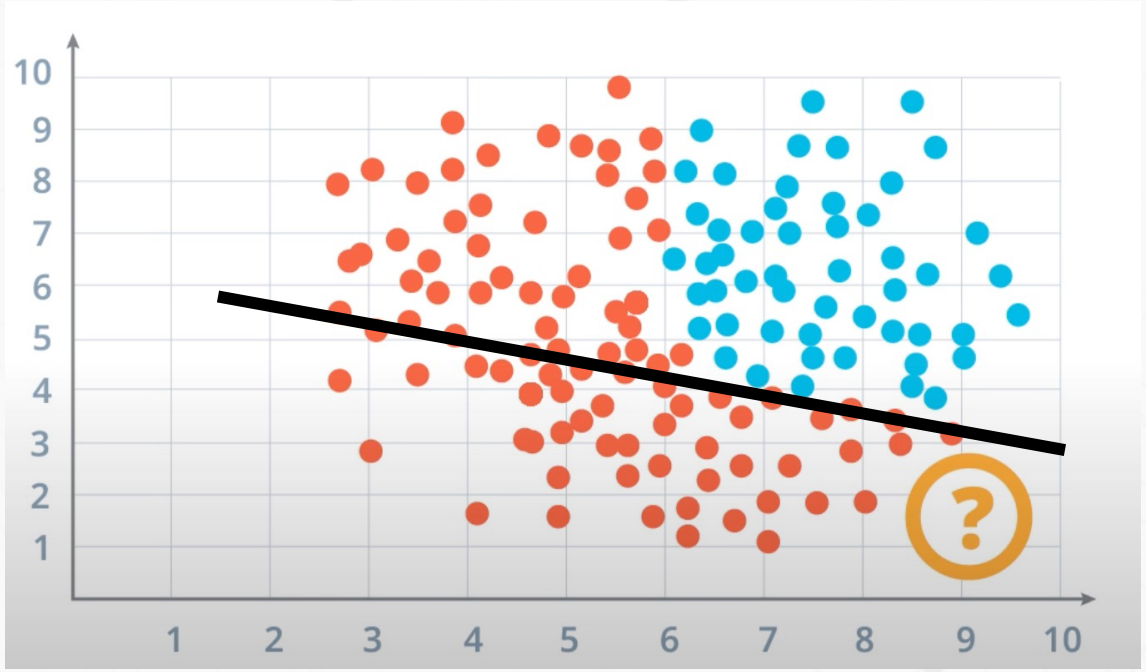


Límites de decisión no lineales



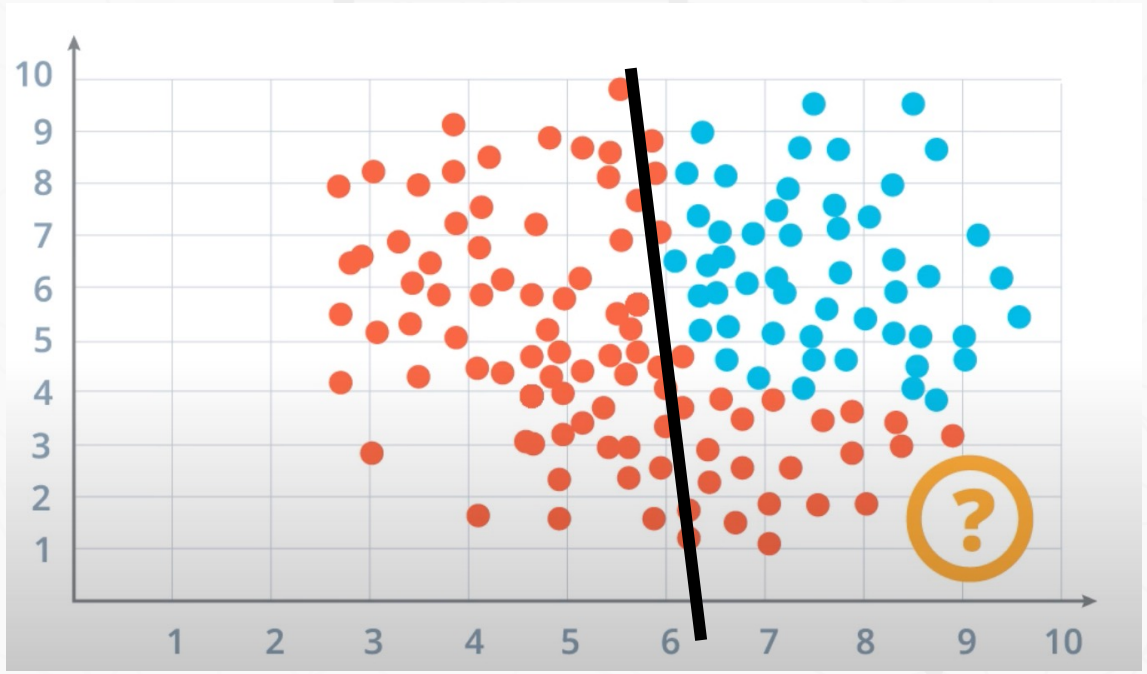
Límites de decisión no lineales

notas

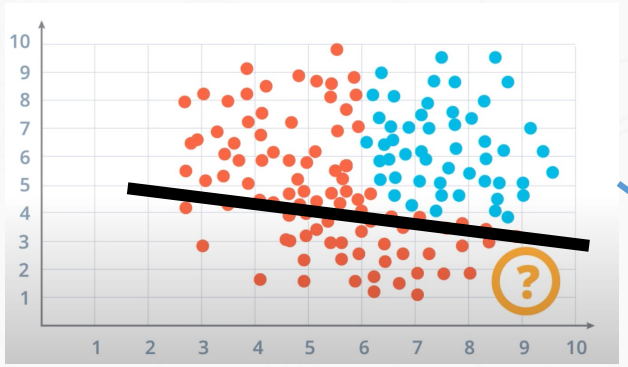


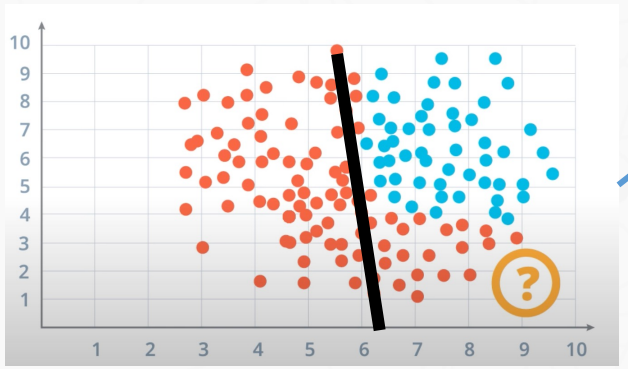
ensayo

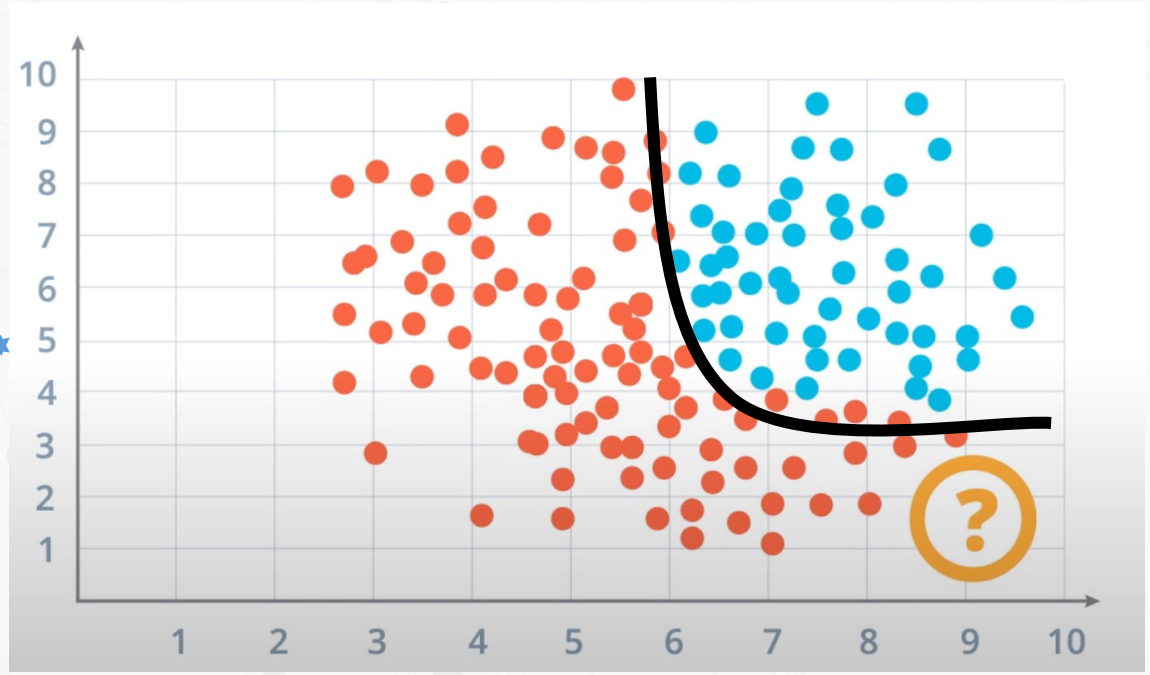
Límites de decisión no lineales



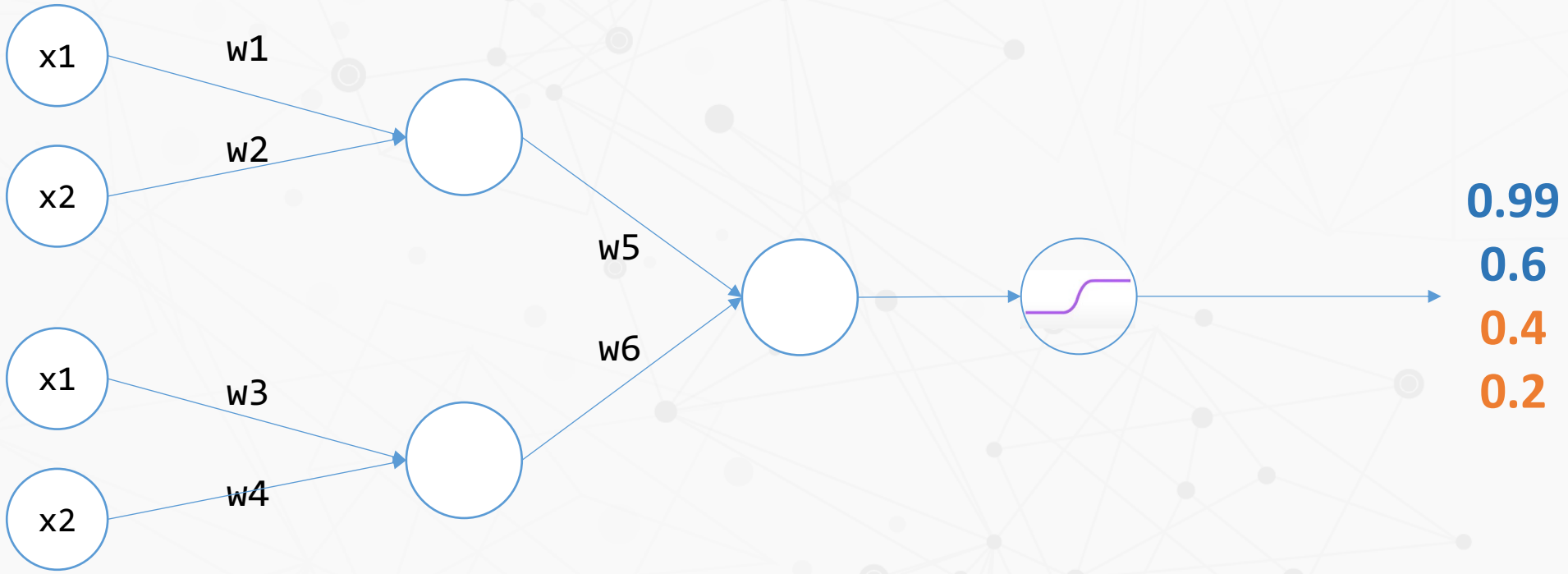
Límites de decisión no lineales



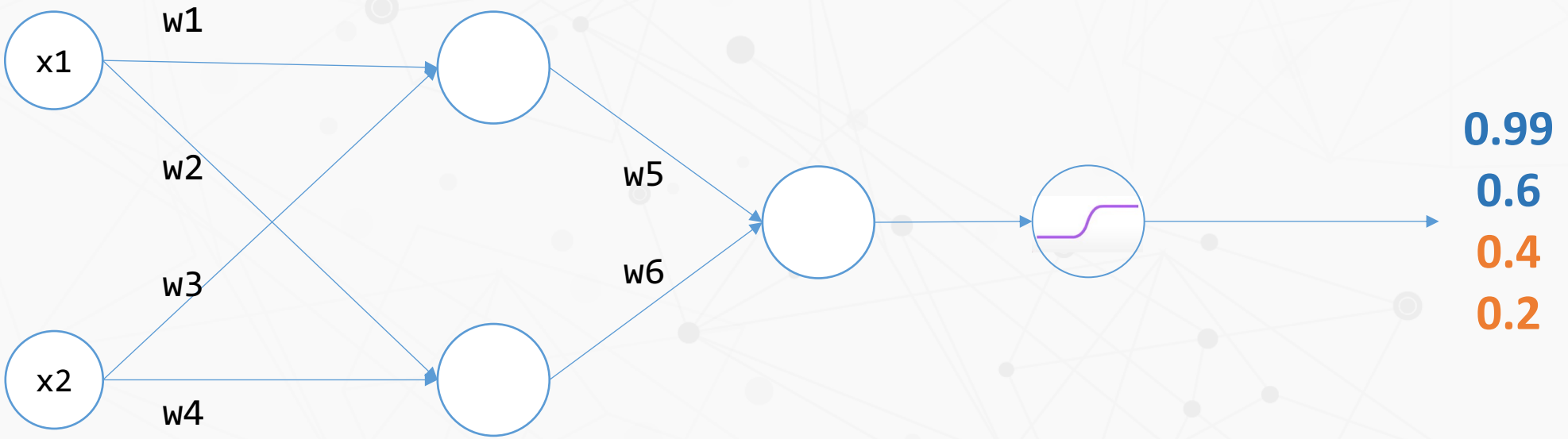




Arquitectura de redes neuronales

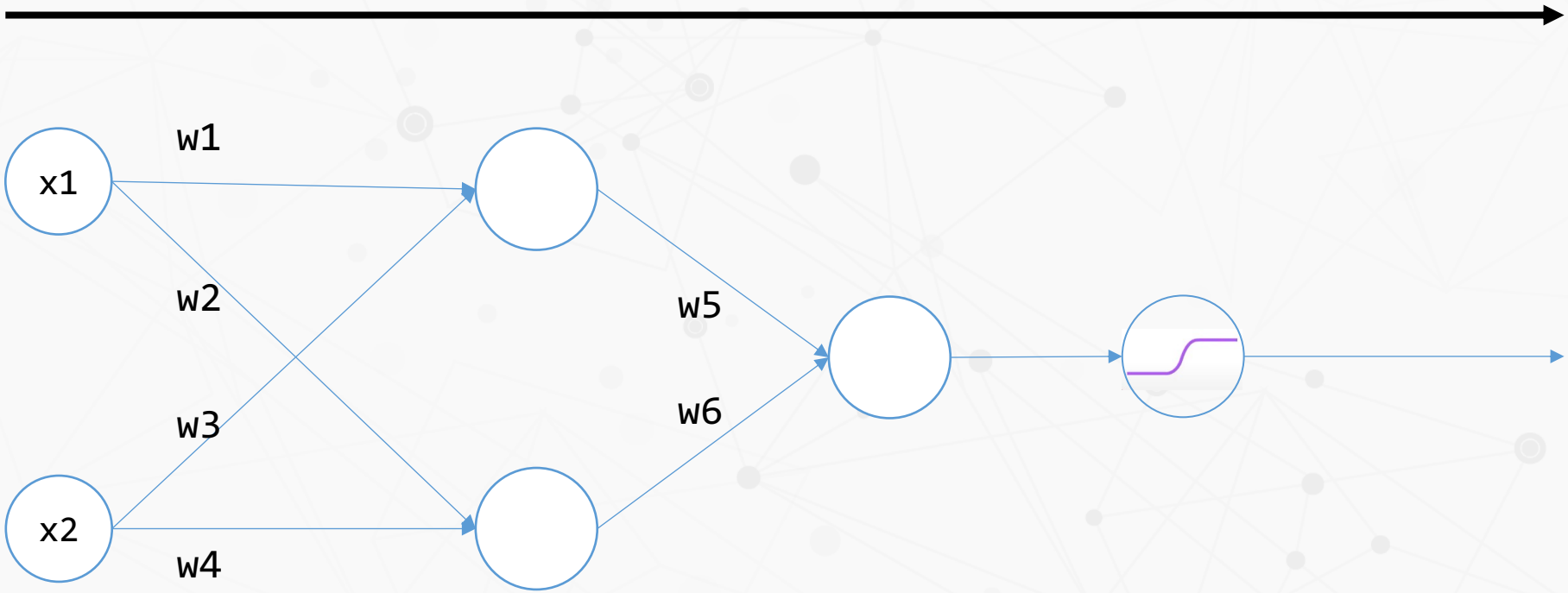


Arquitectura de redes neuronales

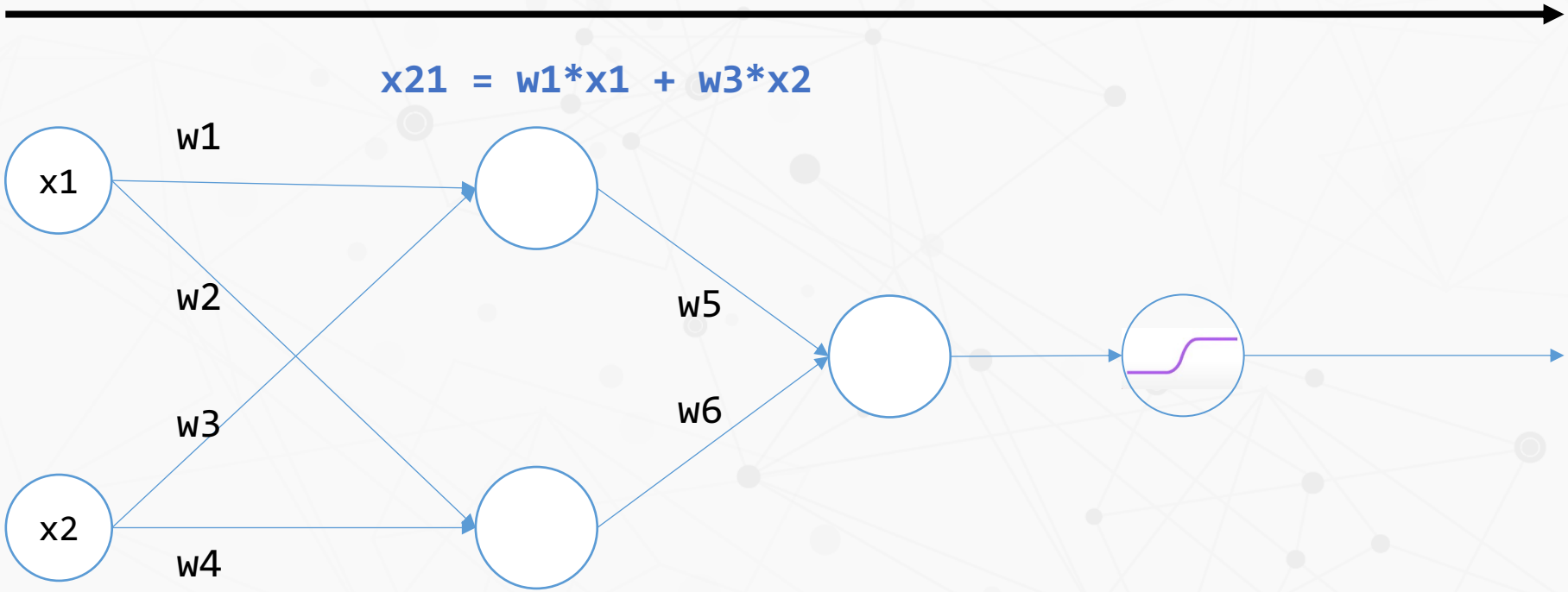


Feedforward y Backpropagation

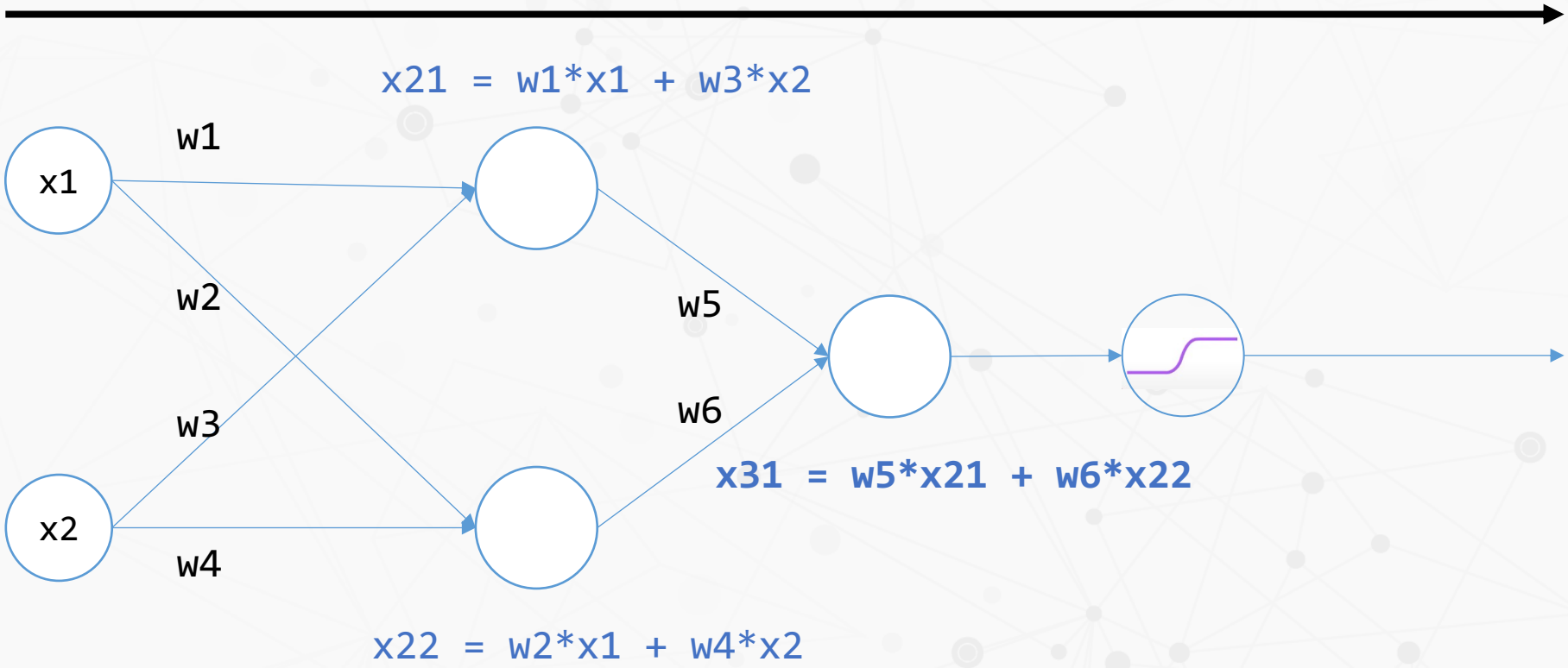
Feedforward



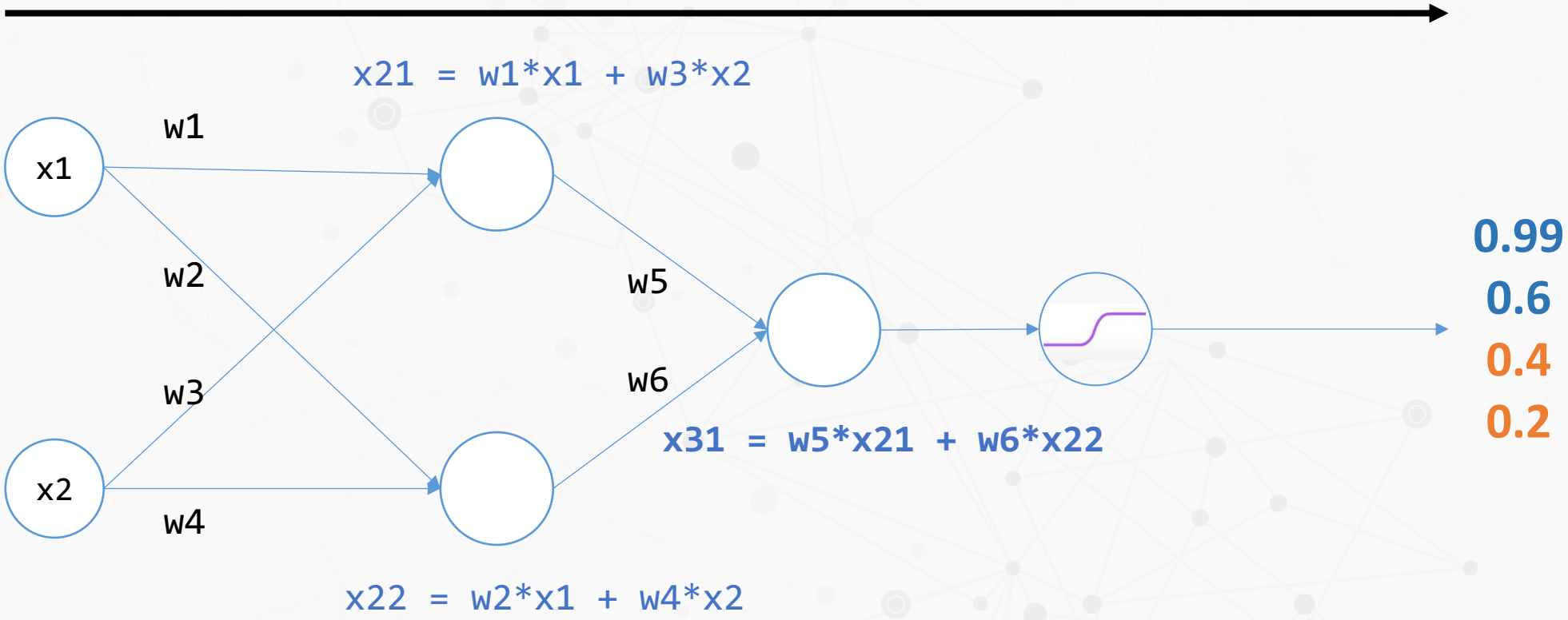
Feedforward



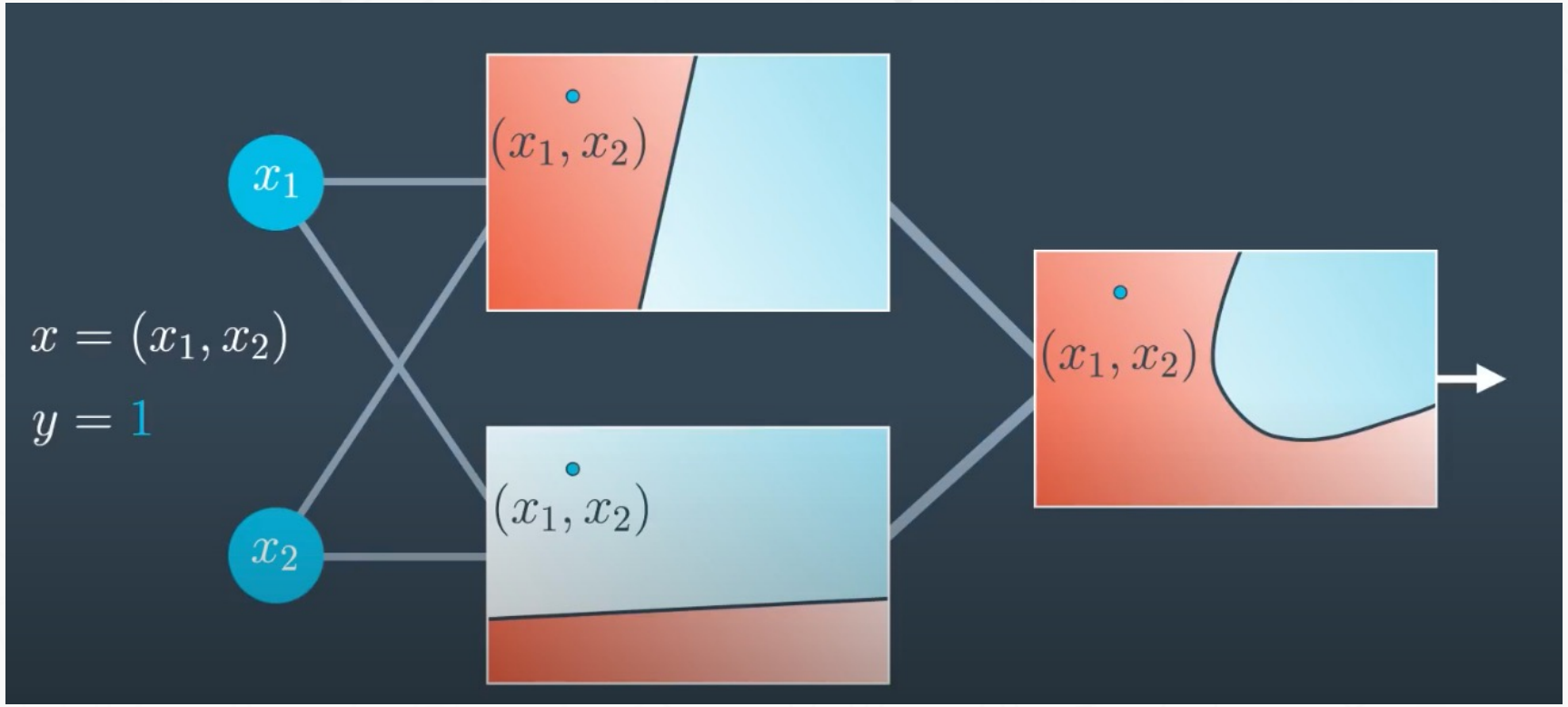
Feedforward



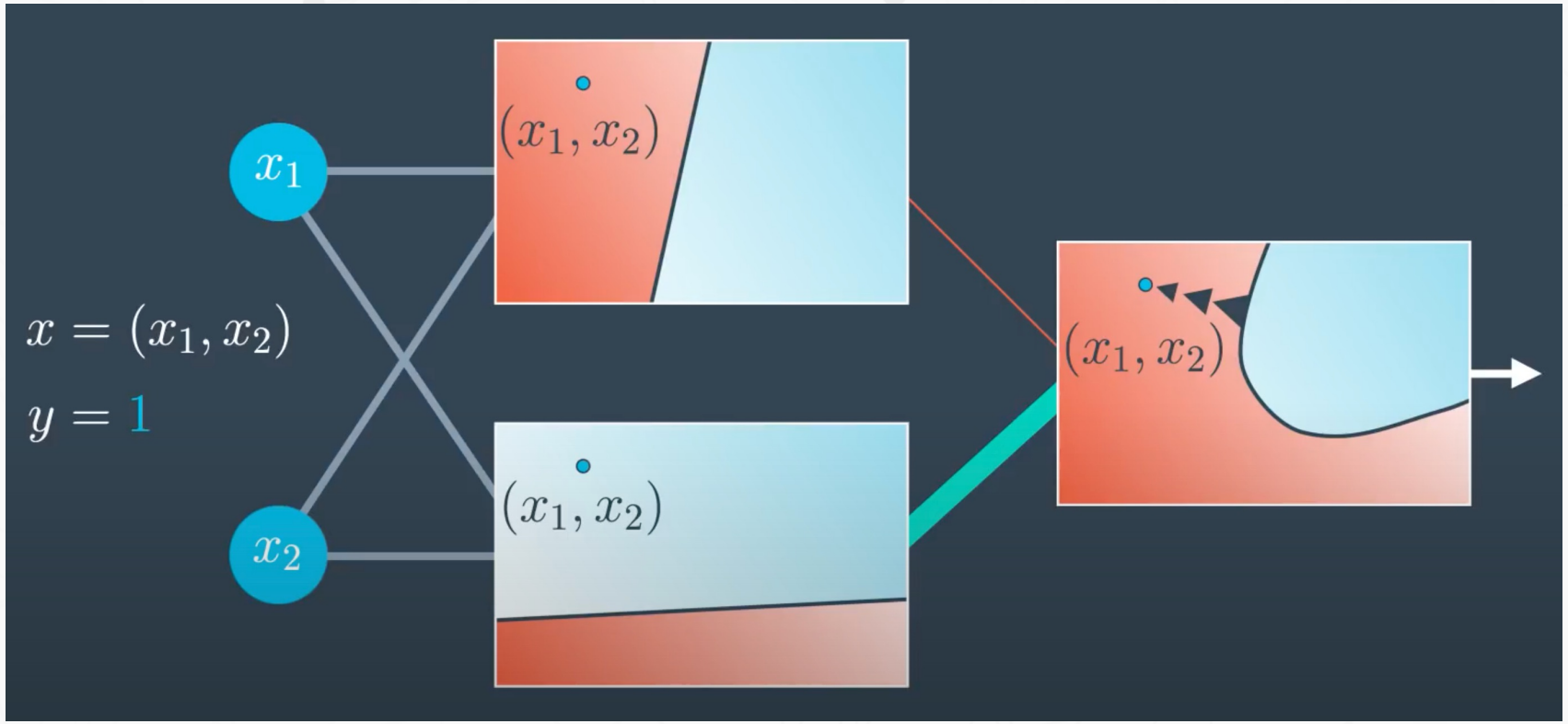
Feedforward



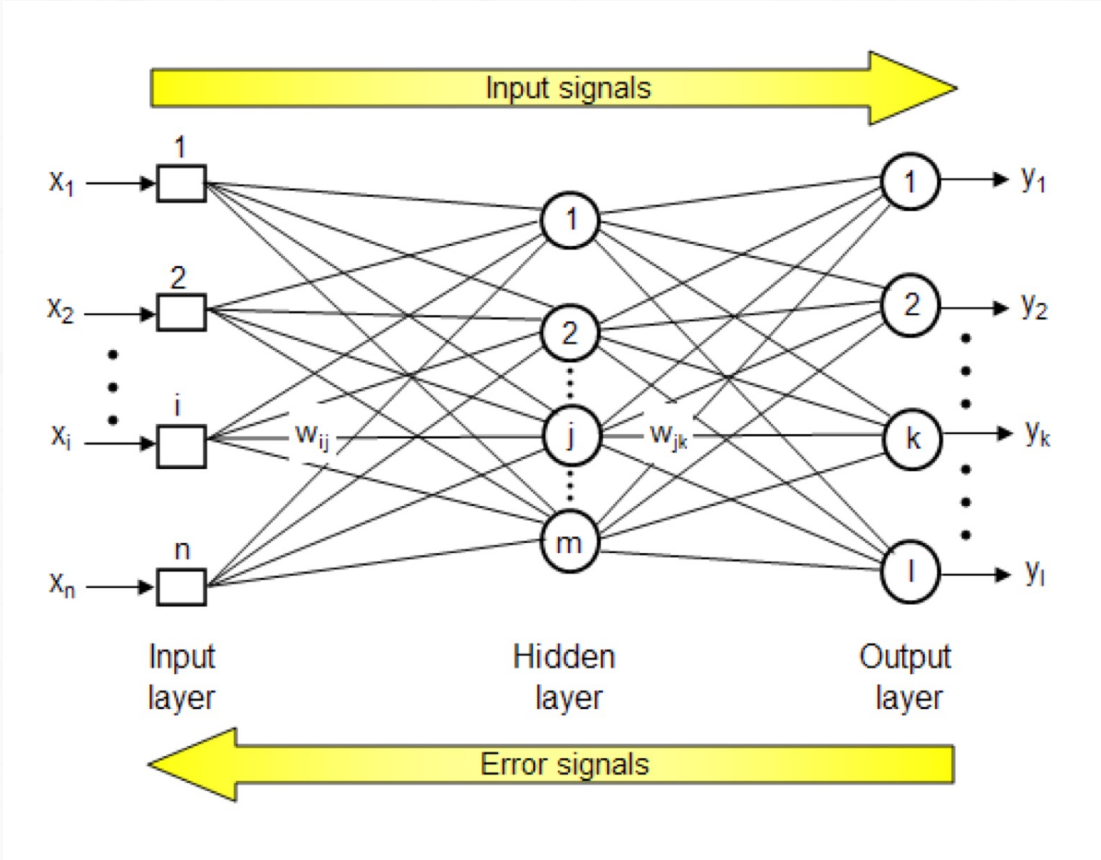
Backpropagation



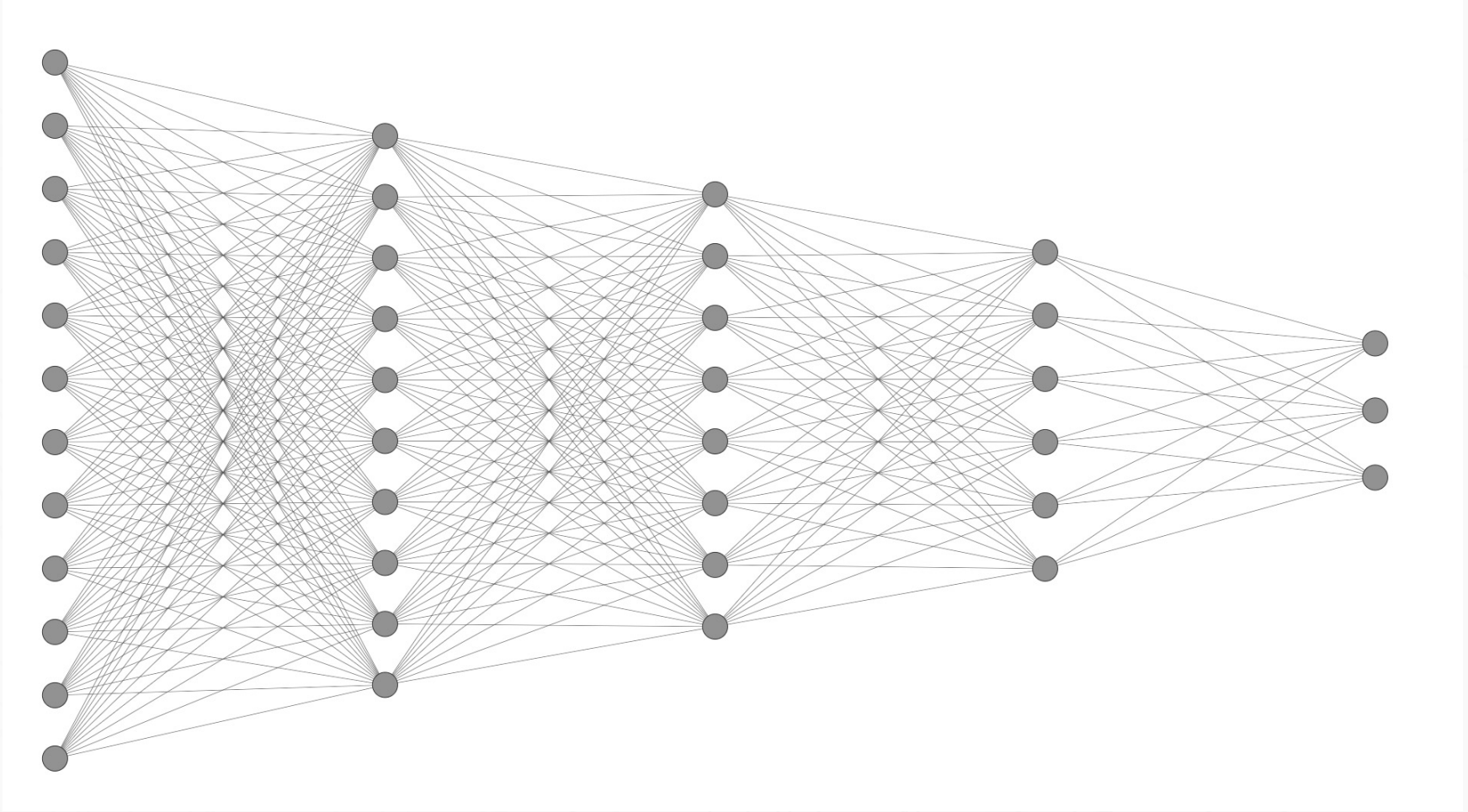
Backpropagation

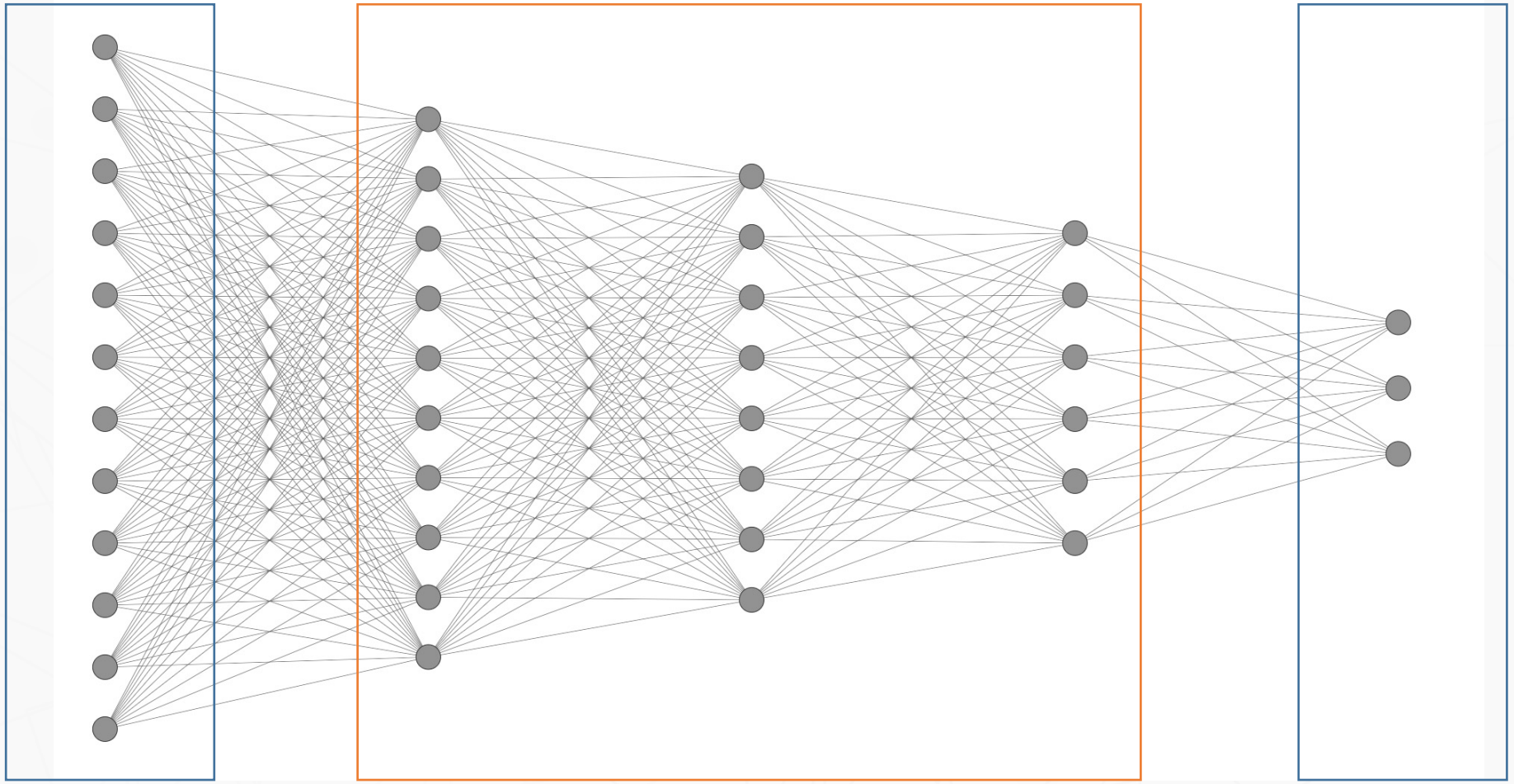


Feedforward y backpropagation



Arquitectura Fully Connected





Input Layer
Capa de entrada

Hidden Layers
Capas intermedias

Output Layer
Capa de salida

Implementación en PyTorch

```
from torch import nn

class Classifier(nn.Module):
    def __init__(self):
        super().__init__()
        self.fc1 = nn.Linear(12, 10)
        self.fc2 = nn.Linear(10, 8)
        self.fc3 = nn.Linear(8, 6)
        self.fc4 = nn.Linear(6, 3)
```

Primera capa: 12 a 10

Segunda capa: 10 a 8

Tercera capa: 8 a 6

Última capa: 6 a 3

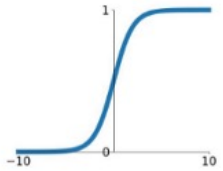
Funciones de activación

Son ecuaciones matemáticas que se aplican a las capas de las redes neuronales para modificar los valores de salida y contribuir a un mejor entrenamiento y convergencia.

La elección de la función de activación depende del caso de uso y de si la capa es intermedia o final.

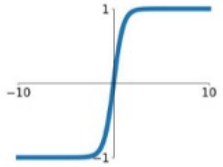
Sigmoid

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



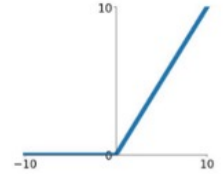
tanh

$$\tanh(x)$$



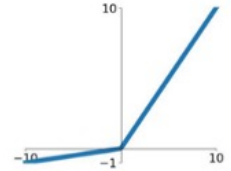
ReLU

$$\max(0, x)$$



Leaky ReLU

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

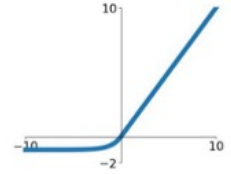


Maxout

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

ELU

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



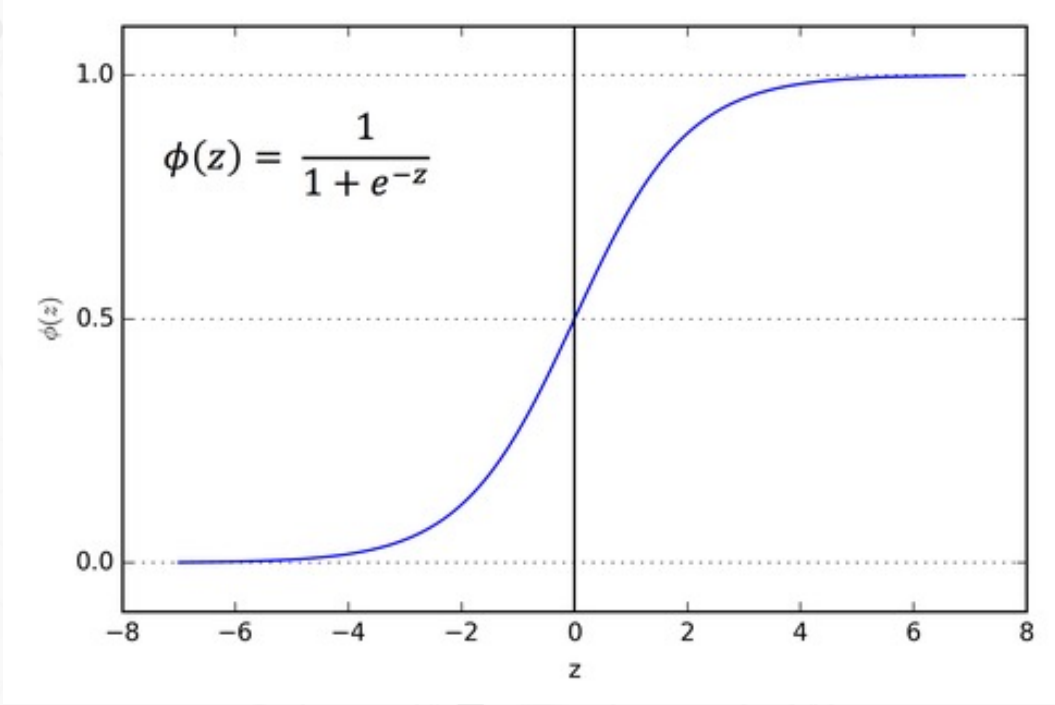
Funciones de activación

Sigmoidal

Recibe valores reales y devuelve valores continuos entre 0 y 1. Se suele utilizar en capas finales para targets binarios

Los valores negativos se convierten en valores cercanos a 0

Los valores positivos se convierten en valores cercanos a 1



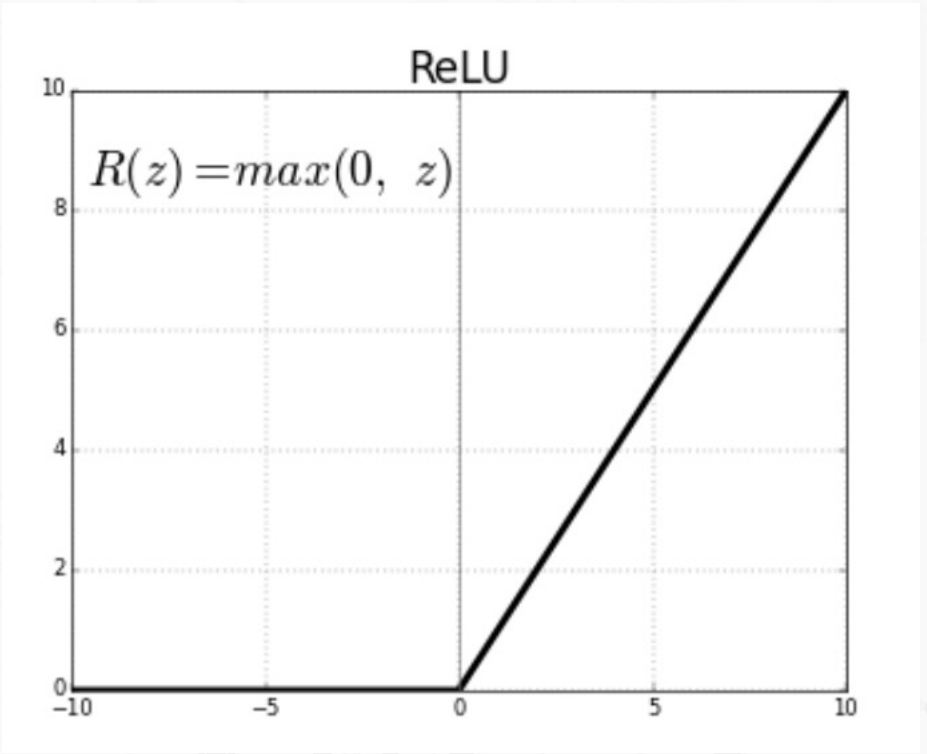
Funciones de activación

ReLU (Rectified Linear Units)

Una de las funciones de activación más utilizadas en capas intermedias

Los valores negativos se convierten en 0

Los valores positivos mantienen su valor



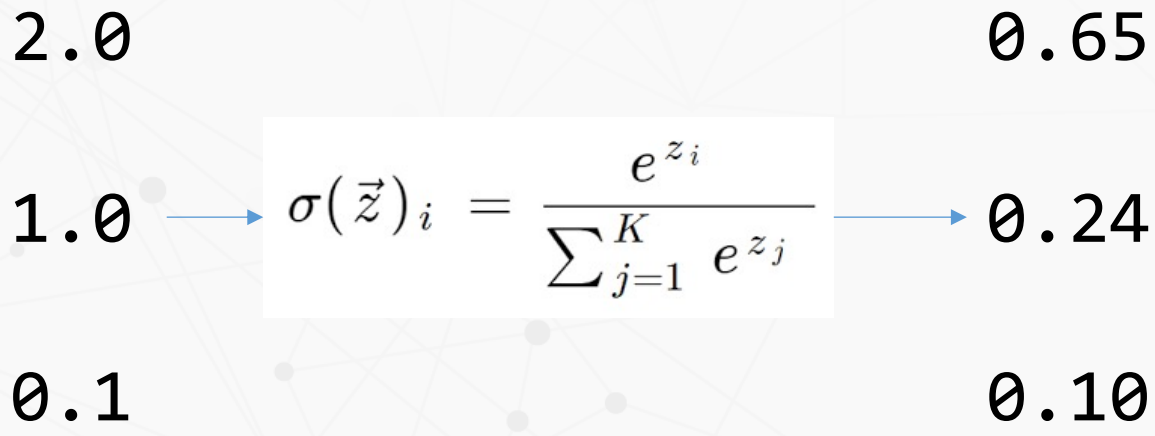
Funciones de activación

Softmax

Se utiliza usualmente en la última capa de una red para obtener las probabilidades de cada clase

Los valores obtenidos son las probabilidades para cada clase y suman 1

Se elige como predicción la clase de mayor probabilidad



Ejemplo: MNIST

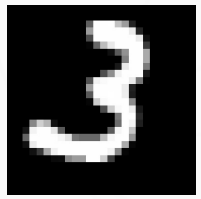
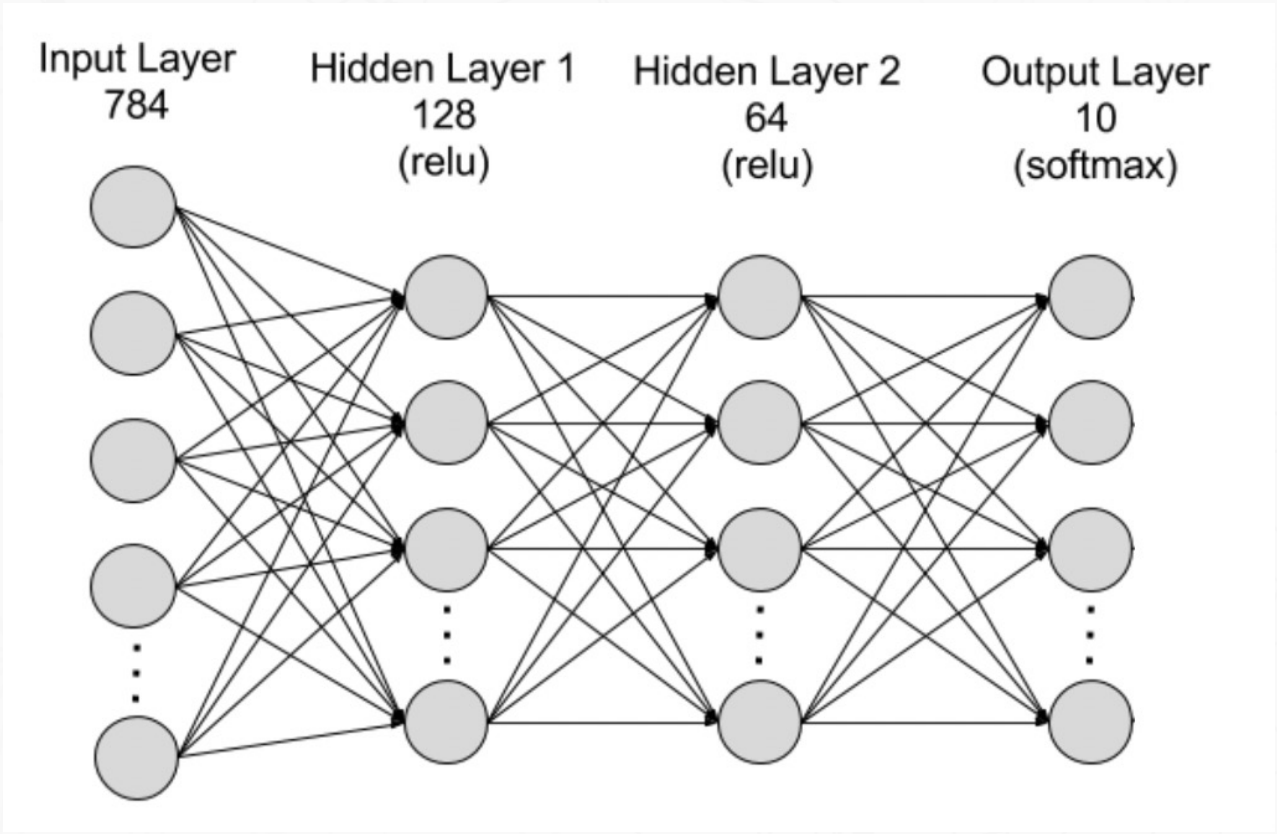


Imagen de
28x28 píxeles



Ejemplo: MNIST

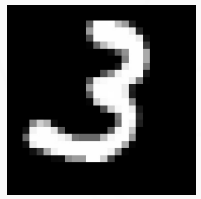
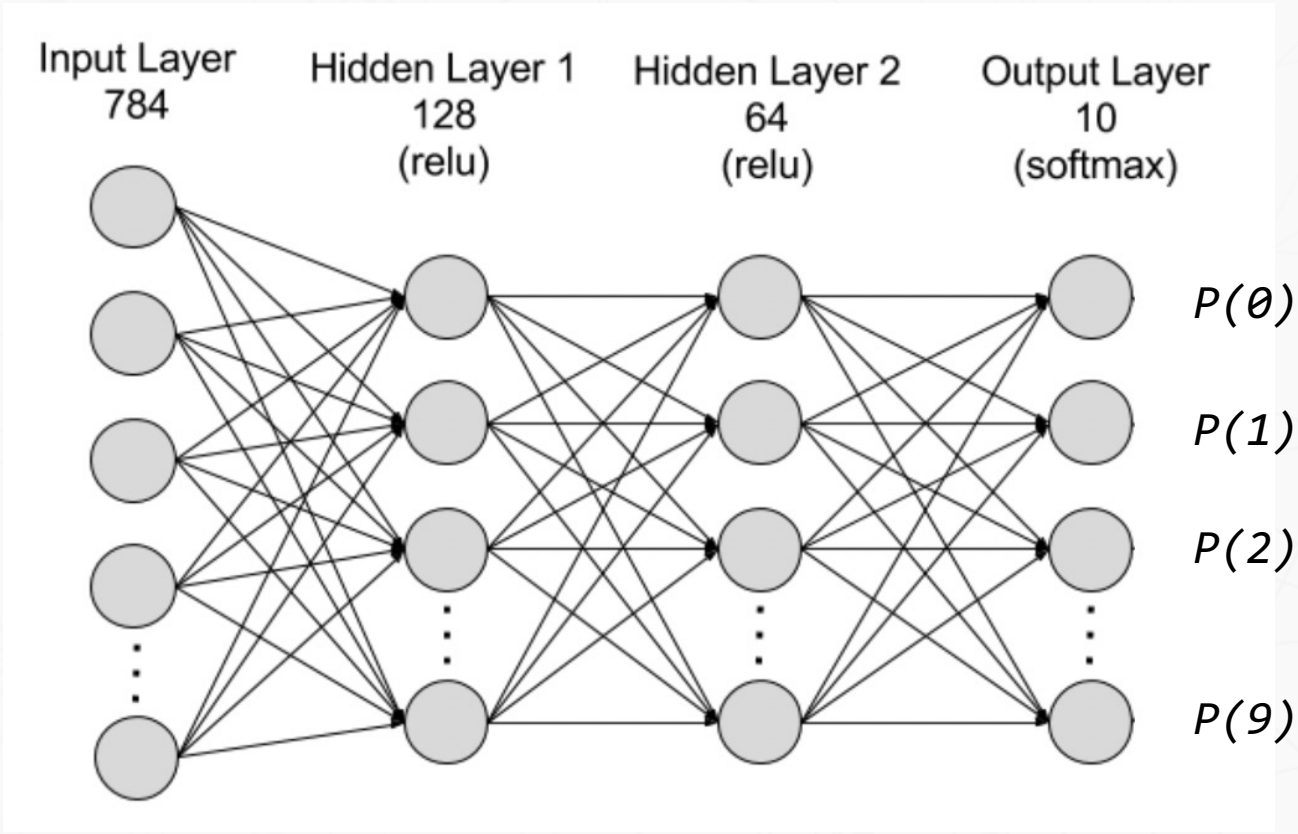


Imagen de
28x28 píxeles



Ejemplo: MNIST

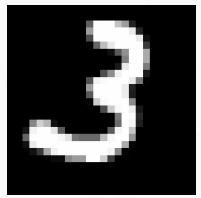
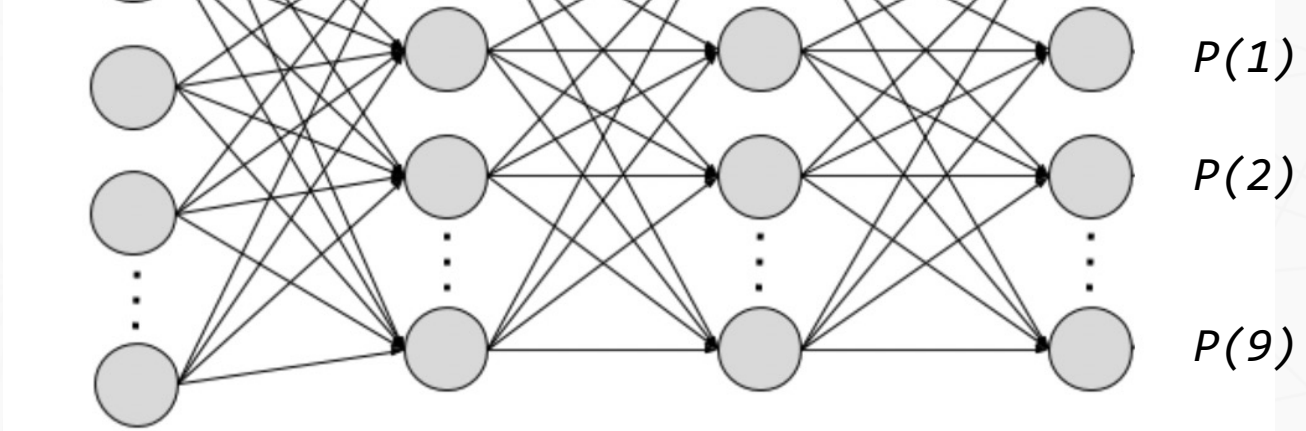


Imagen de
28x28 píxeles



Predicción p	Target y
P(0)	0
P(1)	0
P(2)	0
P(3)	1
P(4)	0
P(5)	0
P(6)	0
P(7)	0
P(8)	0
P(9)	0

Cálculo de función de costo

Ejemplo: MNIST

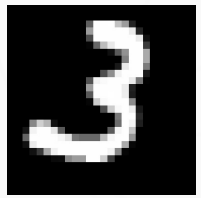
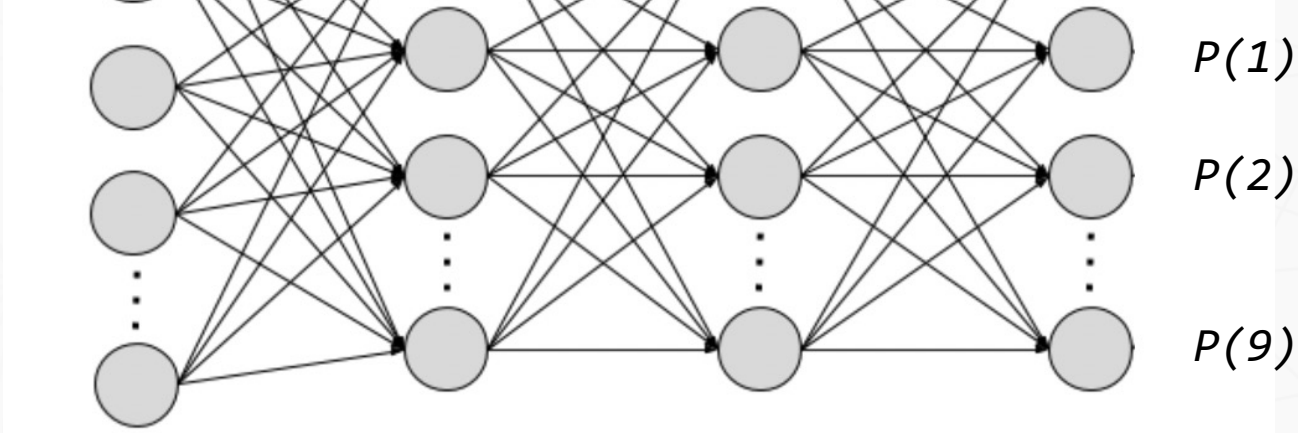


Imagen de
28x28 píxeles



Backpropagation

Predicción p	Target y
P(0)	0
P(1)	0
P(2)	0
P(3)	1
P(4)	0
P(5)	0
P(6)	0
P(7)	0
P(8)	0
P(9)	0

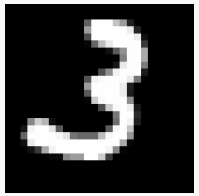
Cálculo de función de costo

Función de costo: Cross Entropy

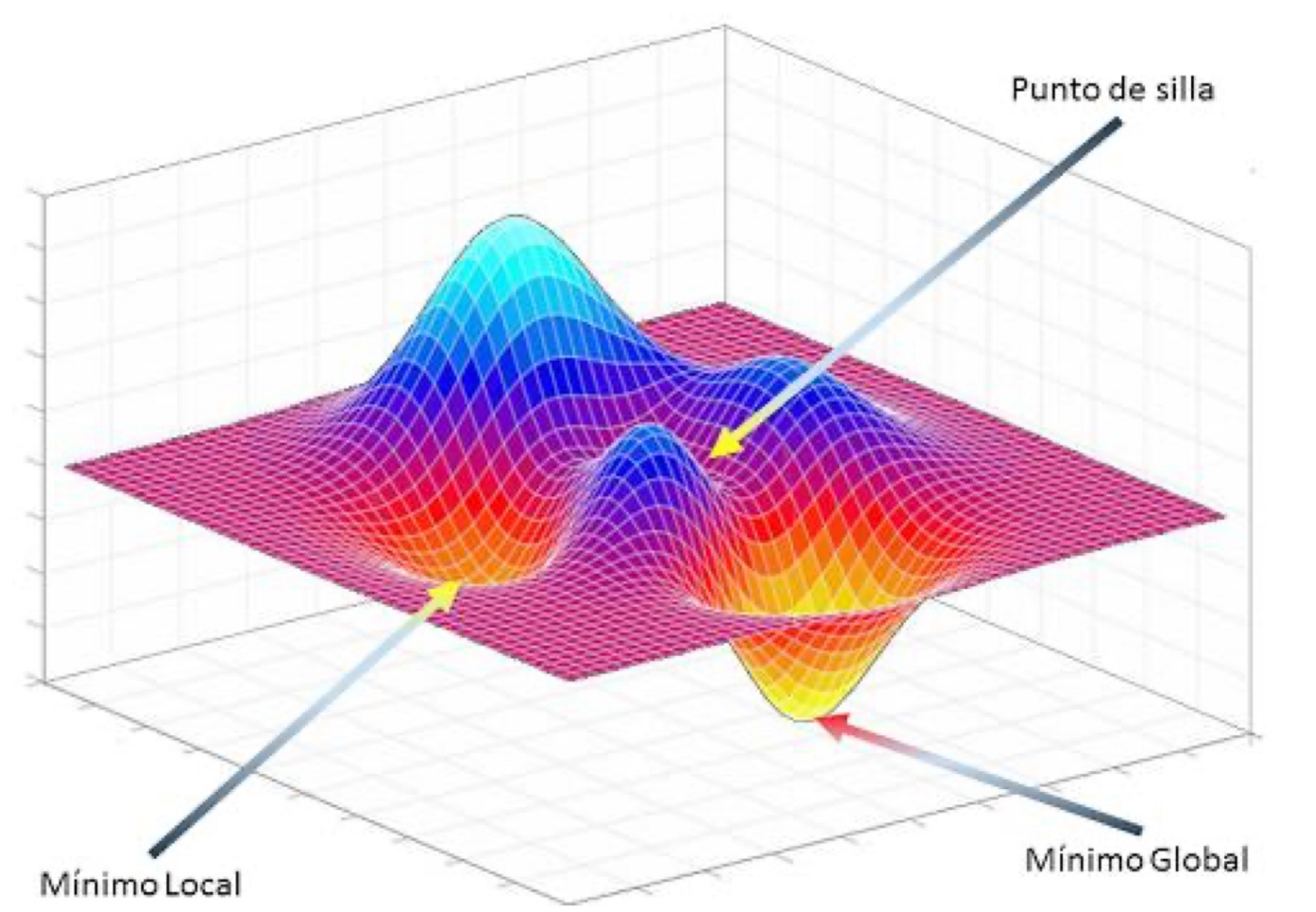
	p
P(0)	0.0
P(1)	0.0
P(2)	0.0
P(3)	0.9
P(4)	0.0
P(5)	0.0
P(6)	0.0
P(7)	0.0
P(8)	0.1
P(9)	0.0

$$CE(p, y) = -\sum y * \log(p)$$

y
0
0
0
1
0
0
0
0
0
0



Gradiente descendiente



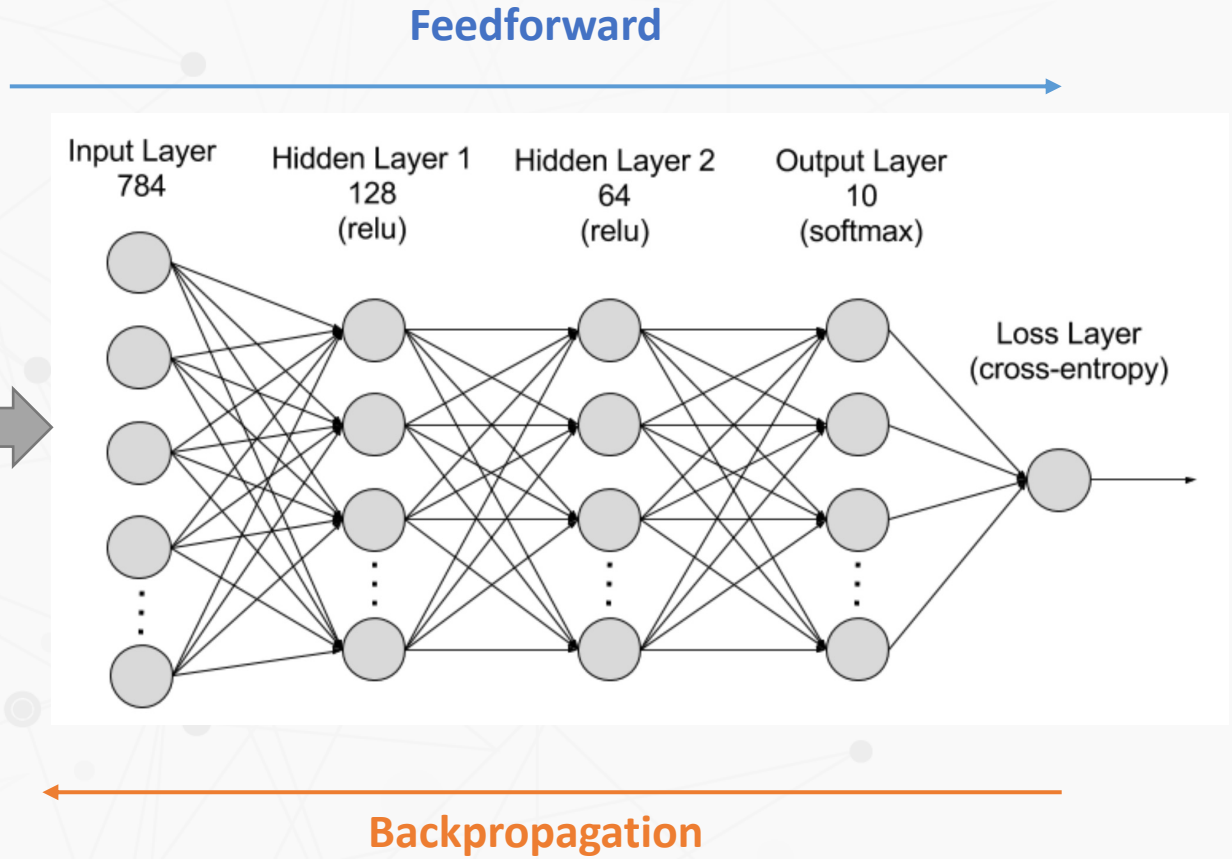
Parámetros de entrenamiento

Epochs:

Una época equivale a una iteración de feedforward y backpropagation de un dataset COMPLETO

Cada iteración implica un ajuste de los pesos de la red para minimizar la función de costo

Más épocas -> Más ajuste (Cuidado con el overfitting)

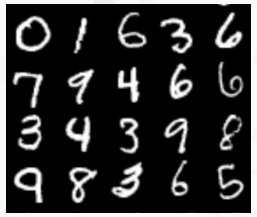


Parámetros de entrenamiento

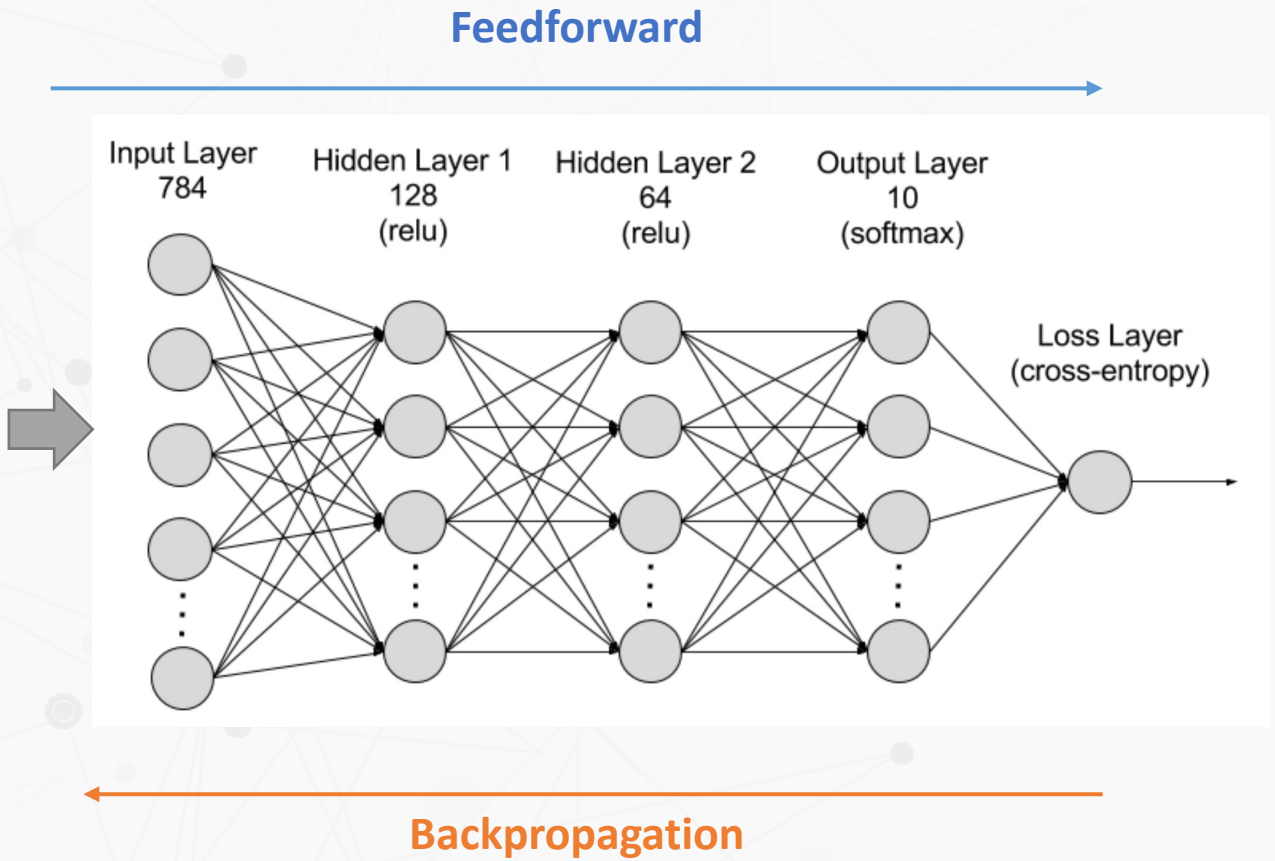
Batch size:

No se puede utilizar todo el dataset en cada iteración porque implicaría muchos recursos

El dataset se divide en grupos o **batches** de un tamaño específico y este número de registros se utiliza en cada época



Batch size

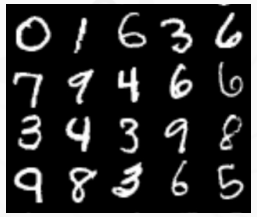


Parámetros de entrenamiento

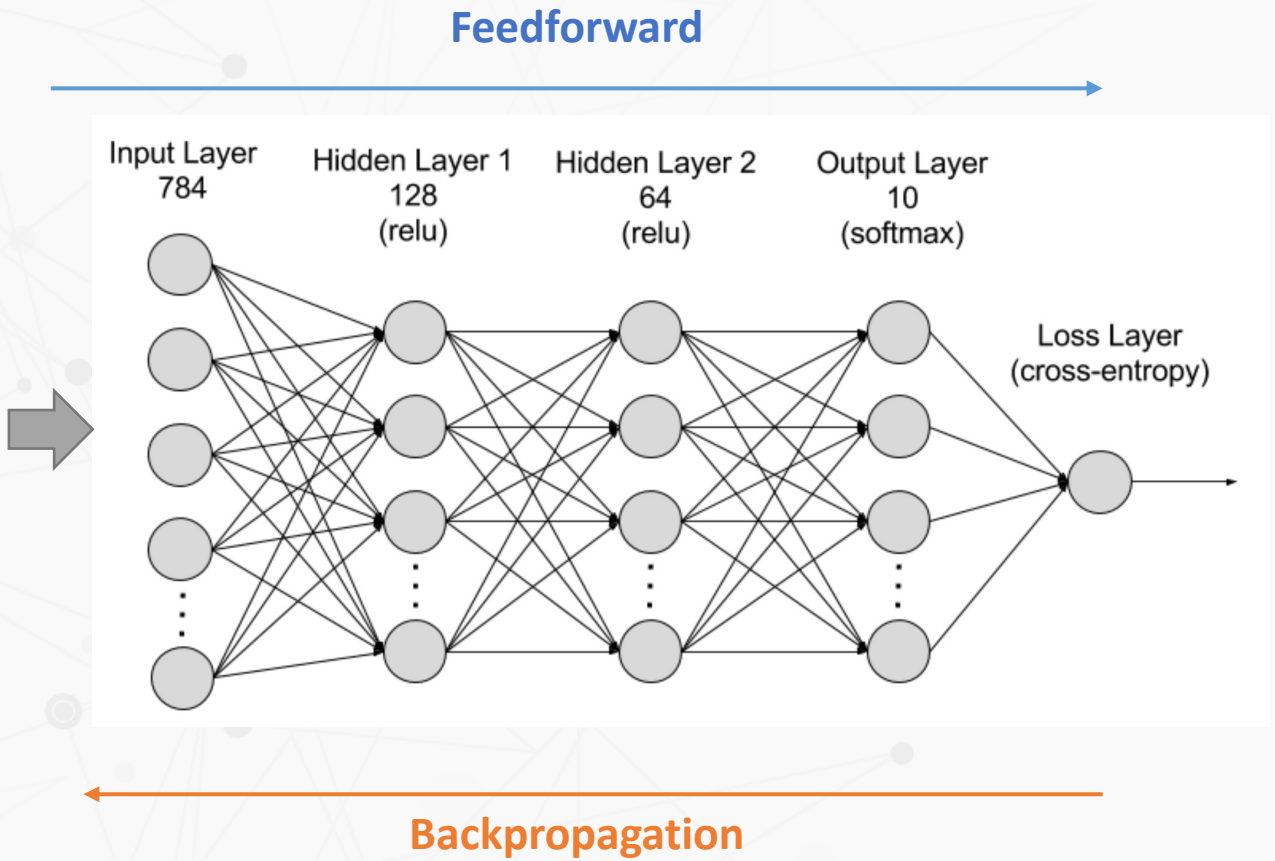
Por ejemplo, si tenemos un dataset de 2000 registros:

Para que se cumpla una época estos 2000 registros deben pasar por un feedforward y backpropagartion

Si elegimos un batch size de 100, para poder completar una época es necesario que se realicen 20 iteraciones.



Batch size

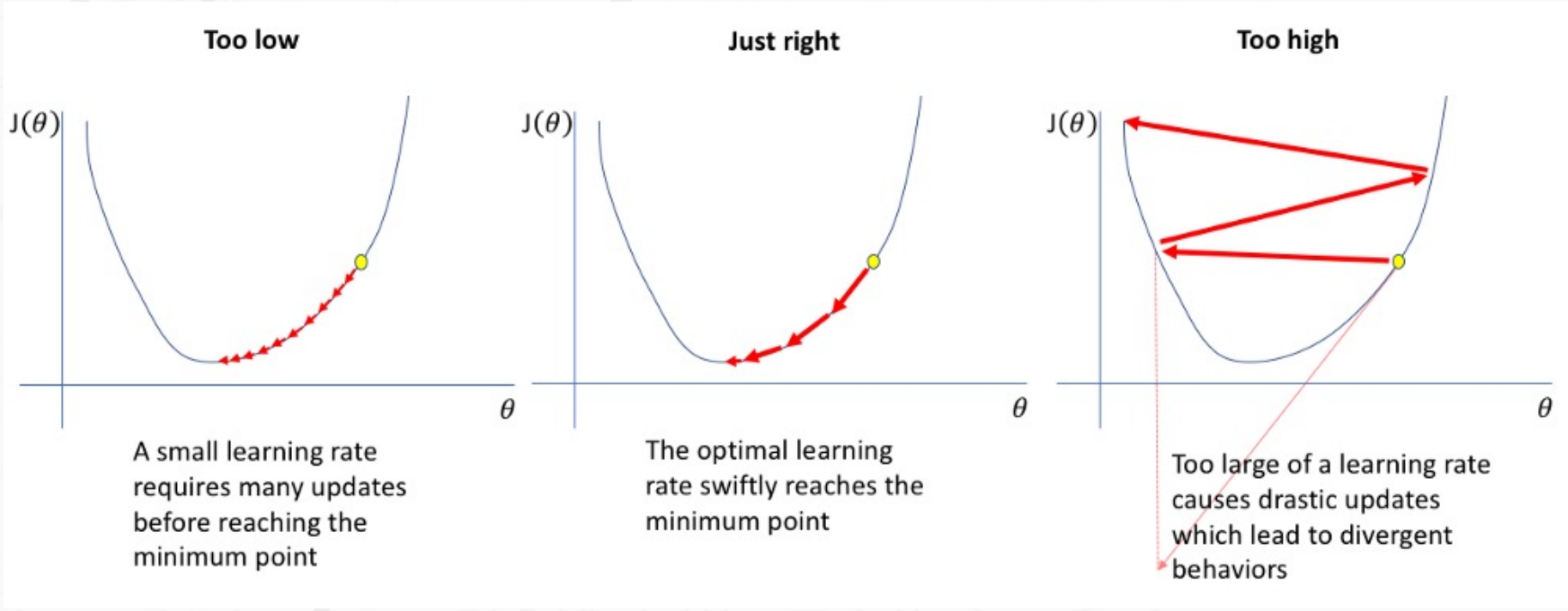


Parámetros de entrenamiento

Learning Rate

En cada iteración se ajustan los pesos de la red para reducir la función de costo.

El learning rate nos permite controlar la intensidad de este ajuste y varía de acuerdo al caso de uso



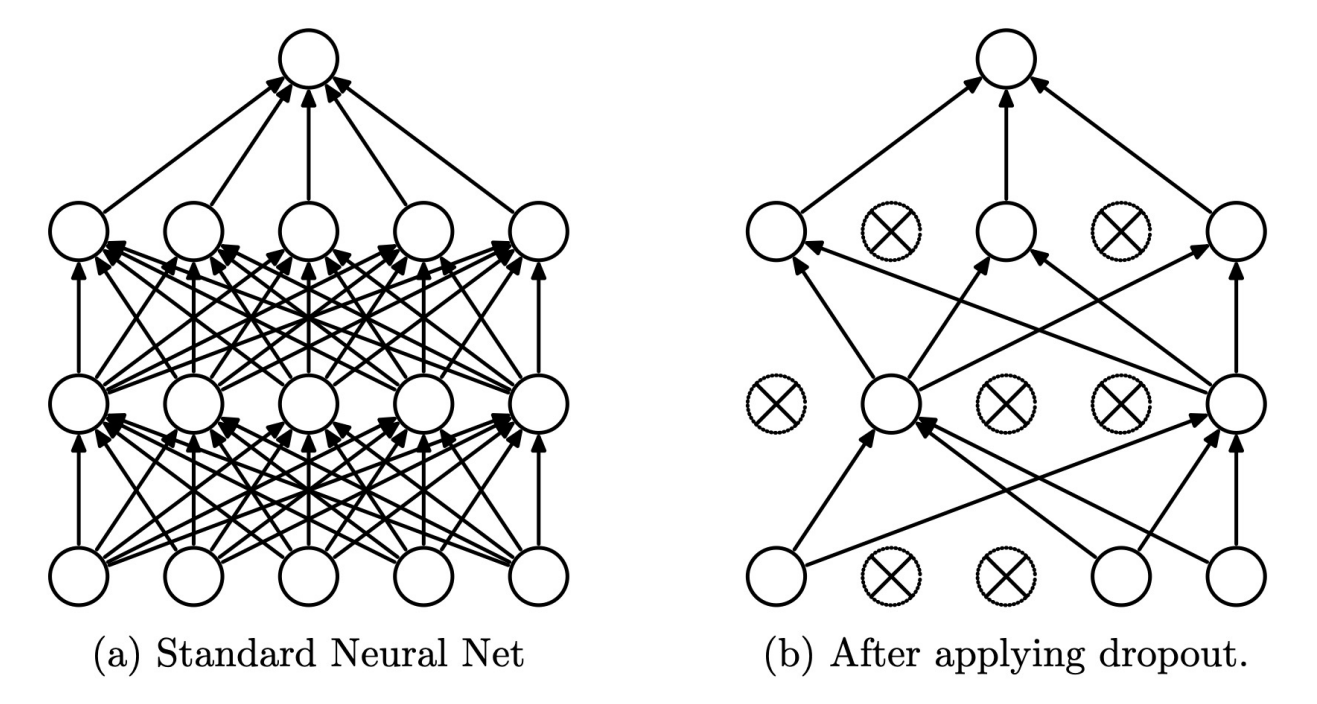
Parámetros de entrenamiento

Dropout

Las redes neuronales pueden aprender patrones muy complejos entre los inputs y outputs, lo cual las hace muy susceptibles a overfitting.

Una forma de reducir este efecto es desactivar aleatoriamente algunas neuronas durante el entrenamiento

La elección de qué neuronas desactivar se rige por un parámetro de probabilidad llamado **dropout**



(a) Standard Neural Net

(b) After applying dropout.

Recursos complementarios

MIT Introduction to deep learning
https://www.youtube.com/watch?v=njKP3FqW3Sk&list=PLtBw6njQRU-rwp5__7C0oIVt26ZgjG9NI&index=1

Why are deep neural networks hard to train?
http://neuralnetworksanddeeplearning.com/chap5.html

Dropout: A Simple Way to Prevent Neural Networks from Overfitting
https://www.cs.toronto.edu/~hinton/absps/JMLRdropout.pdf

Neural Networks & The Backpropagation Algorithm, Explained
https://ayearofai.com/rohan-lenny-1-neural-networks-the-backpropagation-algorithm-explained-abf4609d4f9d

¡Gracias!

Online | DMC Perú