REDES NEURONALES

CONJUNTO DE ALGORITMOS DE APRENDIZAJE SUPERVISADO PARA REGRESIÓN Y CLASIFICACIÓN BINARIA (O MULTICLASE)

DEEP LEARNING

ORÍGENES

BASÁNDOSE EN ESTUDIOS PROVENIENTES DE LA BIOLOGÍA, EN 1958 FRANK ROSENBLATT PRESENTA EL PERCEPTRÓN (SOBRE IDEAS ANTERIORES DE MC CULLOCH Y PITTS DE 1943)

FUE LA PRIMERA ‘PROTO’ RED NEURONAL. SE TRATA DE UN CLASIFICADOR BINARIO O DISCRIMINADOR LINEAL

https://web.archive.org/web/20180722124753/https://data-speaks.luca-d3.com/2018/07/historia-de-la-ia-frank-rosenblatt-y-el.html

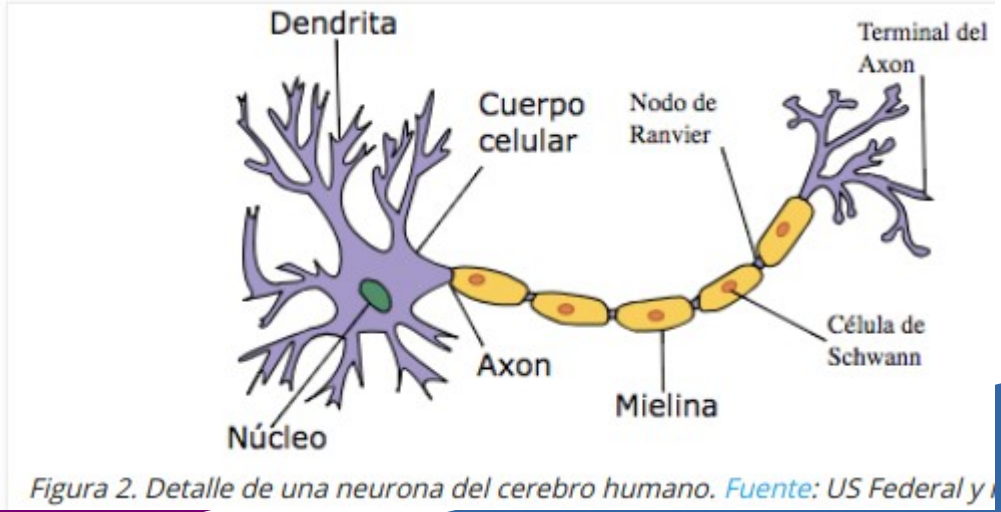
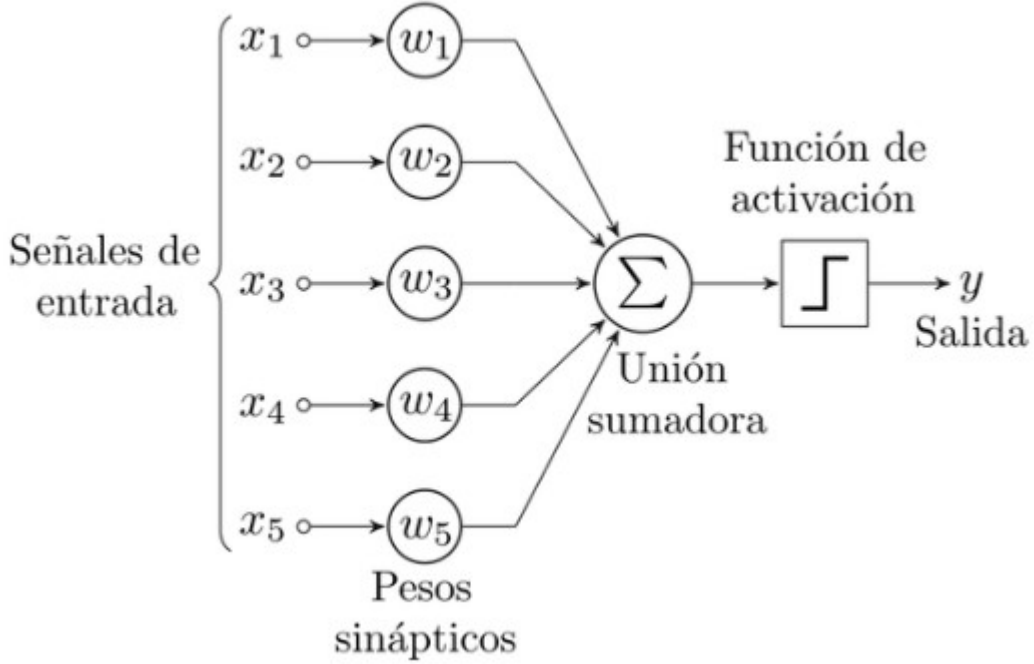


Figura 2. Detalle de una neurona del cerebro humano. Fuente: US Federal y

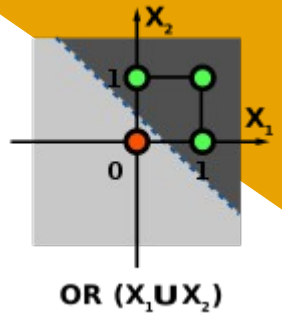


CÓMO FUNCIONA EL PERCEPTRÓN?

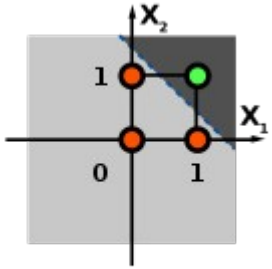
- LA INFORMACIÓN INGRESA A LA NEURONA Y CADA CARACTERÍSTICA xi ES MULTIPLICADA POR UN PESO wi (QUE PUEDE PENSARSE COMO UNA PERILLA QUE ‘HABILITA’ CADA VALOR CON UNA DADA ‘INTENSIDAD’)
- EN LA NEURONA SE SUMAN TODAS LAS CARACTERÍSTICAS ‘PESADAS’ POR LOS wi -» y = SUMA (wi * xi)
- A LA SALIDA TENEMOS ‘CASI’ UNA REGRESIÓN LINEAL, PERO NOS FALTA APLICAR EL UMBRAL DE LA RESPUESTA DE SALIDA
- SI SUMA(wi *xi) =< UMBRAL -> y = 0
- SI SUMA(wi *xi) => UMBRAL -> y = 1
- YA TENEMOS EL CLASIFICADOR!
-
- ESTE UMBRAL ES UNA FN ‘ESCALÓN’ (LLAMADA ‘FUNCIÓN DE ACTIVACIÓN’ POR LOS AMIGOS ;))

MÁS ALLÁ DEL PERCEPTRÓN

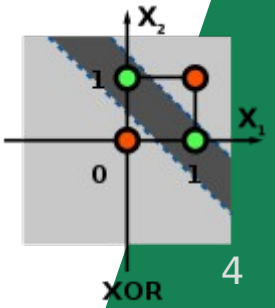
- EL PERCEPTRÓN NO ES CAPAZ DE RESOLVER PROBLEMAS QUE NO SON LINEALMENTE SEPARABLES… OTRA VEZ SOPA
- PERO… PODEMOS ENCADENAR VARIOS PERCEPTRONES (COMO LAS NEURONAS) Y CAMBIARLE LA FUNCIÓN DE ACTIVACIÓN Y AHÍ LA COSA CAMBIA!
- TENEMOS AL ‘PERCEPTRÓN MULTICAPA’ (MLP) QUE SÍ PUEDE CON ESTO ---->



OR ($X_1 \cup X_2$)



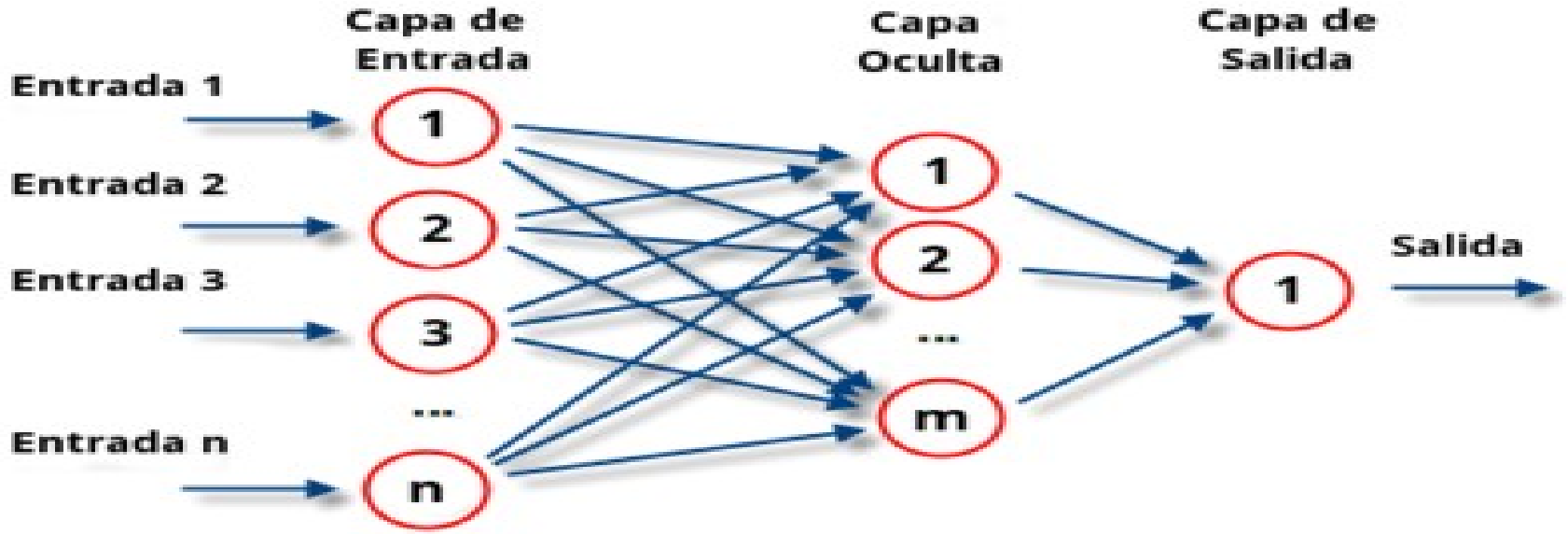
AND ($X_1 \cap X_2$)



XOR

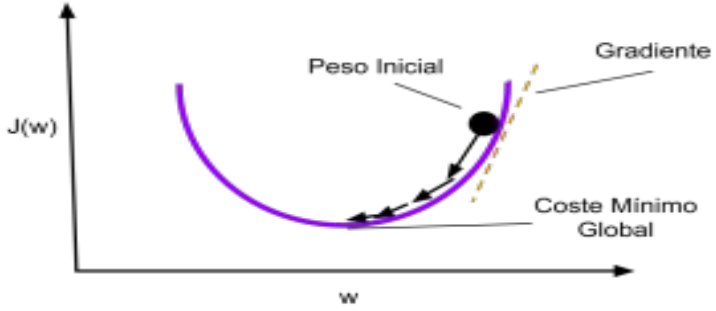
CÓMO CONECTAMOS A LOS PERCEPTRONES MULTICAPA?

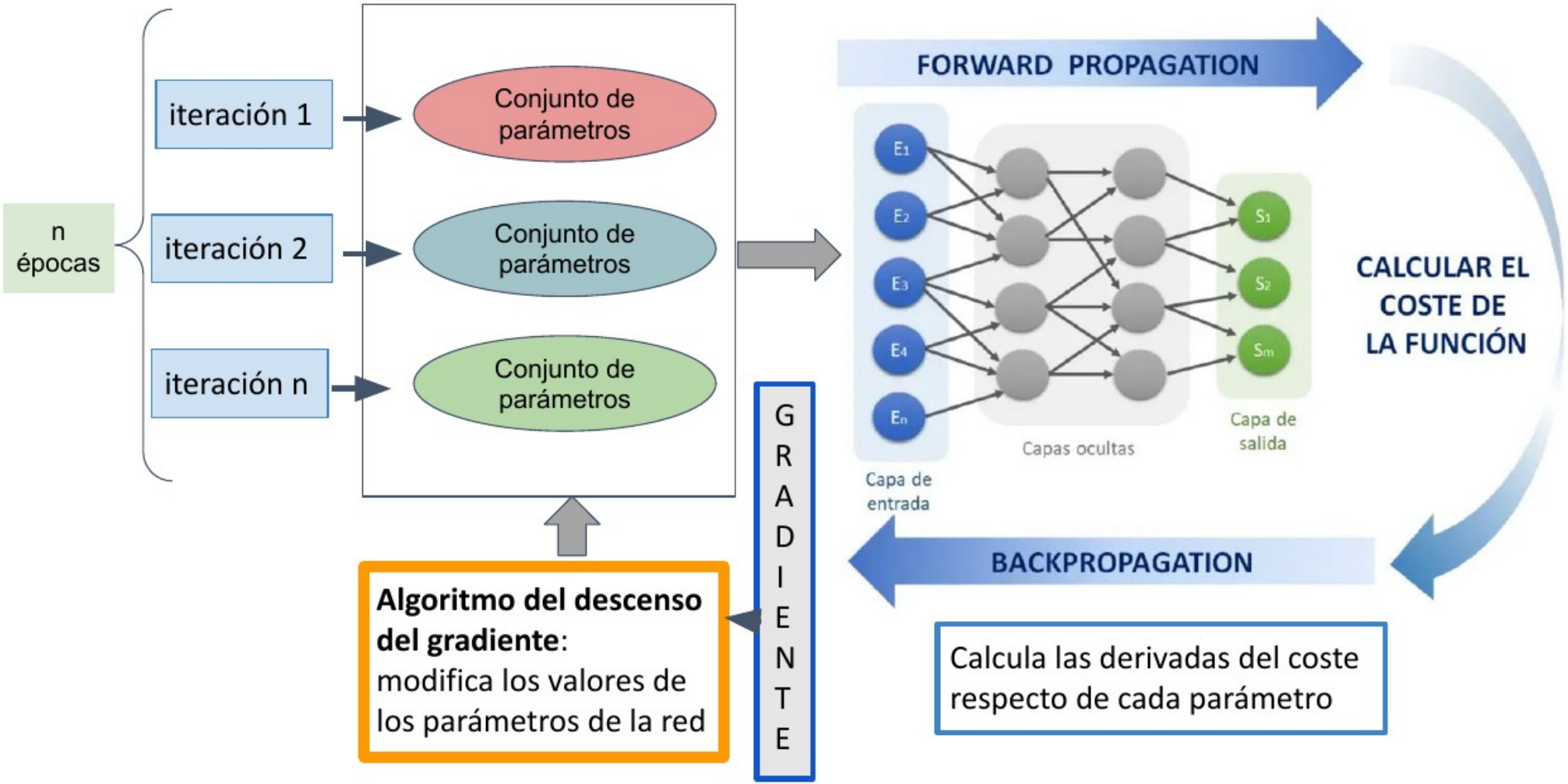
- EN LA MISMA COLUMNA: OBTENEMOS UNA ‘CAPA’
- EN LA MISMA FILA: CONECTAMOS CAPA CON CAPA



MIRANDO ADENTRO DE LA CAJA NEGRA…

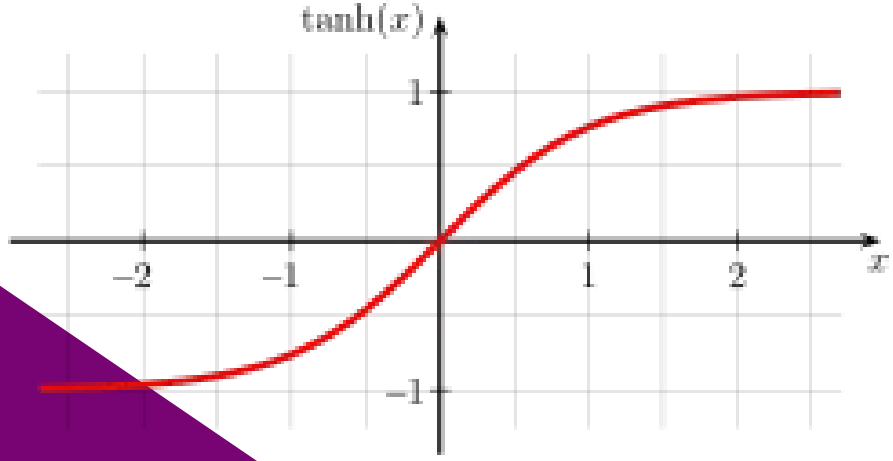
- TODO BIEN, PERO CÓMO LO HACE?
- PROPAGACIÓN HACIA ADELANTE: LOS PESOS wi Y EL BIAS
- PROPAGACIÓN HACIA ATRÁS: PARA PODER APRENDER DE LOS DATOS, DEFINIMOS UNA FUNCIÓN DE COSTO (EL ERROR CUADRÁTICO MEDIO) QUE ES LA QUE DEBE HACERSE MÍNIMA (PARA QUE EL RESULTADO PREDICHO NO SE ALEJE DEL VERDADERO) ---> APARECE EL ’DESCENSO DEL GRADIENTE’ COMO TÉCNICA PARA BUSCAR ESE MÍNIMO.

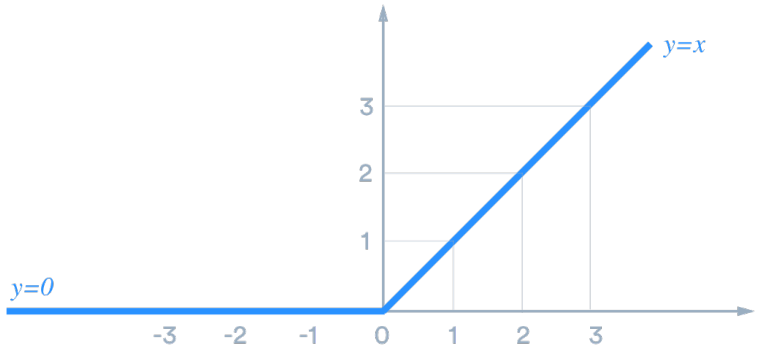



n épocas
iteración 1
iteración 2
iteración n
Conjunto de parámetros
Conjunto de parámetros
Conjunto de parámetros
FORWARD PROPAGATION
E1
E2
E3
E4
En
S1
S2
Sm
Capa de entrada
Capas ocultas
Capa de salida
CALCULAR EL COSTE DE LA FUNCIÓN
BACKPROPAGATION
Calcula las derivadas del coste respecto de cada parámetro
GRADIENTE
Algoritmo del descenso del gradiente: modifica los valores de los parámetros de la red

FUNCIONES DE ACTIVACIÓN

- SIGMOIDE: f(t) = 1/(1 + EXP(-t))
- TANH: f(t)= (EXP(t) – EXP(-t))/ (EXP(t) + EXP(-t))
- RELU: f(t) = max(0, t)

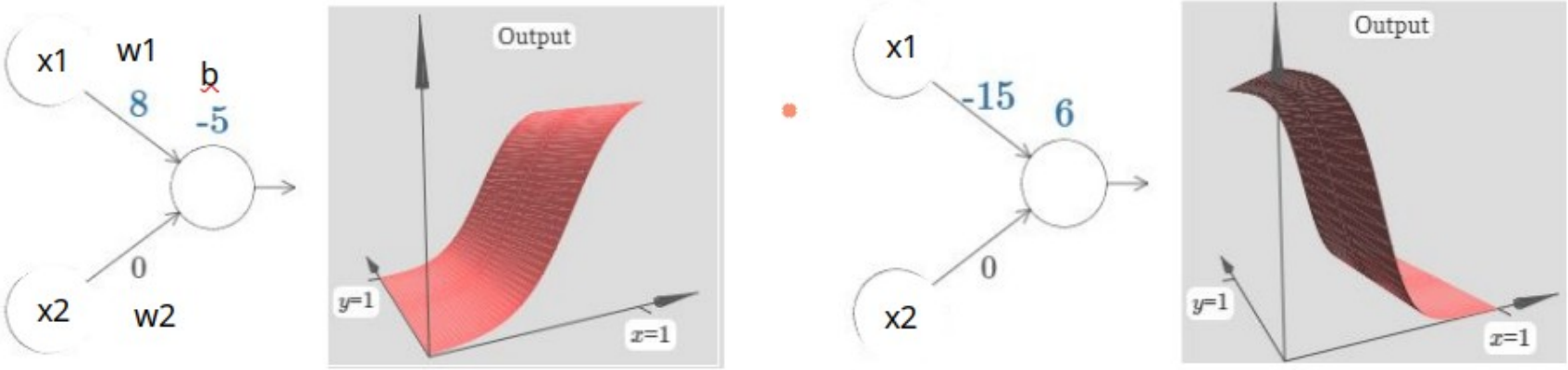




Los “w” y “b” afectan a la función de activación

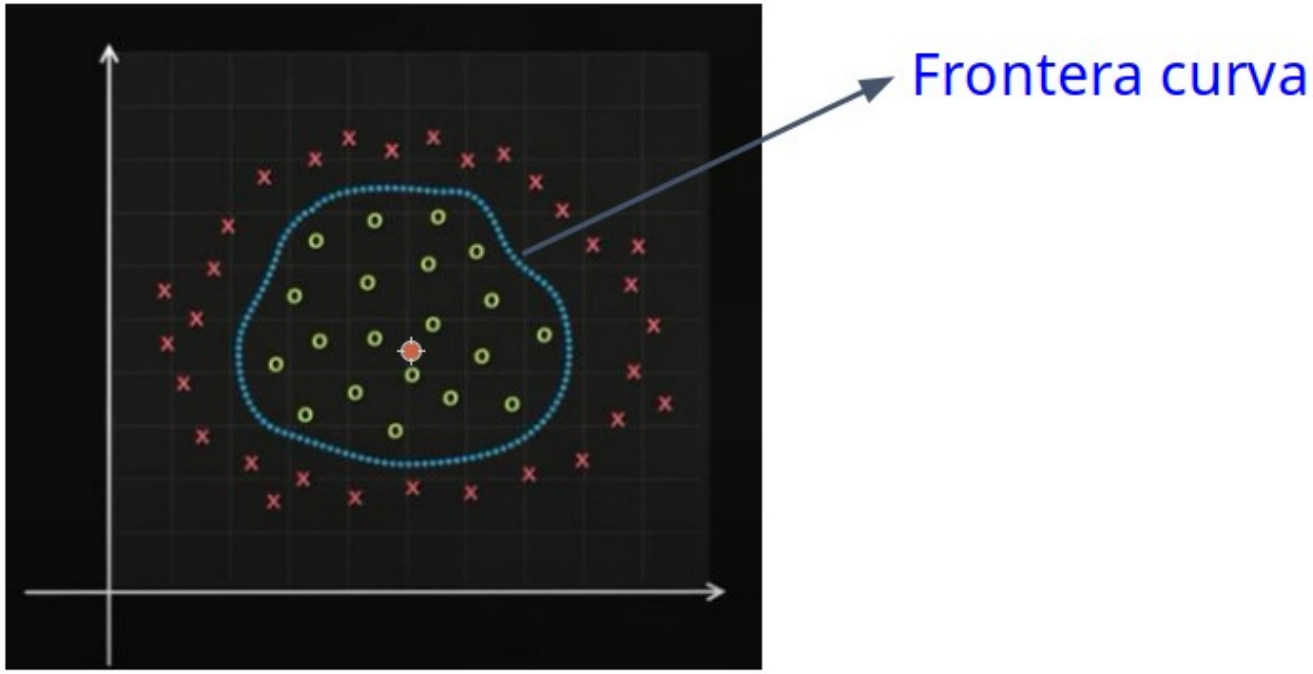
La función de activación se aplica a la suma ponderada:

$$f\left(\sum_j w_j x_j + b\right)$$



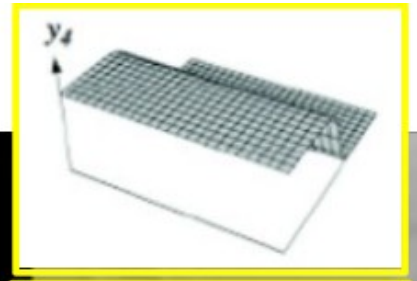
- Si modificamos “b” se desplaza la curva en el eje
- Si modificamos “w” se modifica la pendiente.

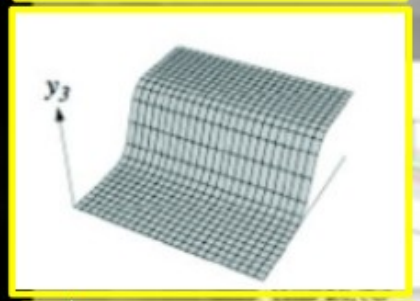
¿Cómo resolvemos esta situación?

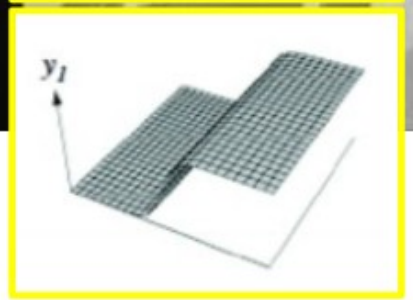


Lo podemos solucionar encadenando varias neuronas al mismo tiempo

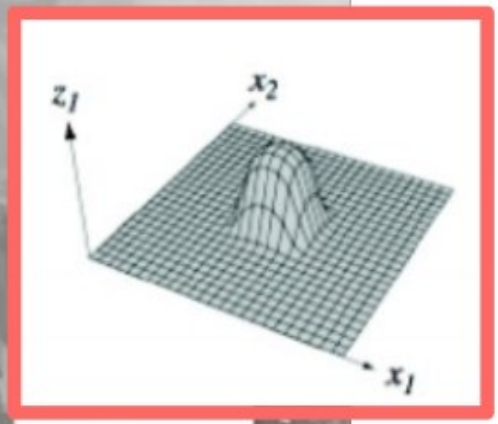
Una solución posible

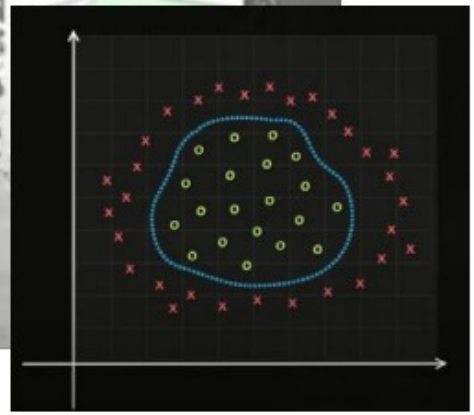






Combinación de las 4 figuras geomét



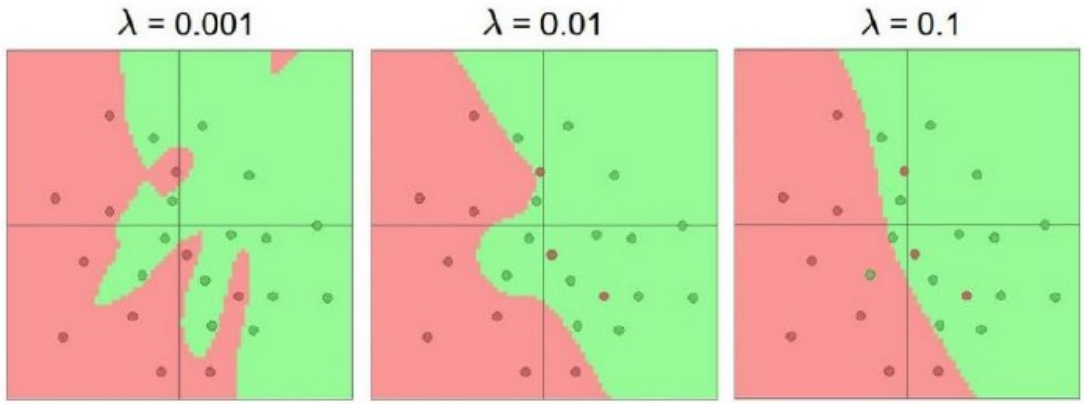


La regularización

La regularización es un modo para restringir la complejidad de un modelo.
Es útil cuando hay mucha correlación entre características, para filtrar ruido en los datos y también para evitar el Overfitting.

Aprendizaje= Loss + Penalización

$$Loss = \underbrace{\frac{1}{n}\sum_{i}^{n} E_i}_{\text{Error plano}} + \underbrace{\lambda \sum_{j}^{m} w_j^2}_{\text{Penalización}}$$



Los hiperparámetros

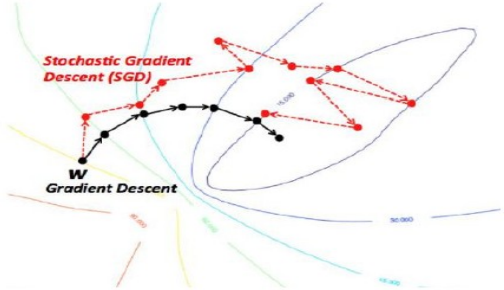
Los hiperparámetros son los que guían el algoritmo de optimización de nuestro modelo.
No existe una configuración ideal, sino que depende del modelo y del dataset específico.

- Cantidad de capas
- Cantidad de neuronas por capa.
- Función de activación
- Tasa de aprendizaje (Learning rate (alfa))
- Batch size
- Épocas
- Regularización

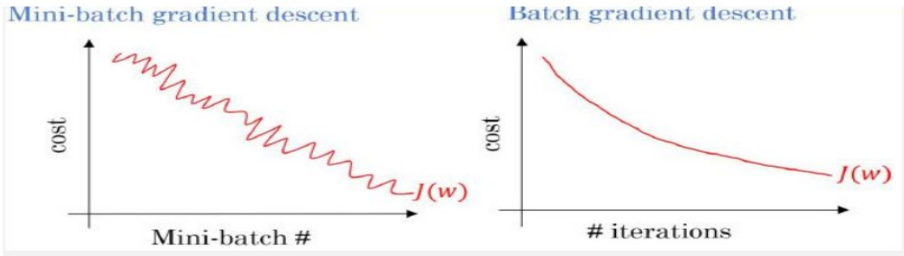
OPTIMIZADORES

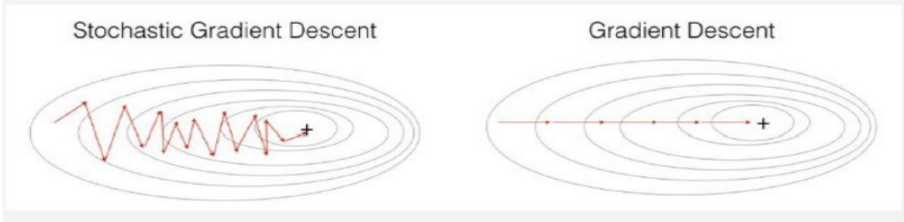
- https://velascoluis.medium.com/optimizadores-en-redes-neuronales-profundas-un-enfoque-pr%C3%A1ctico-819b39a3eb5
- DESCENSO DEL GRADIENTE (TRADICIONAL)
- DESCENSO DEL GRADIENTE ESTOCÁSTICO (SGD)
- ADAGRAD
- RMSPROP
- ADAM

Descenso del gradiente: Tradicional vs Estocástico

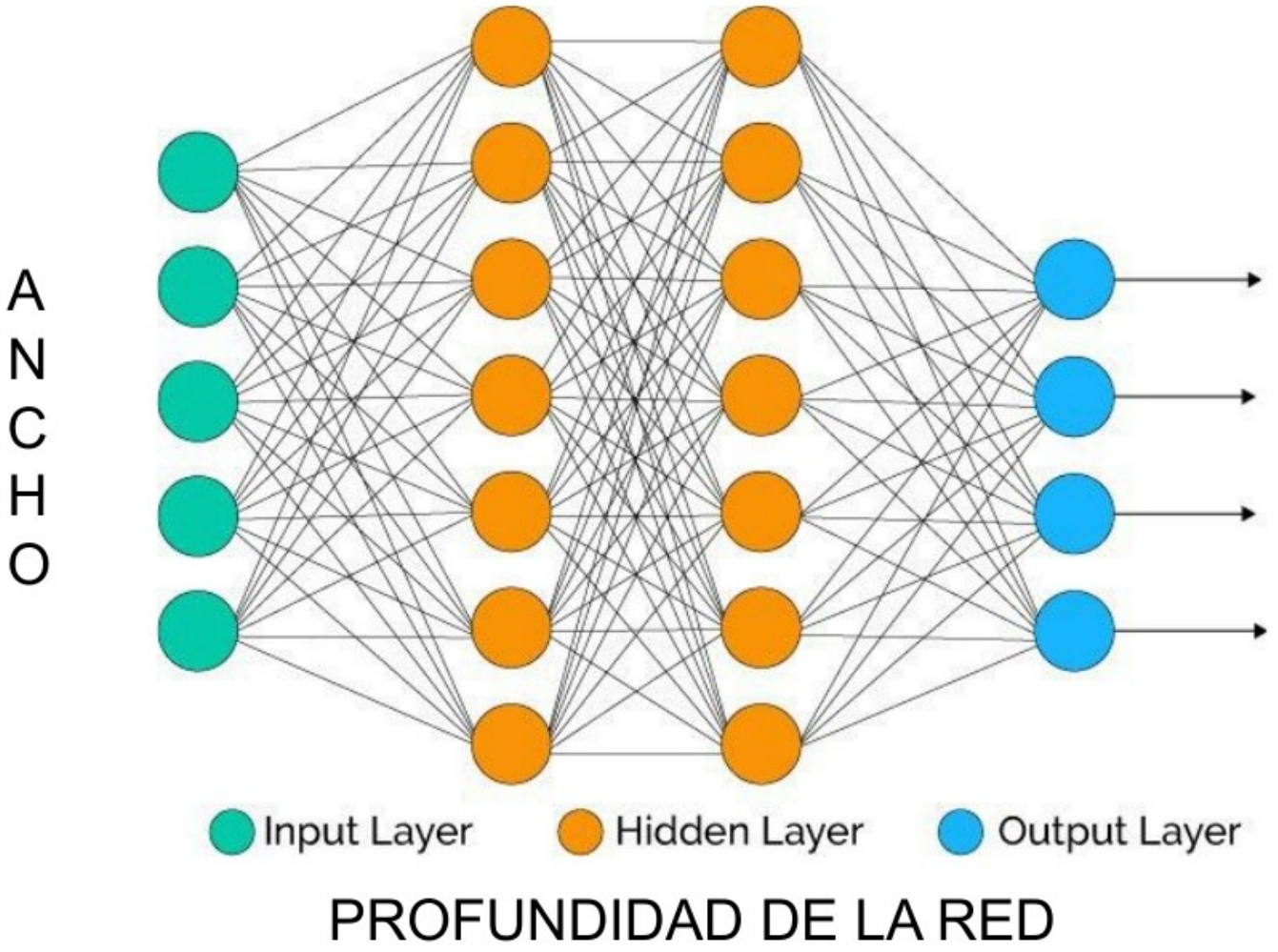


El descenso del gradiente clásico es lento y cae en mínimos locales.
El descenso del gradiente estocástico es más rápido.





Profundidad y ancho de una red



Problemas de las redes: Underfitting y Overfitting

Underfitting

Ocurre cuando **el modelo no aprende lo suficientemente** bien del problema. Tiene un rendimiento deficiente en el conjunto de datos de entrenamiento y no funciona bien en el de test

Overfitting

Ocurre cuando **el modelo aprende demasiado** el conjunto de datos de entrenamiento, con un buen rendimiento para el conjunto de datos de entrenamiento, pero no funciona bien con los datos de prueba.

Pasos para reducirlos

- Agregar más datos
- Usar data-aumentation (aumento de datos)
- Usar arquitecturas que generalicen bien
- Agregar regularización (Dropout, L1/L2, etc.)
- Reducir la complejidad de la arquitectura

BREVE HISTORIA

- 1958 – Perceptron
- 1965 – Multilayer Perceptron
- 1980’s
- Neuronas Sigmoidales
- Redes Feedforward
- Backpropagation
- 1989 – Convolutional neural networks (CNN) / Recurrent neural networks (RNN)
- 1997 – Long short term memory (LSTM)
- 2006 – Deep Belief Networks (DBN): Nace deep learning
- Restricted Boltzmann Machine
- Encoder / Decoder = Auto-encoder
- 2014 – Generative Adversarial Networks (GAN)
- 2017 – Transformers
- Hoy: ChatGPT y el futuro...

USOS

- UN MONTÓN!
- CLASIFICACIÓN BINARIA Y MULTICLASE
- REGRESIÓN
- CLASIFICACIÓN DE IMÁGENES
- DETECCIÓN DE OBJETOS
- PROCESAMIENTO LENGUAJE NATURAL
- PRONÓSTICO SERIES TEMPORALES
- Y UN LARGO ETCÉTERA...

LIBRERÍAS EN PYTHON

- TENEMOS VARIAS:
- SKLEARN: https://scikit-learn.org/stable/modules/neural_networks_supervised.html
- KERAS (TENSORFLOW): https://keras.io/
- PYTORCH: https://pytorch.org/

EJEMPLOS

- EJEMPLO XOR CON KERAS: https://www.aprendemachinelearning.com/una-sencilla-red-neuronal-en-python-con-keras-y-tensorflow/
- EJEMPLO XOR CON NUMPY: https://towardsdatascience.com/implementing-the-xor-gate-using-backpropagation-in-neural-networks-c1f255b4f20d