UNIVERSIDAD SANTO TOMÁS

PRIMER CLAUSTRO UNIVERSITARIO DE COLOMBIA

SECCIONAL TUNJA

VIGILADA MINEDUCACIÓN - SNIES 1732

Acreditación Institucional Internacional

OTORGADA POR EL IAC CINDA ACUERDO 55 DEL 9 DE MAYO–VIGENCIA 5 AÑOS

ACREDITACIÓN INSTITUCIONAL DE ALTA CALIDAD MULTICAMPUS

Res. MEN No. 08456 del 28 de enero de 2016

Vigencia por seis años

QS STARS RATED FOR EXCELLENCE

ISO 9001 icontec SC4289–I

CERTIFIED IQNet MANAGEMENT SYSTEM

UNIVERSIDAD SANTO TOMÁS
PRIMER CLAUSTRO UNIVERSITARIO DE COLOMBIA
SECCIONAL TUNJA
VIGILADA MINEDUCACIÓN - SNIES 1732

Faculty: Systems engineer
Course: Deep Learning
Topic: **Conceptos básico en el Deep learning**

Professor: Luis Fernando Castellanos Guarin
Email: Luis.castellanosg@usantoto.edu.co
Phone: 3214582098

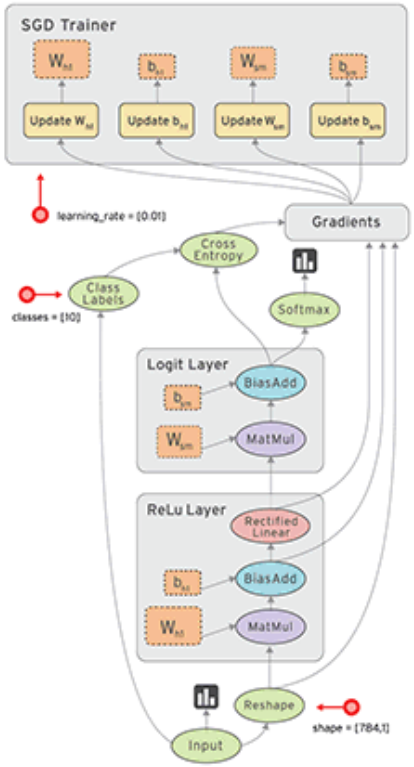
CONTENIDO

Que es?
Historia / timeline
Arquitectura de DL

¡Siempre hacia lo alto!

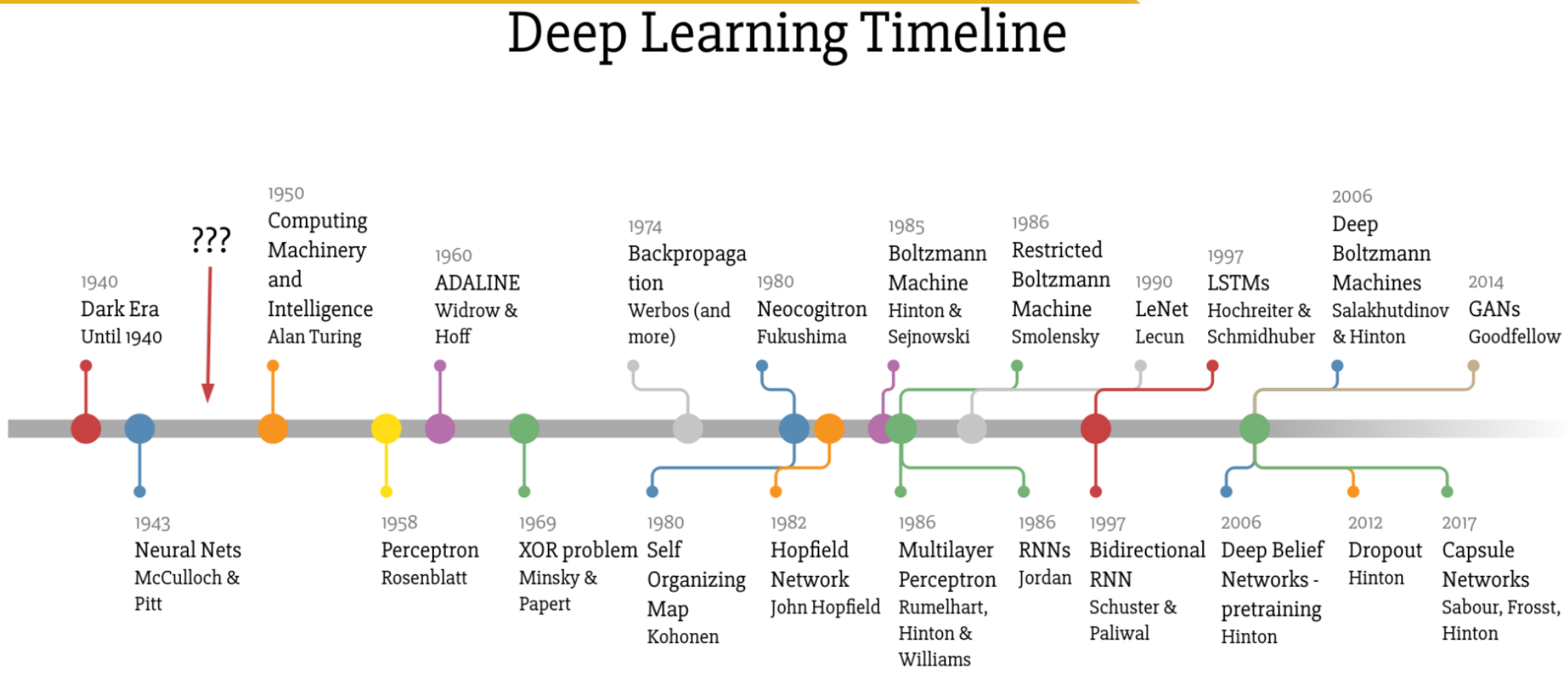
Que es DL

El **Deep Learning/Aprendizaje Profundo,** podemos decir que es una técnica del **Machine Learning** que está basado en una serie de algoritmos, donde su estructura lógica se asimila al funcionamiento del sistema nervioso humano.
Donde en su mayoría se utilizan arquitecturas de redes neuronales que aprenden y extraen de forma automática las características de los datos para producir resultados óptimos



¡Siempre hacia lo alto!

TimeLine DL

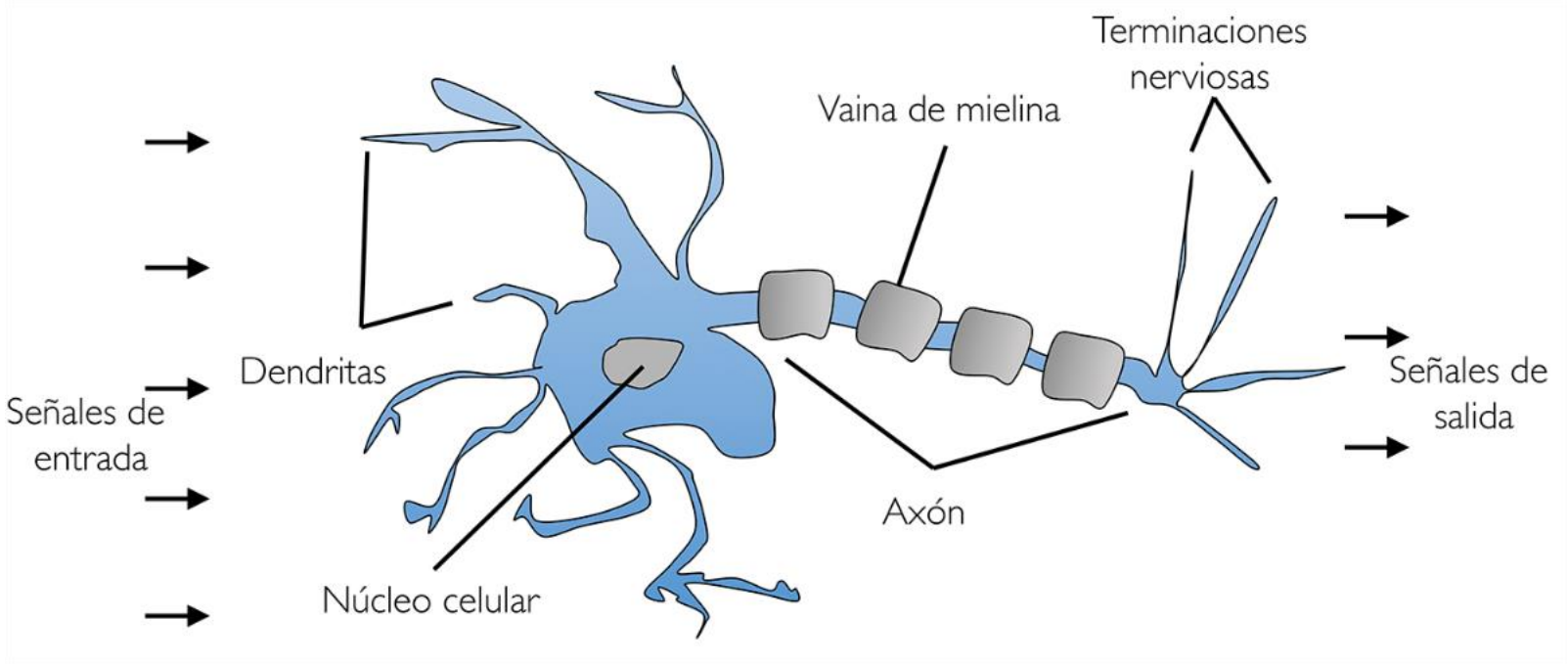


Mejoras teóricas y algorítmicas simples, los avances en hardware (principalmente GPU, ahora TPU) y la generación y acumulación exponencial de datos, hizo que la ultima década el DL sea la nueva y mejorada bandera del Machine Learning (ML)

https://medium.com/datos-y-ciencia/una-introducci%C3%B3n-extra%C3%B1a-al-deep-learning-3407e05e0483

DL teoría sobre redes neuronales

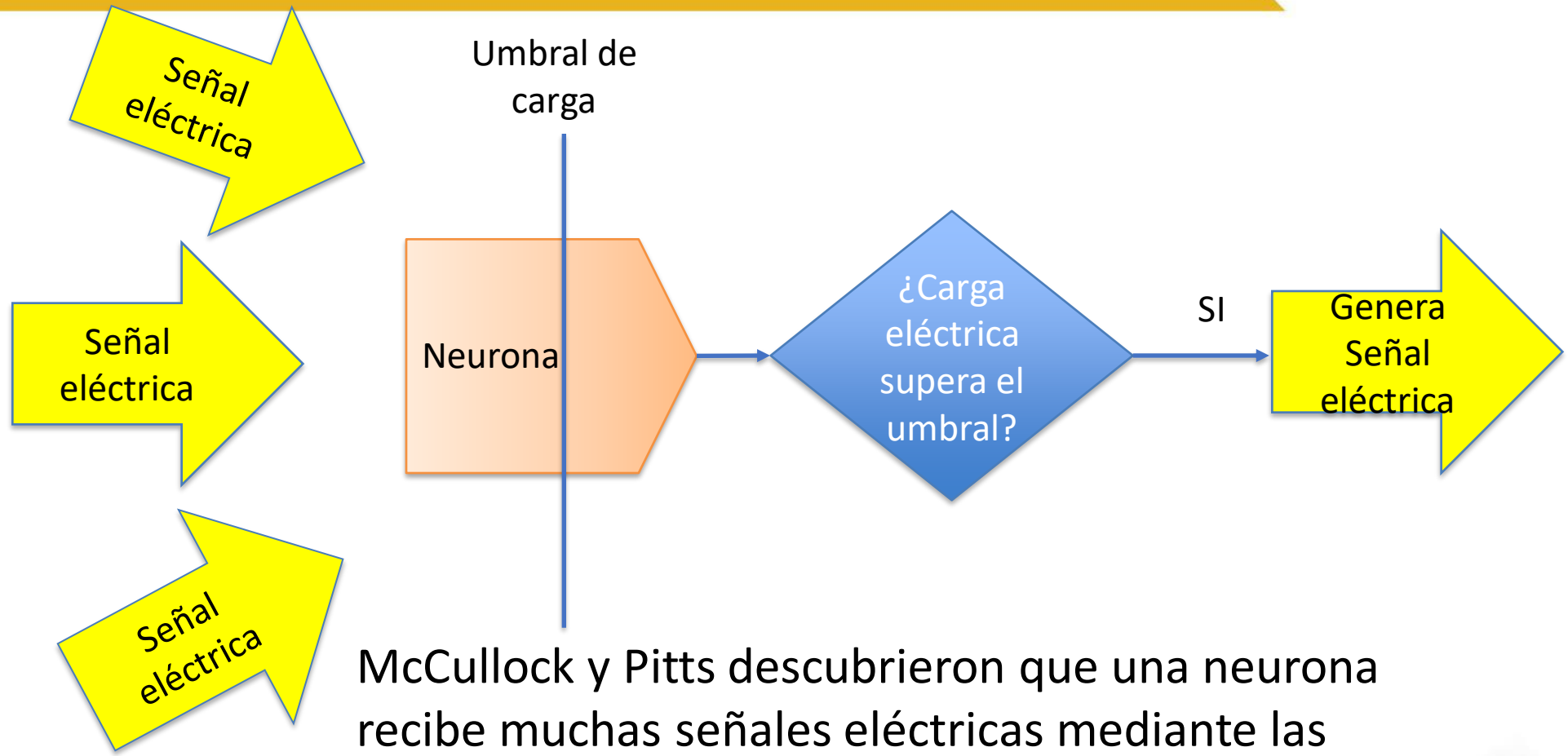
En 1943, Warren McCullock y Walter Pitts publicaron el primer concepto de una célula cerebral.



Las neuronas son células nerviosas interconectadas en el cerebro que participan en el **proceso y la transmisión** de señales eléctricas y química.

http://neuralnetworksanddeeplearning.com/

DL teoría sobre redes neuronales

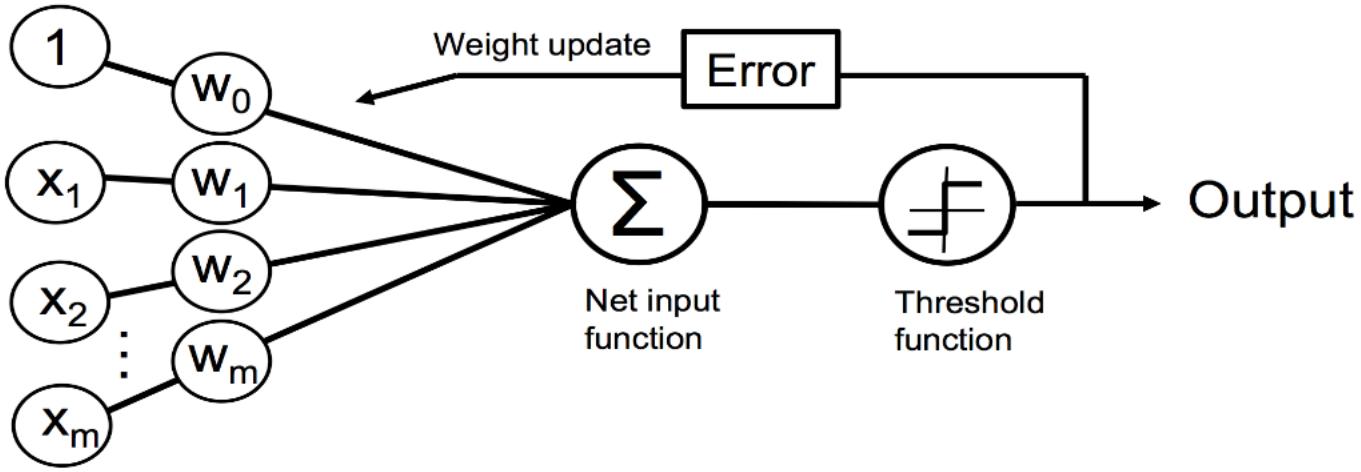


McCullock y Pitts descubrieron que una neurona recibe muchas señales eléctricas mediante las dendritas y si la señal acumulada supera el umbral programado para esa neurona, se generara una señal eléctrica de salida.

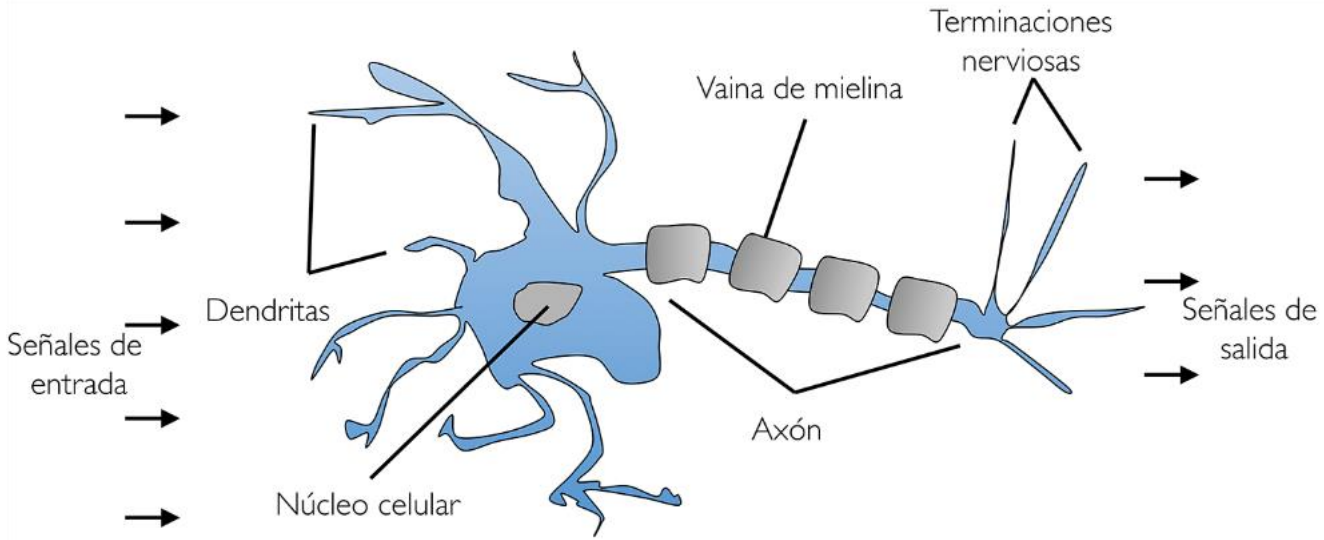
http://neuralnetworksanddeeplearning.com/

DL teoría sobre redes neuronales

Neurona Artificial (software)



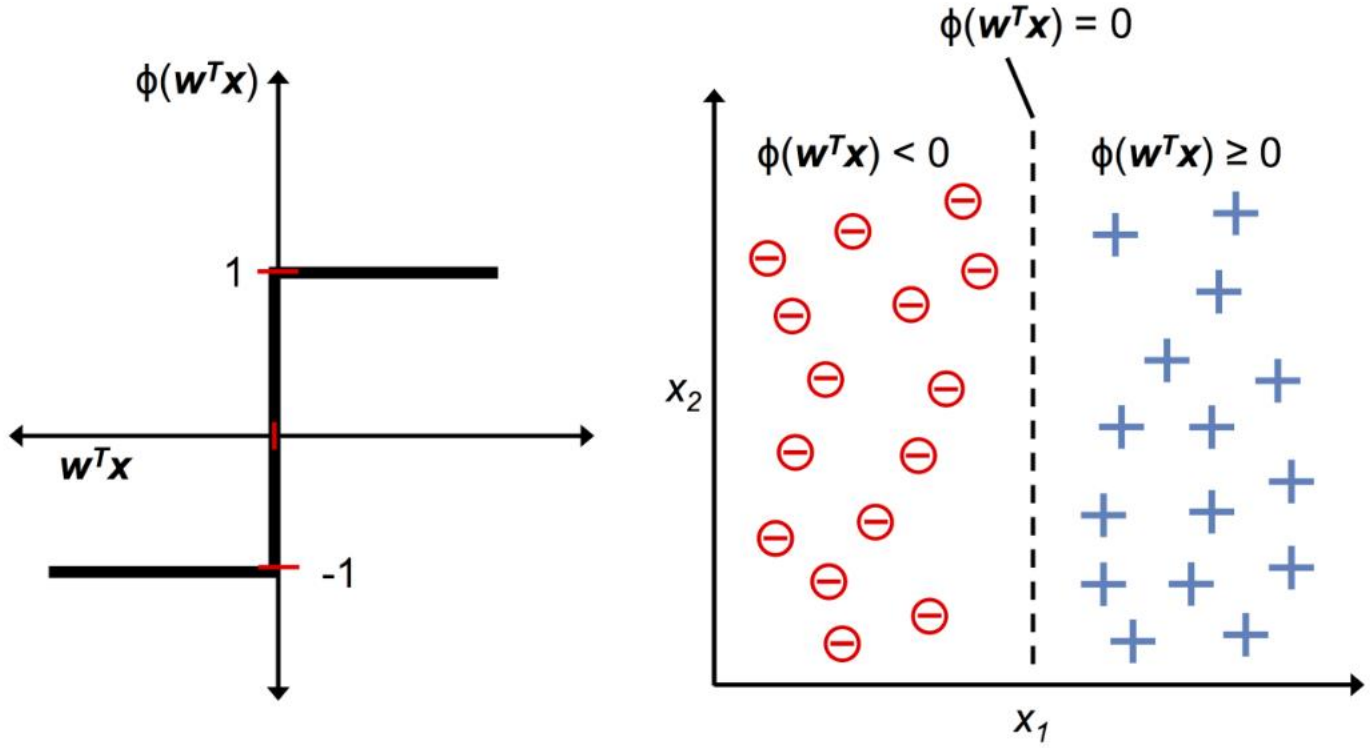
Neurona biológica



http://neuralnetworksanddeeplearning.com/

¡Siempre hacia lo alto!

DL teoría sobre redes neuronales



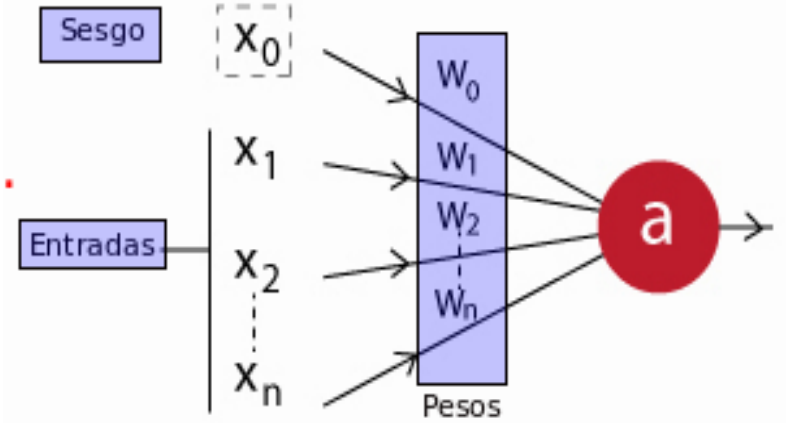
Un algoritmo que recibe una serie de valores negativos y positivos y si la sumatoria de sus valores pueden "excitar" o "no"(En caso que lo haga generara un valor a otro algoritmo).

http://neuralnetworksanddeeplearning.com/

DL teoría sobre redes neuronales

Los components de una Neurona:

- **x1,x2,…,xn:** Los datos de entrada en la neurona, los cuales también puede ser que sean producto de la salida de otra neurona de la red.



- **x0**: La unidad de sesgo; un valor constante que se le suma a la entrada de la función de activación de la neurona. Generalmente tiene el valor 1. Este valor va a permitir cambiar la función de activación hacia la derecha o izquierda, otorgándole más flexibilidad para aprender a la neurona.

- **w0,w1,w2,…,wnw0,w1,w2,…,wn:** Los pesos relativos de cada entrada. Tener en cuenta que incluso la unidad de sesgo tiene un peso.

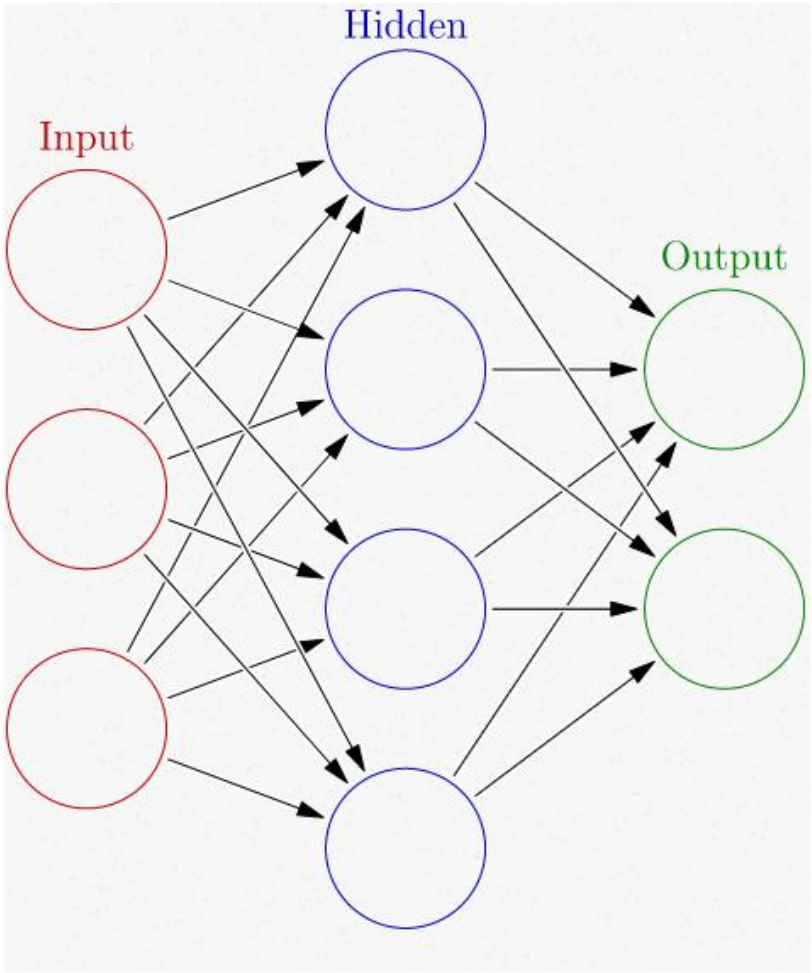
- **a:** La salida de la neurona. Que va a ser calculada de la siguiente forma:

$$a = f\left(\sum_{i=0}^{n} w_i \cdot x_i\right)$$

Aquí **f** es la ***función de activación*** de la neurona. Esta función es la que le otorga tanta flexibilidad a las redes neuronales y le permite estimar complejas relaciones no lineales en los datos. Las más communes son sigmoide, tangente hiperbólica y rectificadora (ReLU)

http://neuralnetworksanddeeplearning.com/

DL teoría sobre redes neuronales



Cada **unidad neuronal** está conectada con muchas otras y los enlaces entre ellas pueden incrementar o inhibir el estado de activación de las neuronas adyacentes. Estos sistemas aprenden y se forman a sí mismos, en lugar de ser programados de forma explícita, y sobresalen en áreas donde la detección de soluciones o características es difícil de expresar con la programación convencional.

http://neuralnetworksanddeeplearning.com/

DL teoría sobre redes neuronales

La **red** aprende algo simple en la capa inicial de la jerarquía y luego envía esta información a la siguiente capa. La siguiente capa toma esta información, **lo combina** en algo que es un poco más complejo, y **lo pasa** a la tercer capa. Este proceso continúa de forma tal que cada capa de la jerarquía construye algo más complejo de la entrada que recibió de la capa anterior. De esta forma, la red irá aprendiendo por medio de la exposición a los datos de ejemplo.

Para que la red **aprenda** debemos encontrar los pesos de todas las capas de forma tal que la red realice un mapeo perfecto entre los ejemplos de entrada con sus respectivas salidas objetivo.

http://neuralnetworksanddeeplearning.com/

DL teoría sobre redes neuronales

"***Para poder controlar algo, en primer lugar debemos poder observarlo***".

En este sentido, para controlar la salida de la red neuronal, es necesario poder medir cuan lejos esta la salida que se obtiene de la que se esperaba obtener. Este es el trabajo de la **función de coste/perdida de la red**. Existen varios algoritmos:

- **Raíz cuadrada media – RMSE:** Se entiende como residuos la diferencia entre el valor previsto (correcto) y el valor real obtenido
- **Error absoluto medio – MAE:** se calcula como la **suma media de los valores absolutos** de los errores.
- **Error absoluto medio escalado – MASE: similar al MAE** pero escalado.
- **Entropía cruzada categórica – Categorical Cross-Entropy:** es una medida de precisión para **variables categóricas**.
- **Entropía cruzada binaria – Binary Cross-Entropy:** es una medida de precisión para **variables binarias**.

DL teoría sobre redes neuronales

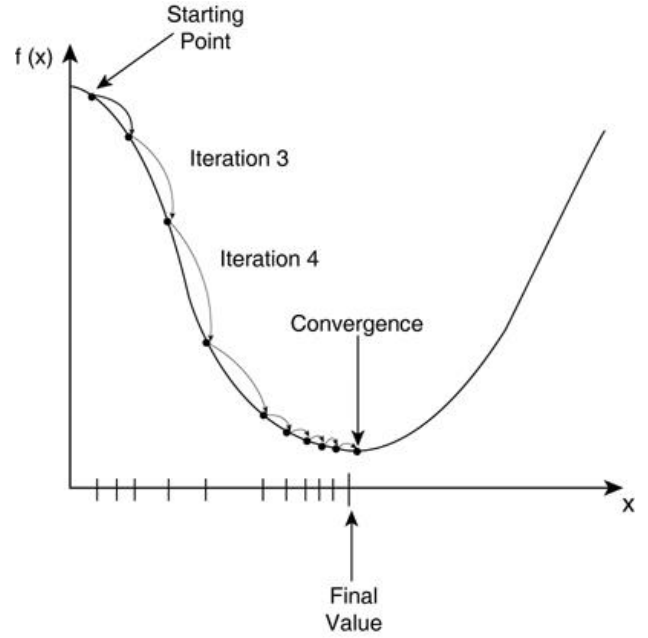
función de coste, debe ser elegida en base al tipo de problema concreto, para poder evaluar de la forma más adecuada la diferencia entre las predicciones de nuestra red y las salidas reales.

Problem Type	Output Type	Final Activation Function	Loss Function
Regression	Numerical value	Linear	Mean Squared Error (MSE)
Classification	Binary outcome	Sigmoid	Binary Cross Entropy
Classification	Single label, multiple classes	Softmax	Cross Entropy
Classification	Multiple labels, multiple classes	Sigmoid	Binary Cross Entropy

DL teoría sobre redes neuronales

Y como todo se puede optimizar, obviamente tenemos los **optimizadores de la función de coste.**

- **Descenso estocástico del gradiente – SGD (Stochastic Gradient Descent):** es una aproximación estocástica del gradiente descendiente usado para minimizar una función objetivo que se escribe como una suma de funciones diferenciables



- **Adam:** Busca solventar el problema con la fijación de el ratio de aprendizaje del SGD, Adam es una actualización del Optimizador RMSProp (Root Mean Square Propagation) que se basa en un ratio de aprendizaje adaptativa.

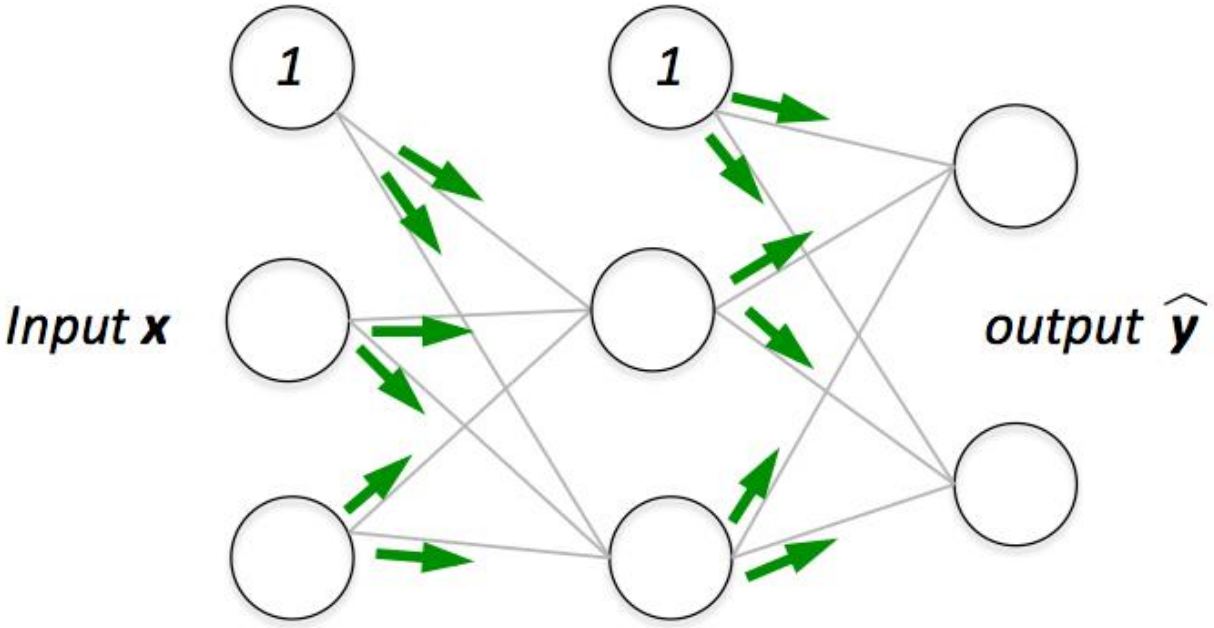
- **Adagrad:** Realiza grandes actualizaciones cuando los parámetros son poco frecuentes y pequeñas actualizaciones cuando son muy frecuentes.

- **Adadelta:** El optimizador Adadelta es una extensión del algoritmo Adagrad.

DL teoría sobre redes neuronales

Propagación hacia atrás

La propagación hacia atrás o **backpropagation** es un algoritmo que funciona mediante la determinación de **la pérdida (o error)** en la salida y luego propagándolo de nuevo hacia atrás en la red. De esta forma los pesos se van actualizando para minimizar el error resultante de cada neurona. Este algoritmo es lo que les permite a las redes neuronales aprender.



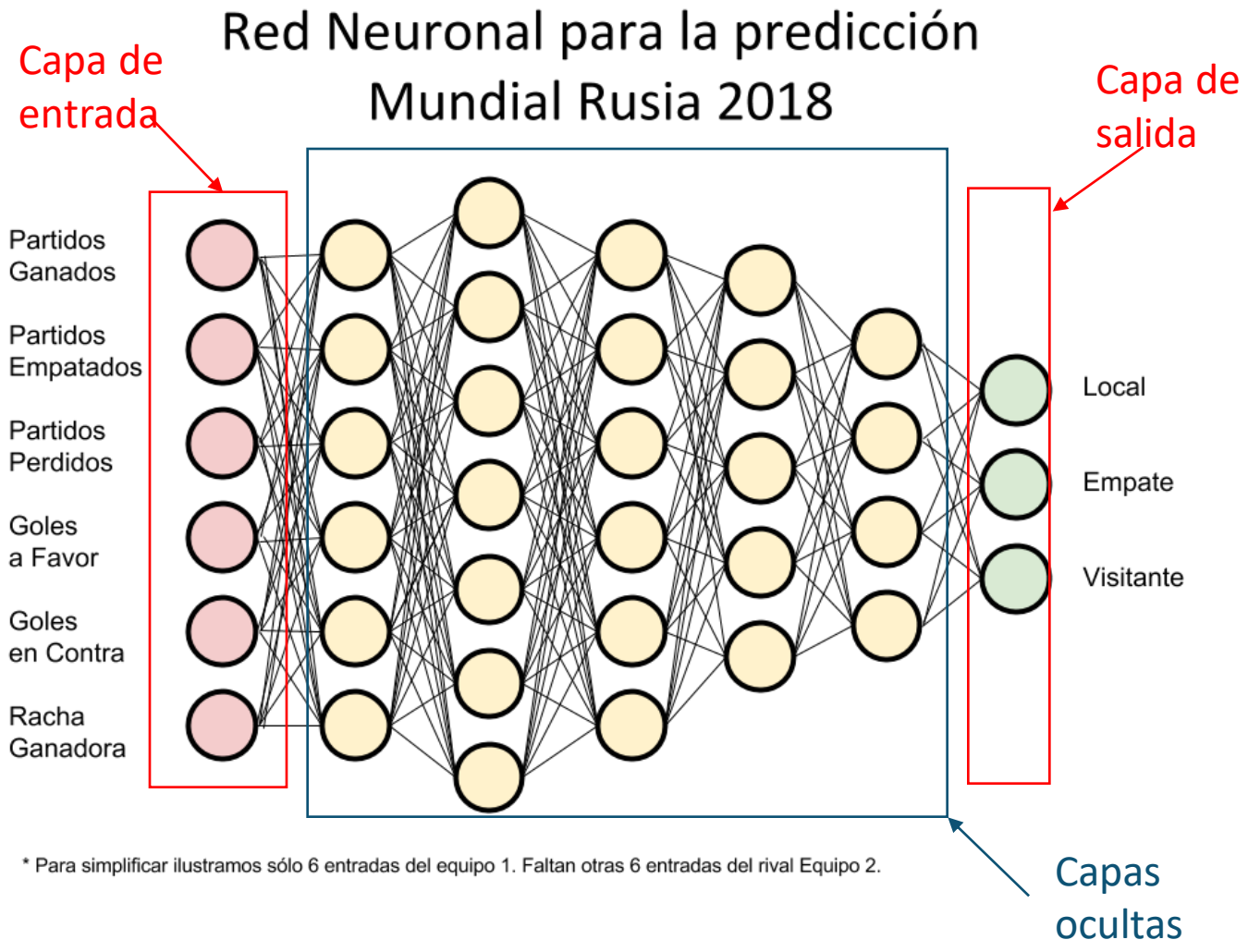
DL teoría sobre redes neuronales

Resumiendo, Si partimos de cero (sin un modelo pre-entrenado) es bastante común proceder de la siguiente forma

1. Definimos la capa 0 (con los parámetros de entrada) y la capa final con los resultados.
2. Creamos las capas ocultas y les asignamos pesos de forma aleatoria, por lo que la red simplemente implementa una serie de transformaciones aleatorias.
3. forward pass (hacia adelante), pasamos el vector de entrada y lo pasamos capa por capa
4. Calculamos la función de coste/error, obviamente la salida del modelo dista bastante del ideal que deseamos obtener, por lo que el valor de la función de pérdida va a ser bastante alto.
5. Aplicamos el algoritmo de **backpropagation**, donde replicamos el error hacia atrás con derivadas de las funciones de activación.
6. Volvemos a aplicar forward pass (hacia adelante), pero con los pesos ajustados de forma tal de ir reduciendo cada vez más el valor de la función de pérdida. Este proceso es el que se conoce como entrenamiento de la red, el cual repetido una suficiente cantidad de veces, generalmente 1000 iteraciones de miles de ejemplos, logra que los pesos se ajusten a los que minimizan la función de pérdida.

Una red que ha minimizado la pérdida es la que logra los resultados que mejor se ajustan a las salidas objetivo, es decir, que el modelo se encuentra entrenado.

Ejemplo básico de una red neuronal



* Para simplificar ilustramos sólo 6 entradas del equipo 1. Faltan otras 6 entradas del rival Equipo 2.

- **La capa de entrada** recibe los datos de entrada y los pasa a la primer capa oculta.
- **Las capas ocultas** realizarán cálculos matemáticos con nuestras entradas.
- **La capa de Salida** devuelve la predicción realizada. En nuestro caso de 3 resultados discretos las salidas podrán ser “1 0 0” para Local, “0 1 0” para Empate y “0 0 1” para Visitante.

Arquitectura de DL

Dependiendo del tipo de entradas con las que trabajemos, van a existir distintas arquitecturas de redes neuronales que mejor se adaptan para procesar esa información. Algunas de las arquitecturas más populares son:

- Redes neuronales prealimentadas (ANN) (básicas, solo hacia Adelante)
- **Redes neuronales convolucionales CNN** *(visión por computadora)*
- Redes neuronales recurrentes (RNN) y redes de memoria de largo plazo a corto plazo (LSTM) *(Procesamiento de lenguaje natural PLN)*
- Redes Generativas Antagónicas/Generative Adversarial Networks (GAN)
- Redes cápsula (capsNet)

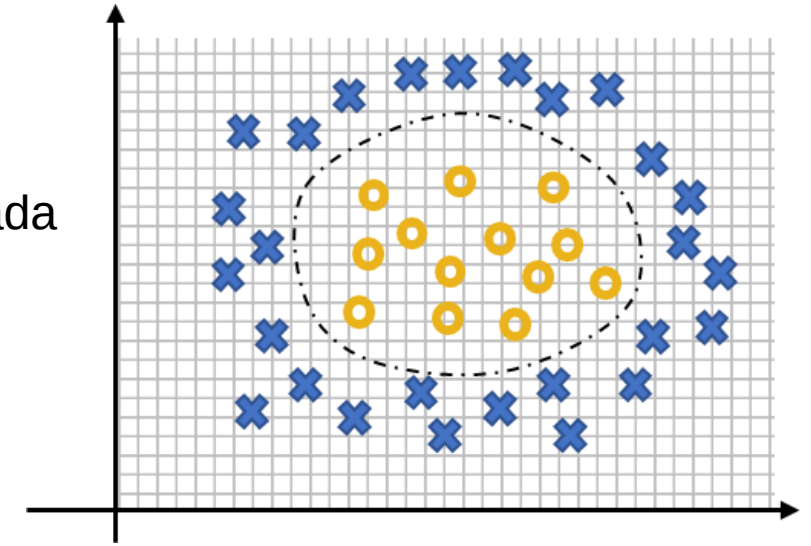
https://relopezbriega.github.io/blog/2017/06/13/introduccion-al-deep-learning/

Como entrenar una red neuronal

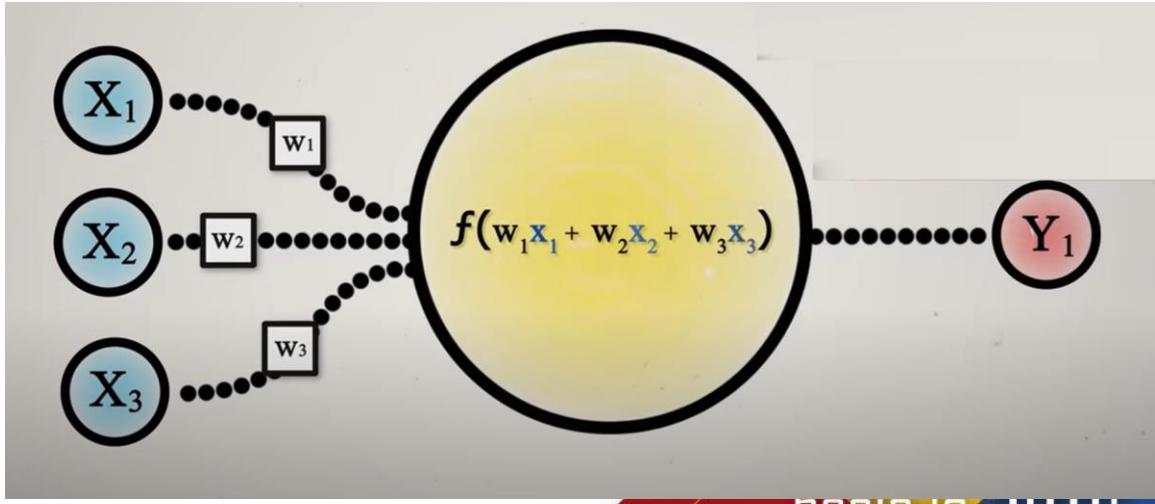
Necesitaremos tres(3) cosas:

1. Gran cantidad de valores en nuestro conjunto de Datos de Entrada (donde ningún método clásico de ML "*sea idóneo*")

2. Gran poder de cálculo computacional (GPU/TPU)



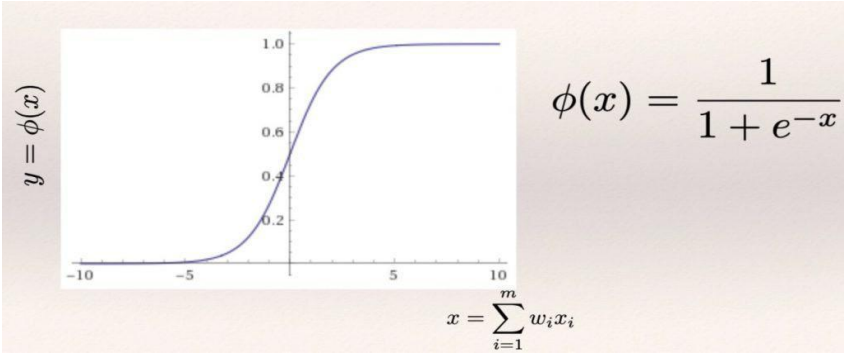
3. Implementar varias funciones:
   - **Función Coste**
   - **Función sigmoide**
   - **Función rectificadora (ReLU)**
   - **Función tangente hiperbólica**



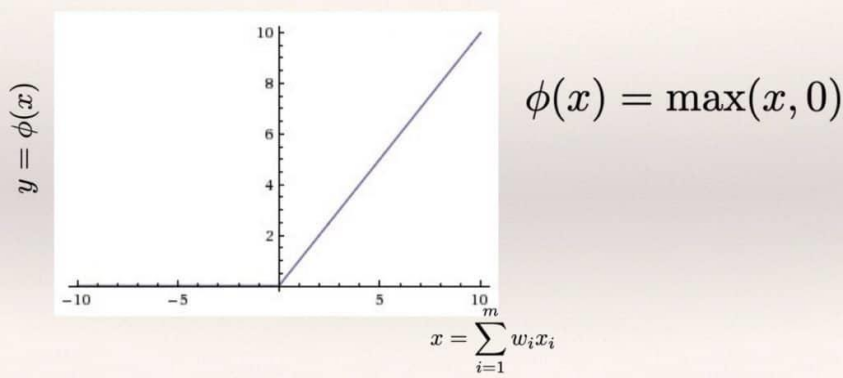
Funciones de una red neuronal

1. "***Función Coste***". **Idealmente queremos que nuestro coste sea cero**, es decir sin error(cuando el valor de la predicción es igual al resultado real del partido). **A medida que entrena el modelo irá ajustando los pesos de inter-conexiones de las neuronas** de manera automática hasta obtener buenas predicciones. A ese proceso de "ir y venir" por las capas de neuronas se le conoce como Back-Propagation.

2. "***Función sigmoide***", Se usa en la regresión logística y en DL se usa para clasificar con valores categóricos y para intentar predecir las probabilidades de pertenencia a cada categoría, donde sabemos que la probabilidad de un suceso imposible es 0 y la de un suceso seguro es 1



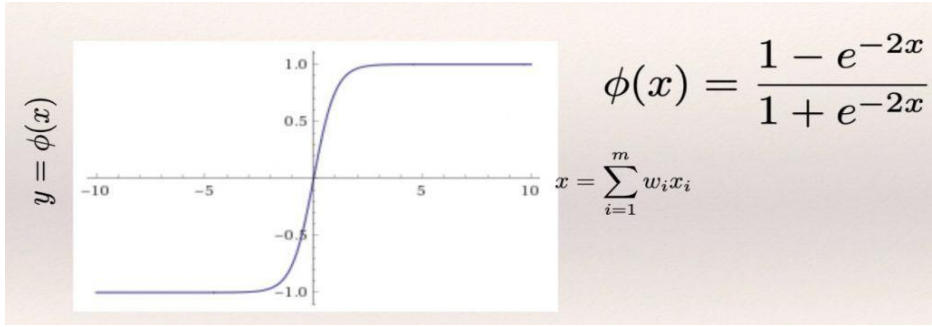
3. **Función rectificadora (ReLU):** se obvian todas aquellas entradas que una vez ponderadas tienen un valor negativo y nos quedamos con el valor exacto de las positivas



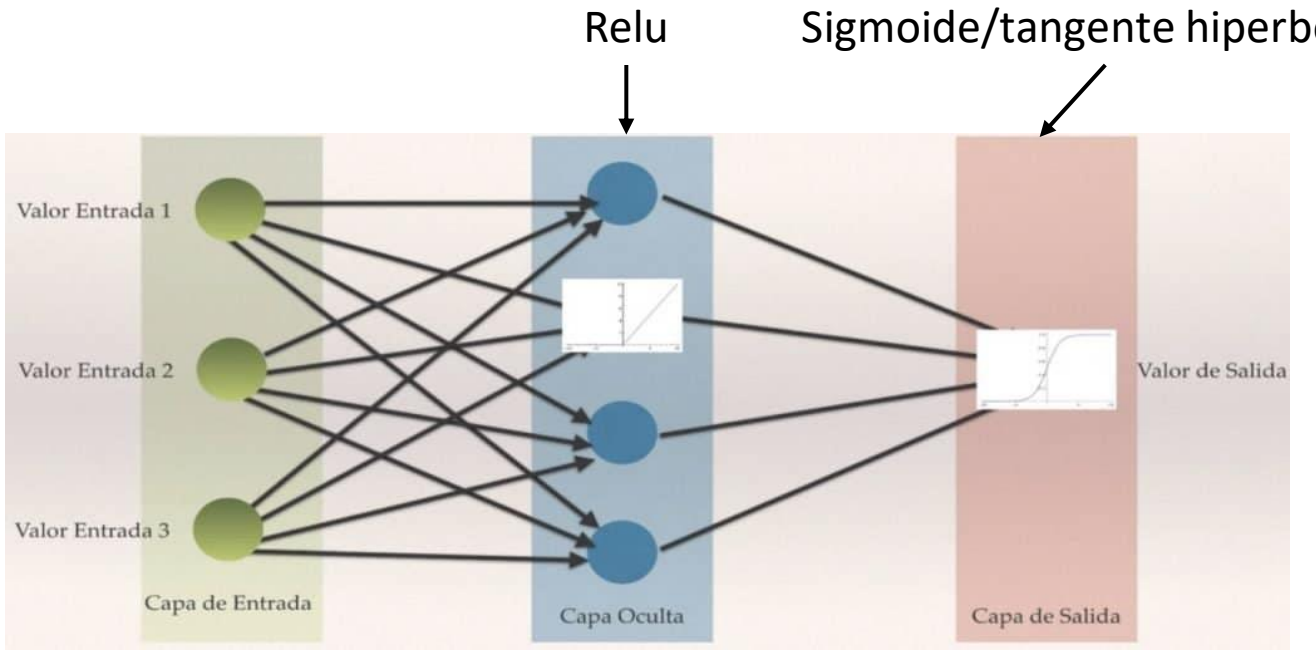
Funciones de una red neuronal

4. Función tangente hiperbólica:

Esta función es muy parecida a la función sigmoidea, pero en este caso, como vemos en la figura de abajo, no empieza en 0 y termina en 1, sino que empieza en -1 y termina en 1.



Donde se usan esas funciones?



¡Siempre hacia lo alto!

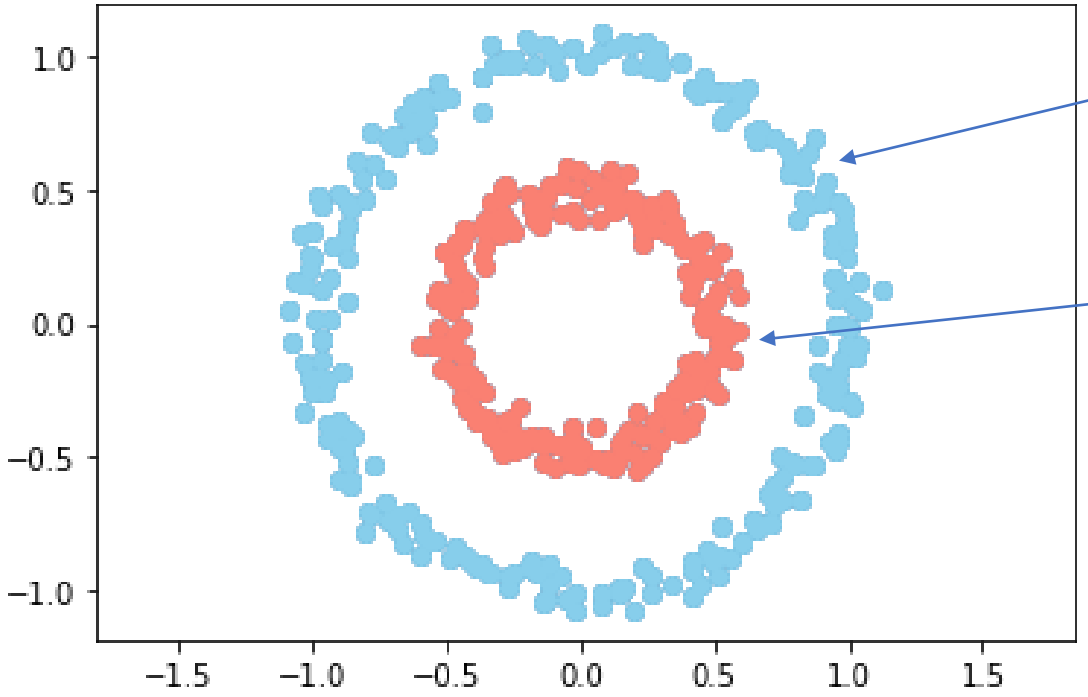
Tensorflow / torch / theano / caffe /keras

Biblioteca	Creador/soporte	Descripción	Plataforma
Torch/pytorch	Facebook	PyTorch trabaja con grafos dinámicos en vez de estáticos	Lua/python
theano	Université de Montréal	definir, optimizar y evaluar expresiones matemáticas que implican cálculos con arrays multidimensionales de forma eficiente	Python
Caffe	UC Berkeley Facebook	usado en visión por computador o clasificación de imagenes	C++
Keras	**Google brian**	API de alto nivel para el desarrollo de redes neuronales	Python
Tensorflow	**Google brian**	proyecto de código abierto de Google, diseñado para computación numérica mediante grafos	C++,Python, js…
CNTK	Microsoft	Altamente funcional en el área del procesamiento de lenguaje natural (reconocimiento de voz)	C++, Python
MindSpore	Huawei	Supuesta competencia a Tensorflow	Python
SageMaker	Amazon Aws	cloud	
Azure ML	Microsoft	cloud	R o Python
Watson ML	IBM	cloud	
Google cloud	Google	cloud	Python

Todas tienes soporte en CUDA, ejecución en paralelo, permiten trabajar en redes CNN, RNN, GAN

Que algoritmo usar?

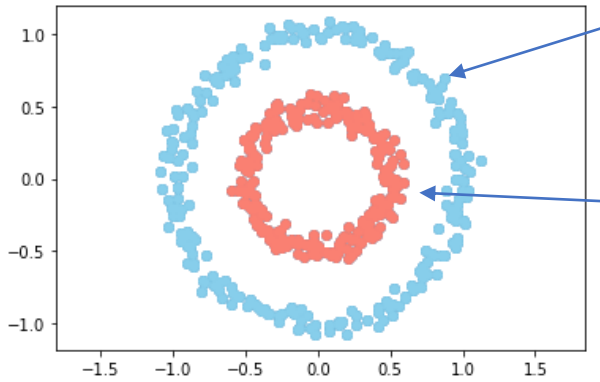
Si tenemos el siguiente dataset, que algoritmo de machine learning recomendaríamos?



Creando una red neuronal básica.

Creando una red Neuronal Artificial sencilla con 4 capas:

- Capa 0: n parámetros de entrada.



Grupo 1 de datos

Grupo 2 de datos

- **Capa ocultas:**
  - **1, con 4 neuronas**
  - **2, con 8 neuronas**

- Capa de salida con una neurona, Salida binario [0,1] (grupo1, grupo2)

¡Siempre hacia lo alto!

Bibliográfia

https://medium.com/datos-y-ciencia/una-introducci%C3%B3n-extra%C3%B1a-al-deep-learning-3407e05e0483

https://www.aprendemachinelearning.com/como-funcionan-las-convolutional-neural-networks-vision-por-ordenador/

https://developers.google.com/machine-learning/crash-course/introduction-to-neural-networks/programming-exercise?hl=es-419

UNIVERSIDAD SANTO TOMÁS

PRIMER CLAUSTRO UNIVERSITARIO DE COLOMBIA

SECCIONAL TUNJA

VIGILADA MINEDUCACIÓN - SNIES 1732

¡Siempre hacia lo alto!

USTATUNJA.EDU.CO

@santotomastunja