Introduction to Artificial Neural Networks (ANN)

- Una gran parte de la sección será basada casi solo en teoría,
- Luego las ideas serán implementadas en código.

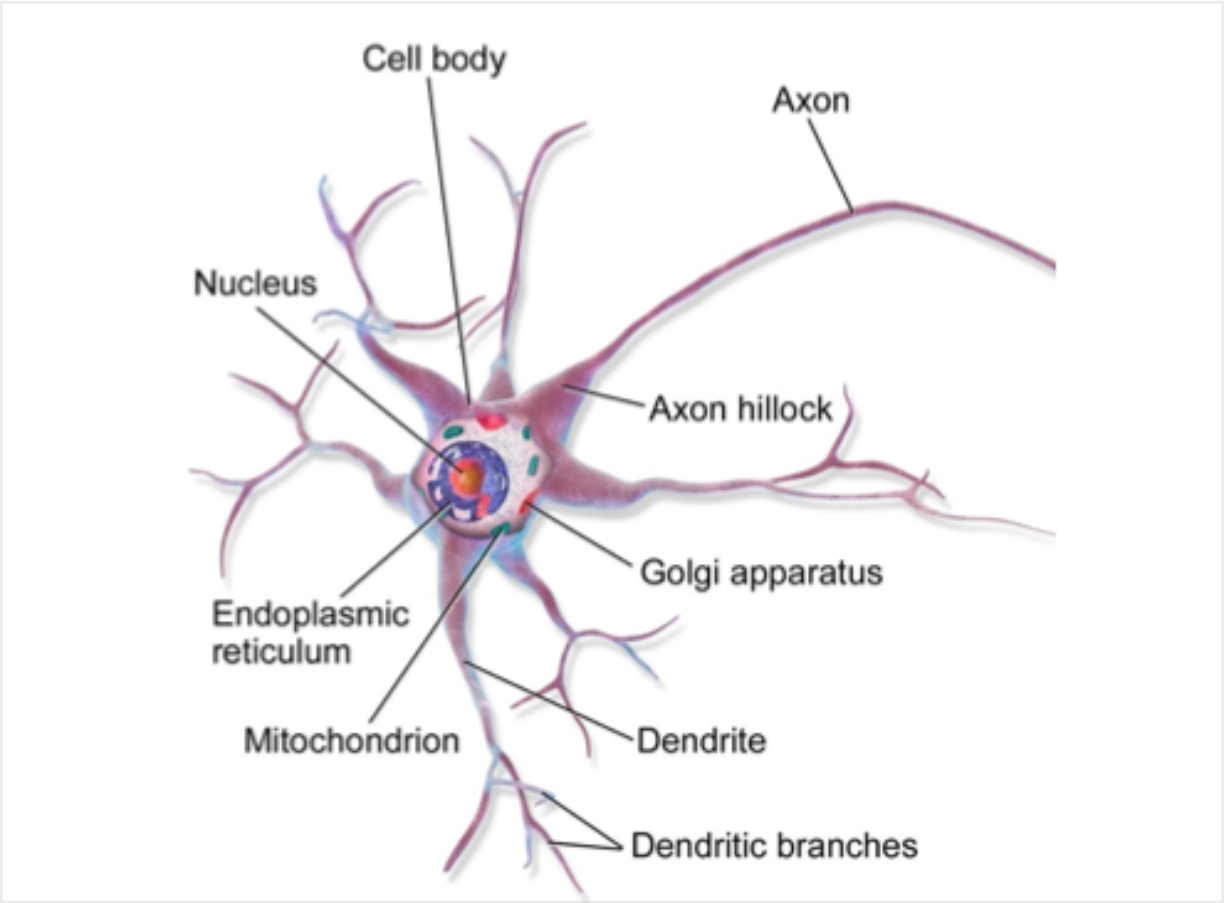
Tópicos:
- Modelo de perceptrón a redes neuronales,
- Funciones de activación,
- Functiones de costo,
- Feed Forward networkds,
- BackPropagation.

Códing topics:
- Tensorflow 2.0 Keras Syntax,
- ANN w/ Keras:
  - Regression,
  - Classification.
- Ejercicios para Keras ANN,
- Tensorboard Visualizations.

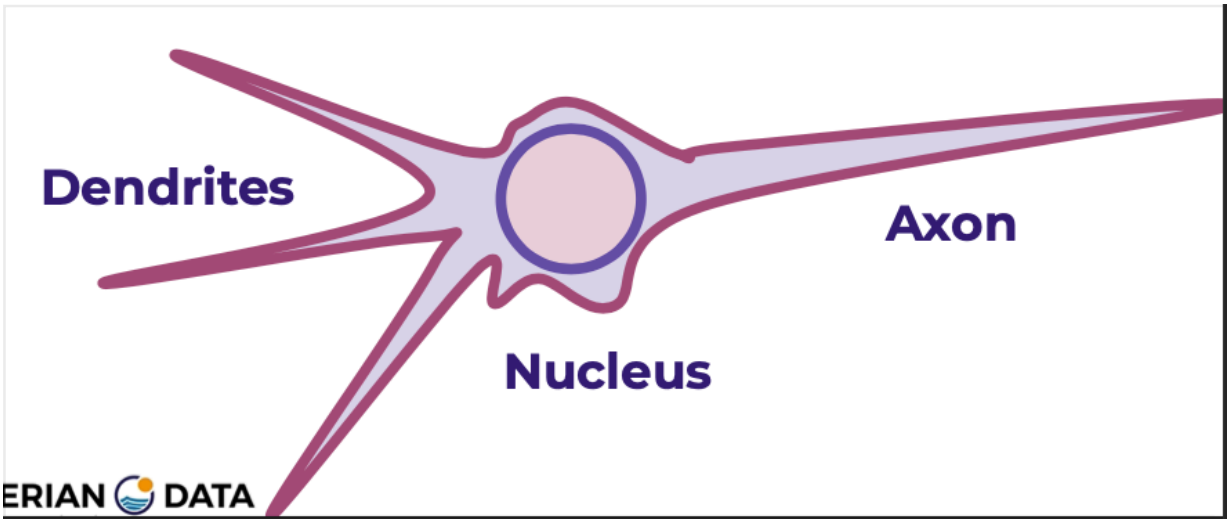
Módelo Perceptrón

para entender DL, hay que entender lo siguiente primero para construir la intuición:
- Neurona biológica,
- Perceptrón,
- Multi-Layer modelo perceptrón,
- Red neuronal de DL.

Ilustración de neurona:



Una versión simplificada para entender el funcionamiento macro se puede ver como:



Las dendritas pueden pensarse como inputs que van al nucleo y el axon se puede entender como un output/salida. El nucleo hace algún tipo de “cálculo” y da una única salida.

En 1958 Fran Rosenblatt creó el modelo matematico qu representa esto. En ese

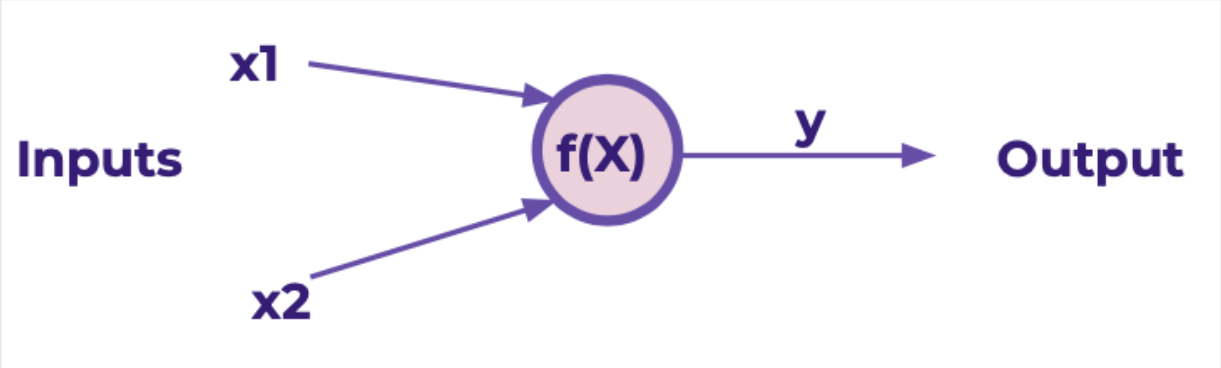
entonces se le vio gran potencial, asegurnado que “El perceptrón podrá eventualmente ser capaz de aprender, tomar decisiones y traducir lenguas”.

Sin embargo, casi 2 décadas después, en 1969, Marvin Minsky y Seymor Papert publicaron el libro “Perceptron”, en donde mataron la idea de un modelo matemático útil del perceptrón. Este libro marcó el inicio del “invierno de la IA”, que se vio representado como una época con muy poca inversión en proyectos de IA.

De todas formas, hoy en día conocemos lo poderosas que son las redes neuronales, las cuales partieron del modelo simple del perceptrón, así que se continuará estudiando el modelo biológico para luego llevarlo a código.

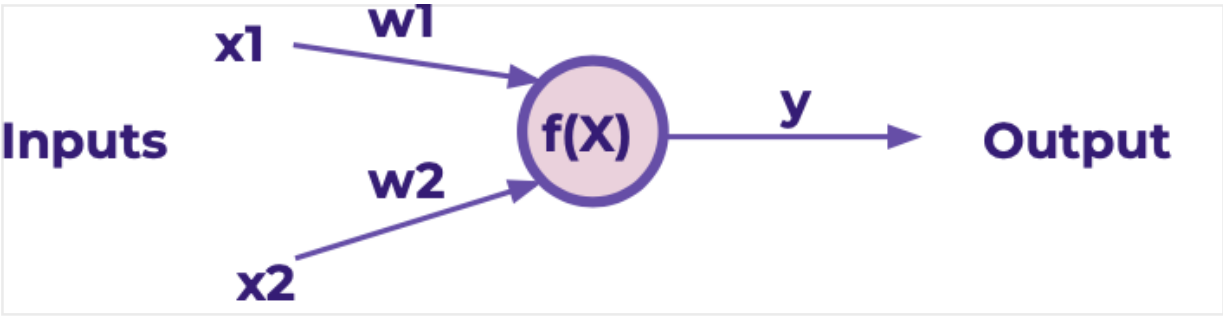
1. Representación de las unidades biológicas por matemáticas:

Se les asigna a las entradas respectivamente x1 & x2,
Existe una función dentro del nucleo que transforma las entradas,
termina en una salida única llamada ´y´.



Si definieramos f(X) como una suma, entonces tendríamos que:
y = x1+x2.

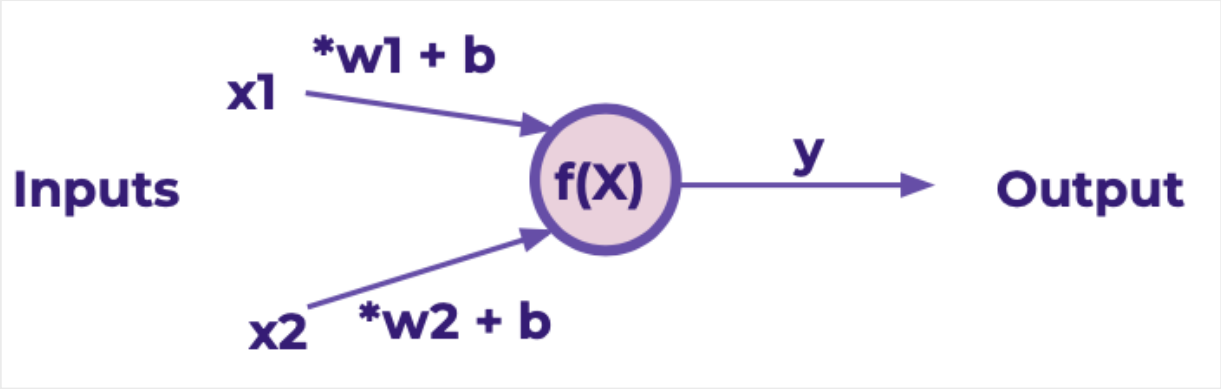
Realistamente, nosotros querríamos ser capaces de ajustar algún parámetro con el fin de “aprender”, para ello se agregan pesos (weights /w) con el fin de ponderarlos a cada entrada.



ahora:

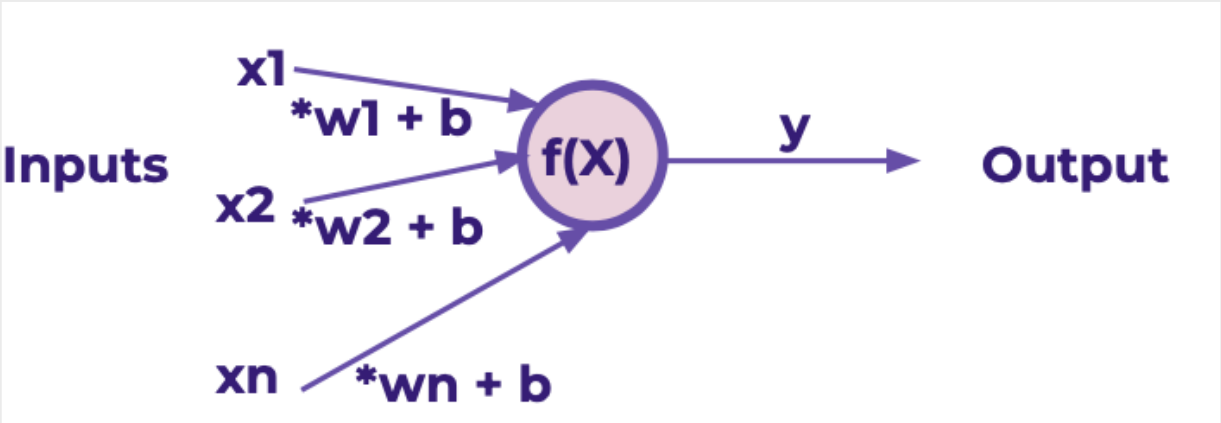
y = x1w1 + x2w2

Pero, ¿qué pasa si los input son 0?, el peso no cambiaria nada.
Es para lo anterior que se ocupa un sesgo /bias, como un escalar que se suma a los pesos,



quedando
y = (x1w1+b) + (x2w2+b)

y finalmente se generaliza a:



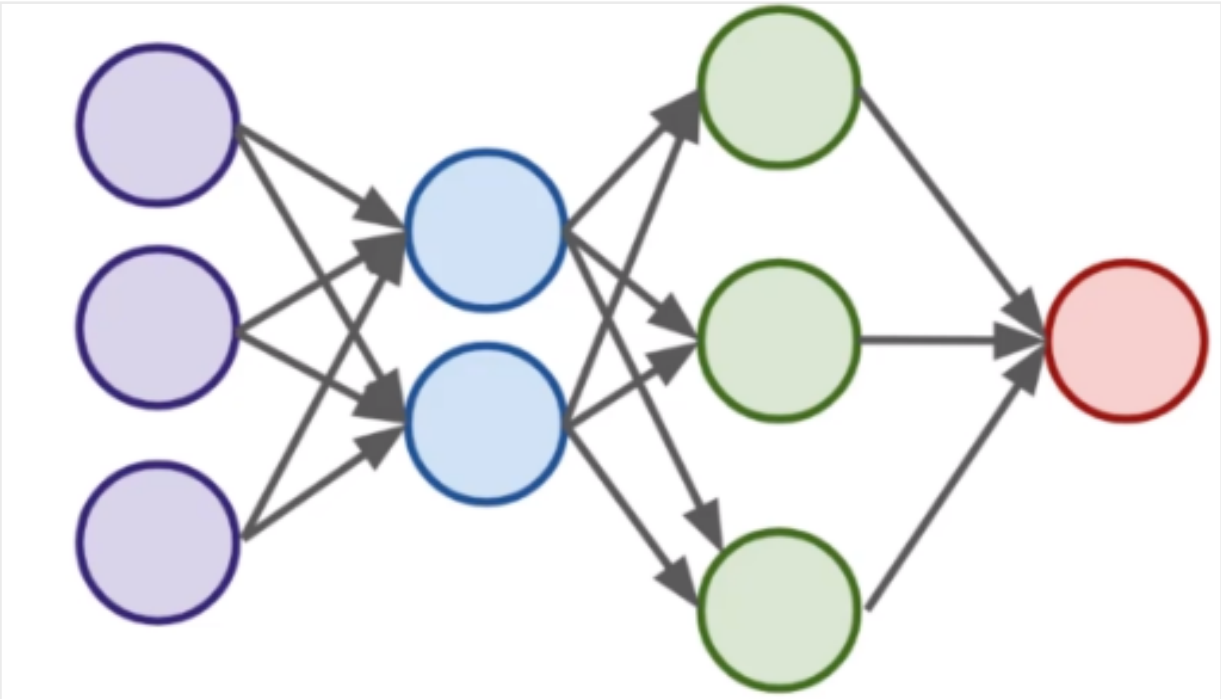
Expresado matematicamente tenemos:

$$\hat{y} = \sum_{i=1}^{n} x_i w_i + b_i$$

Redes Neuronales / Neural Networks

Aprender el funcionamiento de un simple perceptron no será suficiente para aprender sistemas complicados,
Pero podemos expandir la idea de un perceptrón simple para crear un modelo de perceptrón multi capa, también llamado como una red neuronal básica.

¿Cómo construimos una red neuronal de la idea de un perceptrón?

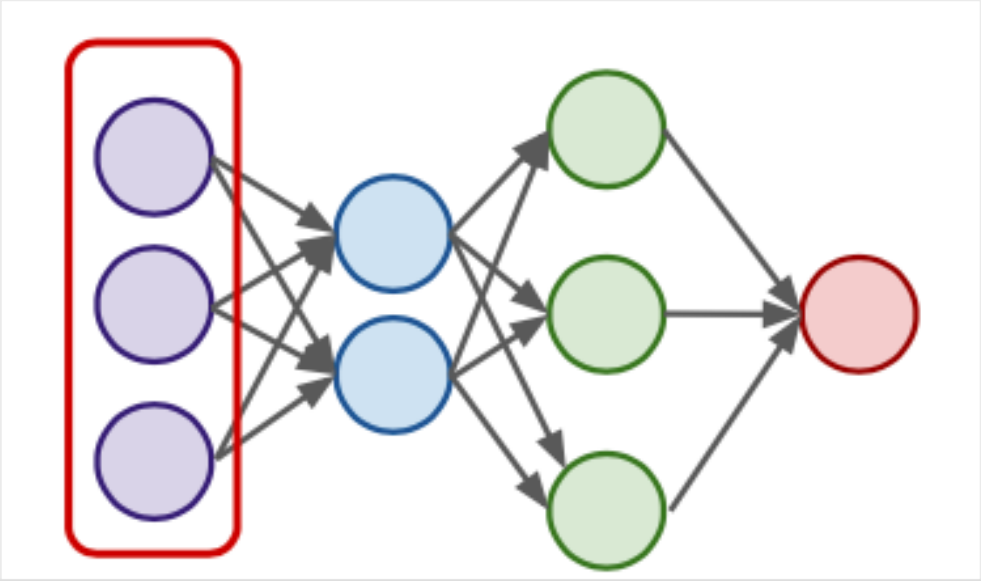


Generamos verticalmente redes de perceptrones singulares y conectamos sus salidas como entradas a la siguiente capa, así queda como la salida de una capa es la entrada de la siguiente.

Por ahora nos enfocaremos en el modelo “Feed forward” que significa que todo el output del primer layer va hacia adelante en el modelo hasta llegar a la salida, esto permite a la red comportarse como un “todo” y aprender relaciones e interacciones entre las variables de entrada /features.

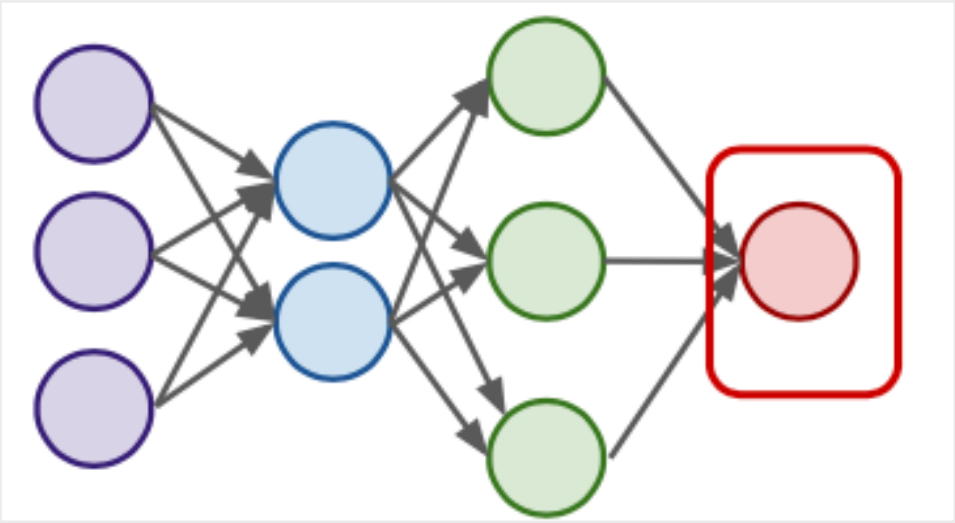
Tipos de capas

Capa de entrada



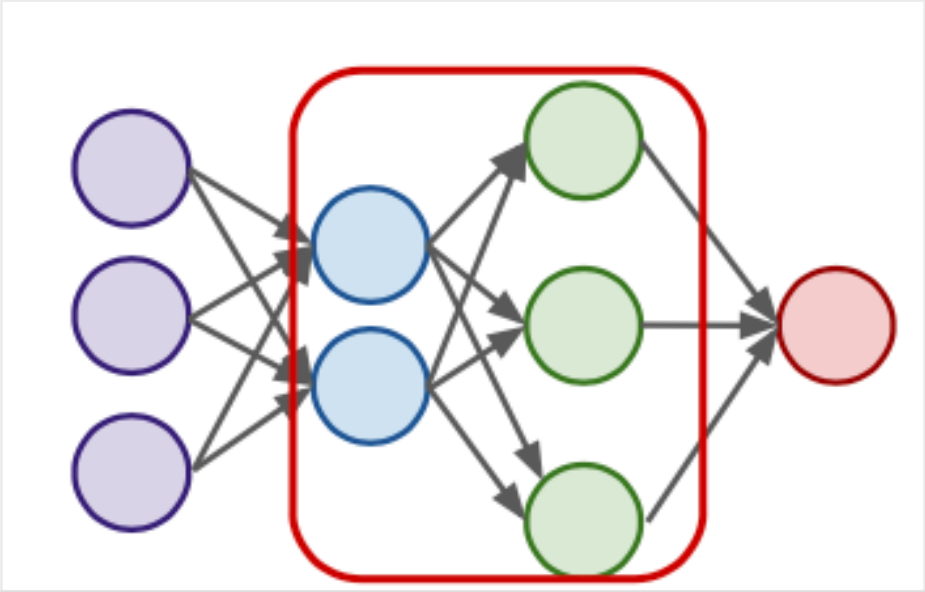
Capa de salida

nota, esta capa puede tener más de una neurona.



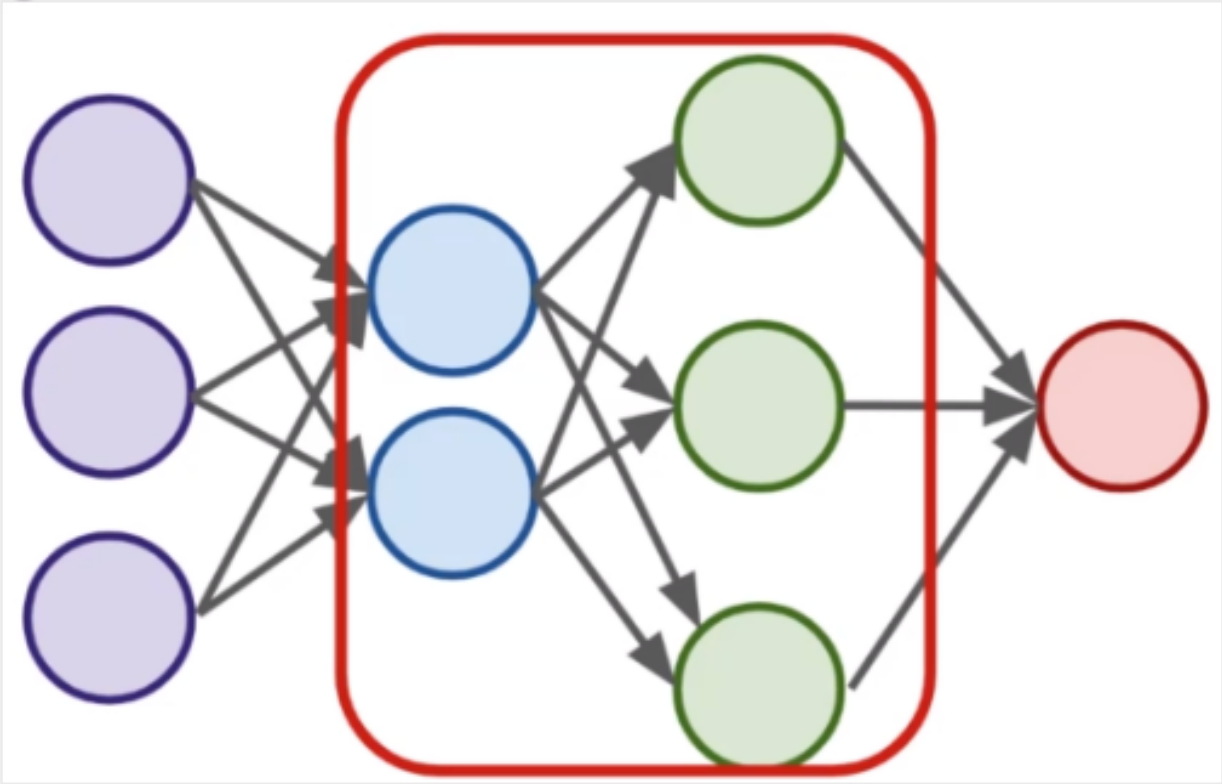
Capas ocultas

las capas entre la salida y la entrada son las capas ocultas.

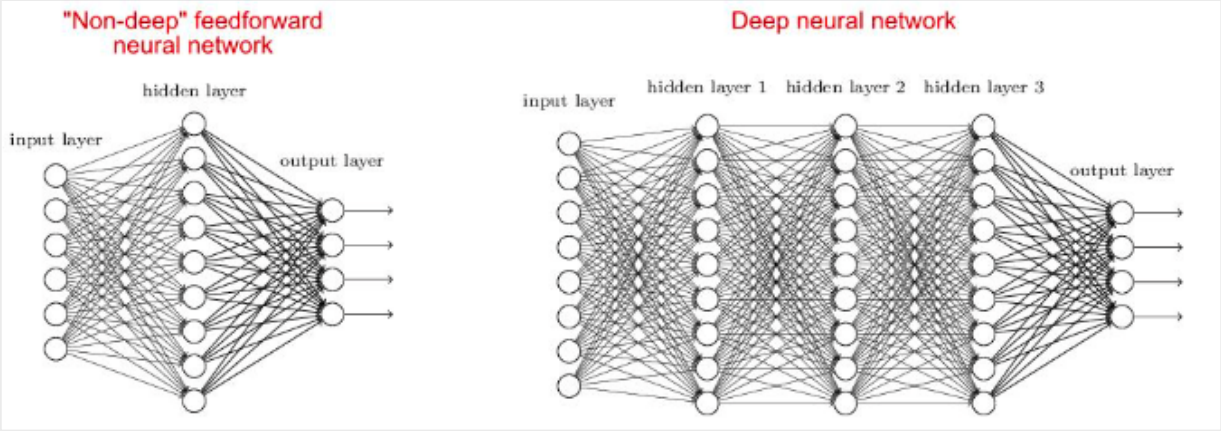


¿Cuándo una red neuronal se convierte en una “red neuronal profunda”?

La terminología indica que cuando se tienen 2 o más capas ocultas, se denomina red neuronal profunda.



Ejemplos



Datos interantes:
Una de las cosas que es increible acerca del marco de trabajo de redes neuronales, es que puede ser usado para aproximar funciones continuas.
Zhou Lu y más tarde Boris Hanin probaron matematicamente que las redes neuronales pueden aproximar cualquier función convexa continua.
Para más info de este punto, ver en wikipedia “Universal approximation theorem”.

OJO
En las presentes notas se habla de la función que contiene las redes neuronales como una suma, pero en realidad eso no será útil en gran parte de los casos, se querrá fijar restricciones a las salidas, especialmente en tareas de clasificación, para ello se detallará a continuación sobre las **funciones de activación.**

Funciones de activación

Podemos interpretar el bias como un offset value, haciendo que la entradas ponderadas por su peso deban alcanzar cierto treshold para poder tener efecto.
Por ejemplo, si tenemos que
b = -10
x*w + b
Entonces vemos que x*w no empezará a tener efecto hasta que sobrepase al bias teniendo un producto superior a 10.

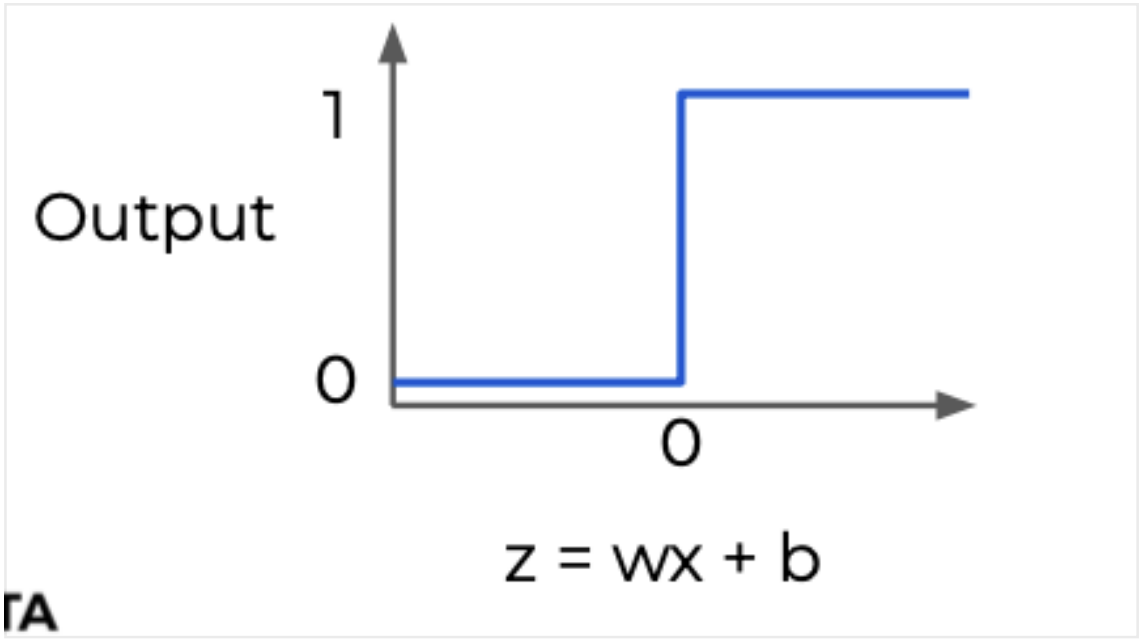
Luego, queremos ajustar un set de limites para la salida total de x*w + b, es decir, pasarlo por unas **funciones de activación** que limiten su valor.

Mucha investigación se ha hecho en las funciones de activación y su efectividad.

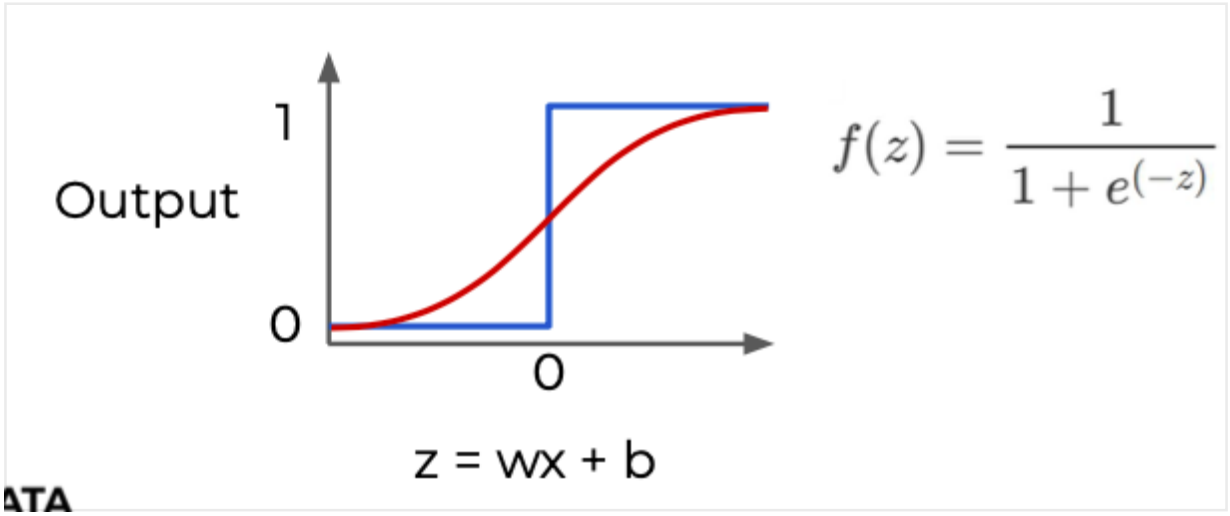
Exploremos algunas útiles:

- Si tuvieramos un problema de clasificación binaria, querríamos un output o 0 ó 1, por lo que podríamos ocupar una función de escalón que mapee ó 0 ó 1.

Independiente del valor, siempre va a dar como salida ó 0 ó 1.

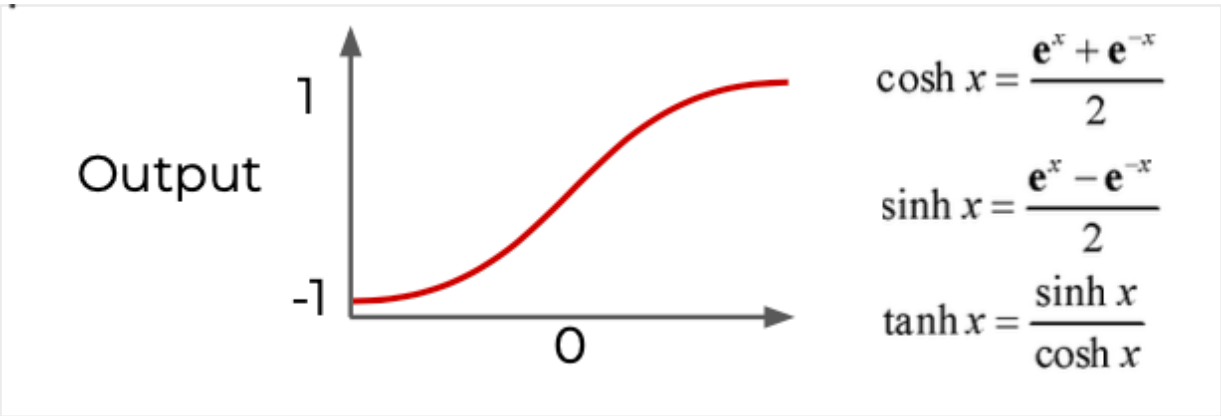


Sin embargo, si quisieramos una función más dinámica que mantenga una forma similar, podríamos utilizar la función sigmoide.

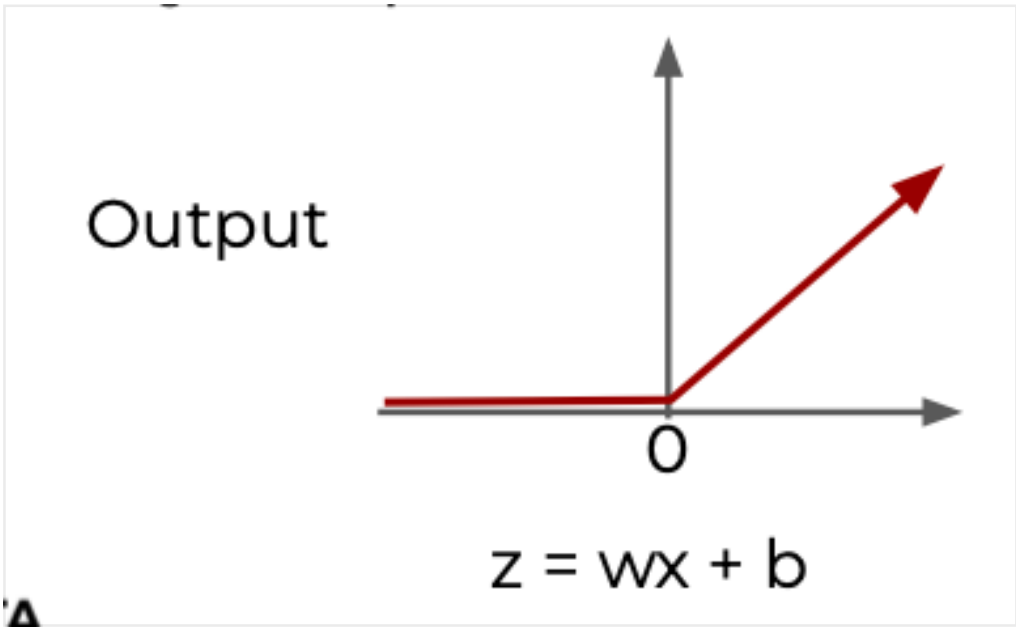


Cambiar la función de activación puede ser bueno dependiendo la tarea, la sigmoide por ejemplo todavía sirve para clasificación y es más sensitiva a pequeños cambios.

Otra función de activación que podrá encontrarse es la tangente hiperbólica.
Tanh(z)



Una de las más usadas y efectivas es la Rectified Linear Unit (ReLU), en realidad bastante simple
max(0,z)



ReLU ha tenido bastante buen performance, especialmente lidiando con el problema del “vanishing gradient”.
Se recomienda siempre ir por default a ReLU dado su buen performance general.