Classification Using Naive Bayes (Lantz)

Kevin García - Alejandro Vargas

26 de agosto de 2019

Contenido

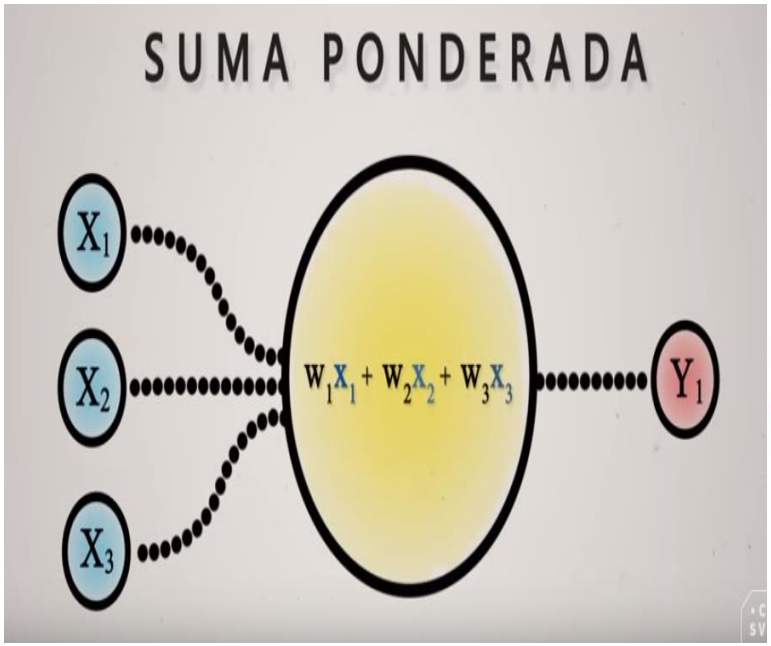
Introducción

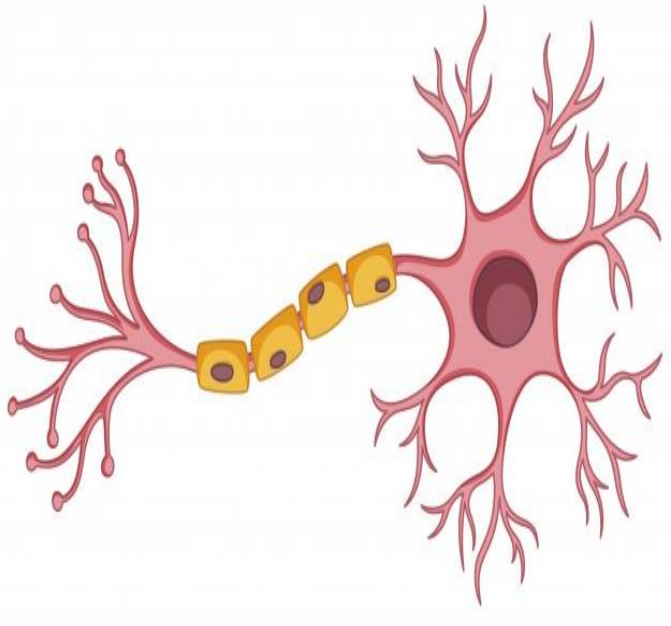
El ser humano en su búsqueda por entender la realidad que lo rodea, se ha empeñado en desarrollar modelos que logran interpretar en planos mas "sencillos" dicha realidad, de forma que sea entendible para todos. Las redes neuronales son el conjunto de algoritmos de Machine Learning más populares; el reconocimiento de caracteres, de imágenes, de voz, generación de texto, traducción de idiomas, predicción de fraude, conducción autónoma y pronóstico de enfermedades son algunos de los muchos ejemplos que existen, en los cuales se hace uso de estos potentes algoritmos con los que podemos modelar comportamientos "inteligentes".

¿Qué son las redes neuronales?

Para entender de manera sencilla este concepto, comenzaremos dando respuesta a la pregunta: ¿Qué es una neurona? Una neurona es la unidad básica de procesamiento dentro de una red neuronal, le hace honor a su nombre, ya que al igual que una neurona biológica, esta tiene unas conexiones de entrada por las cuales recibe estímulos externos, que conocemos como valores de entrada, y con base en estos valores, la neurona realizará un cálculo y arrojará un valor de salida, así pues, una neurona no deja de ser más que una suma ponderada de los valores de entrada $Y = W_1X_1 + W_2X_2 + W_3X_3 + b$, esto lo conocemos como regresión lineal.

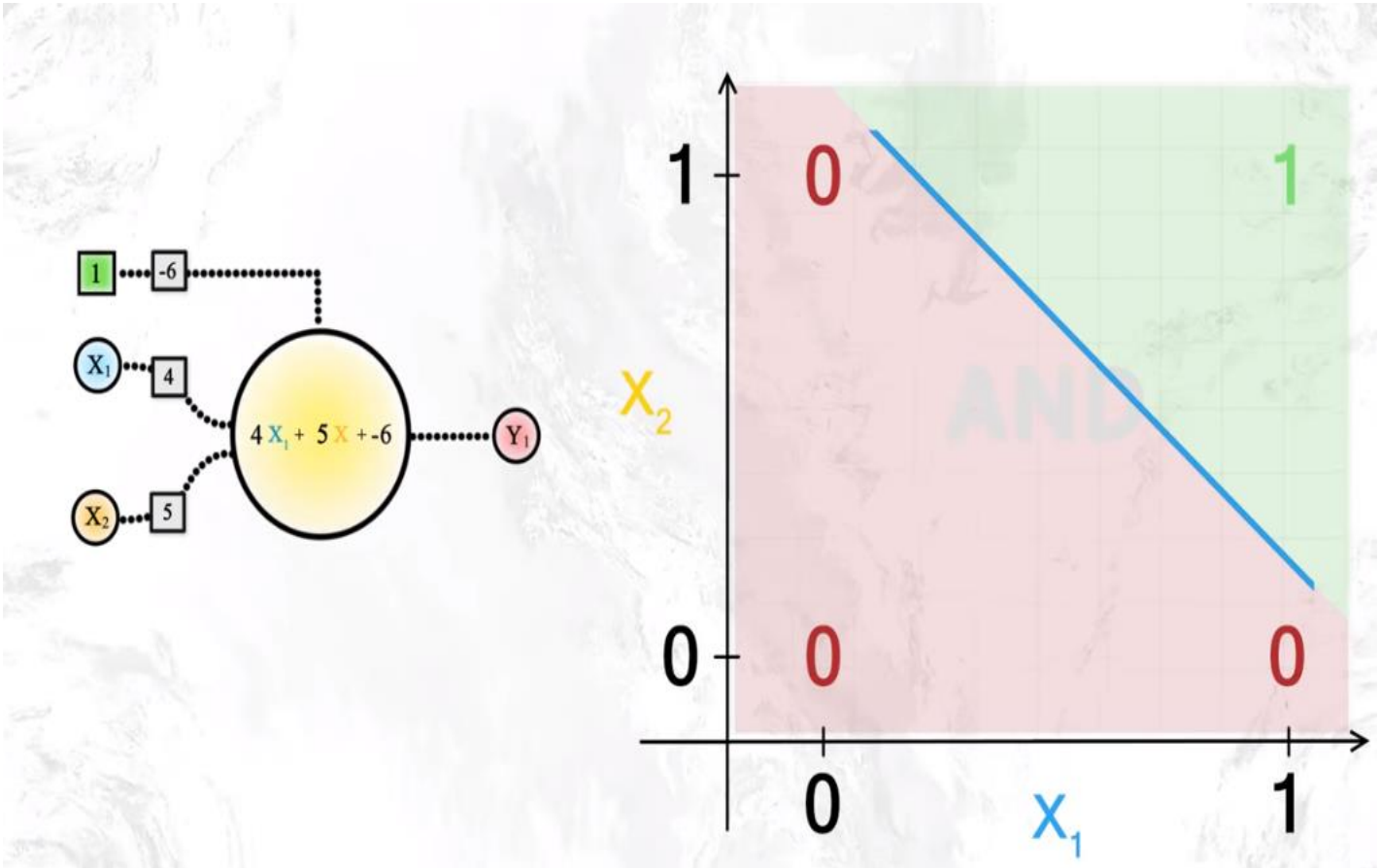
¿Qué son las redes neuronales?



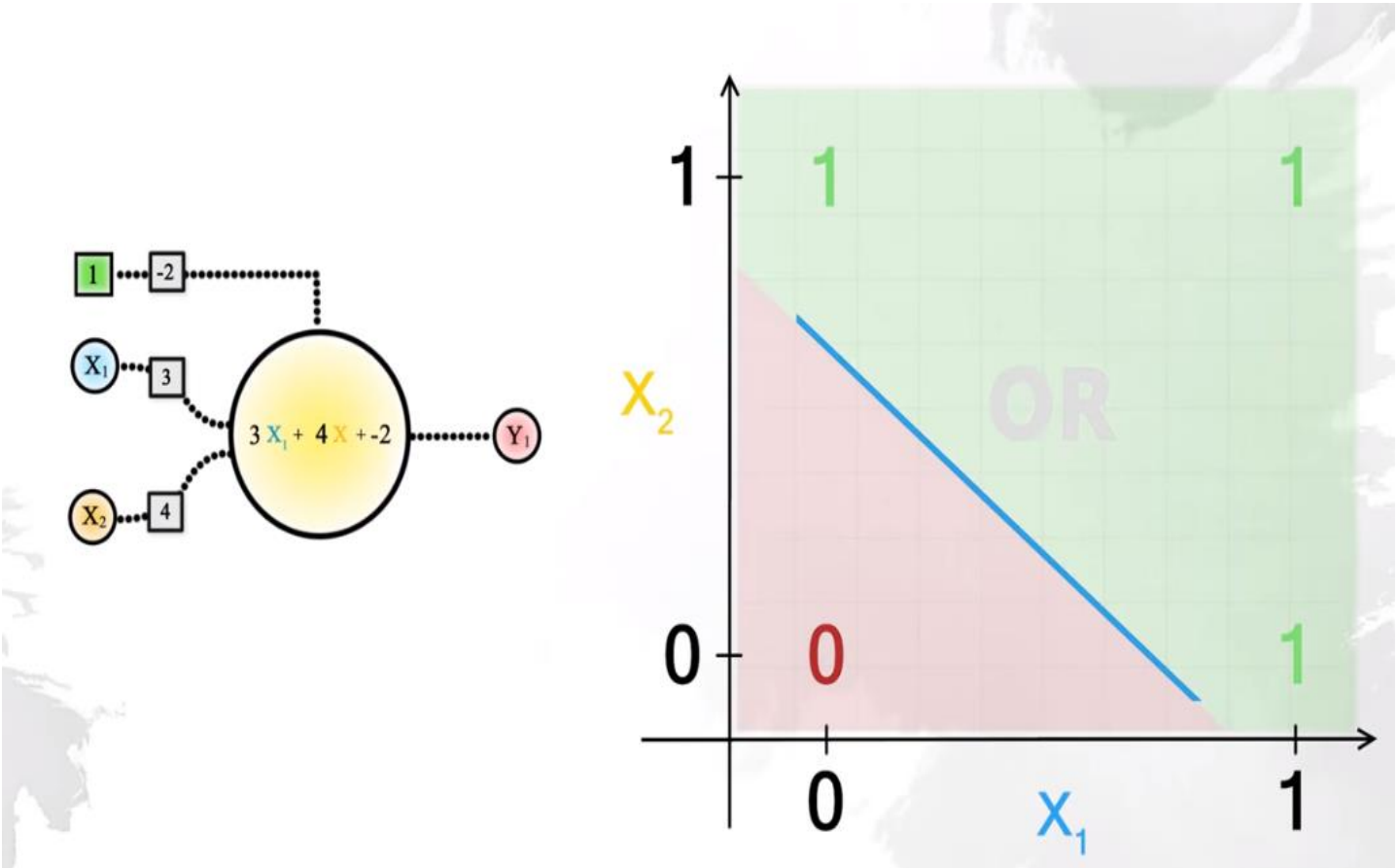


¿Qué son las redes neuronales?

Una regresión separa con una linea recta grupos de puntos:

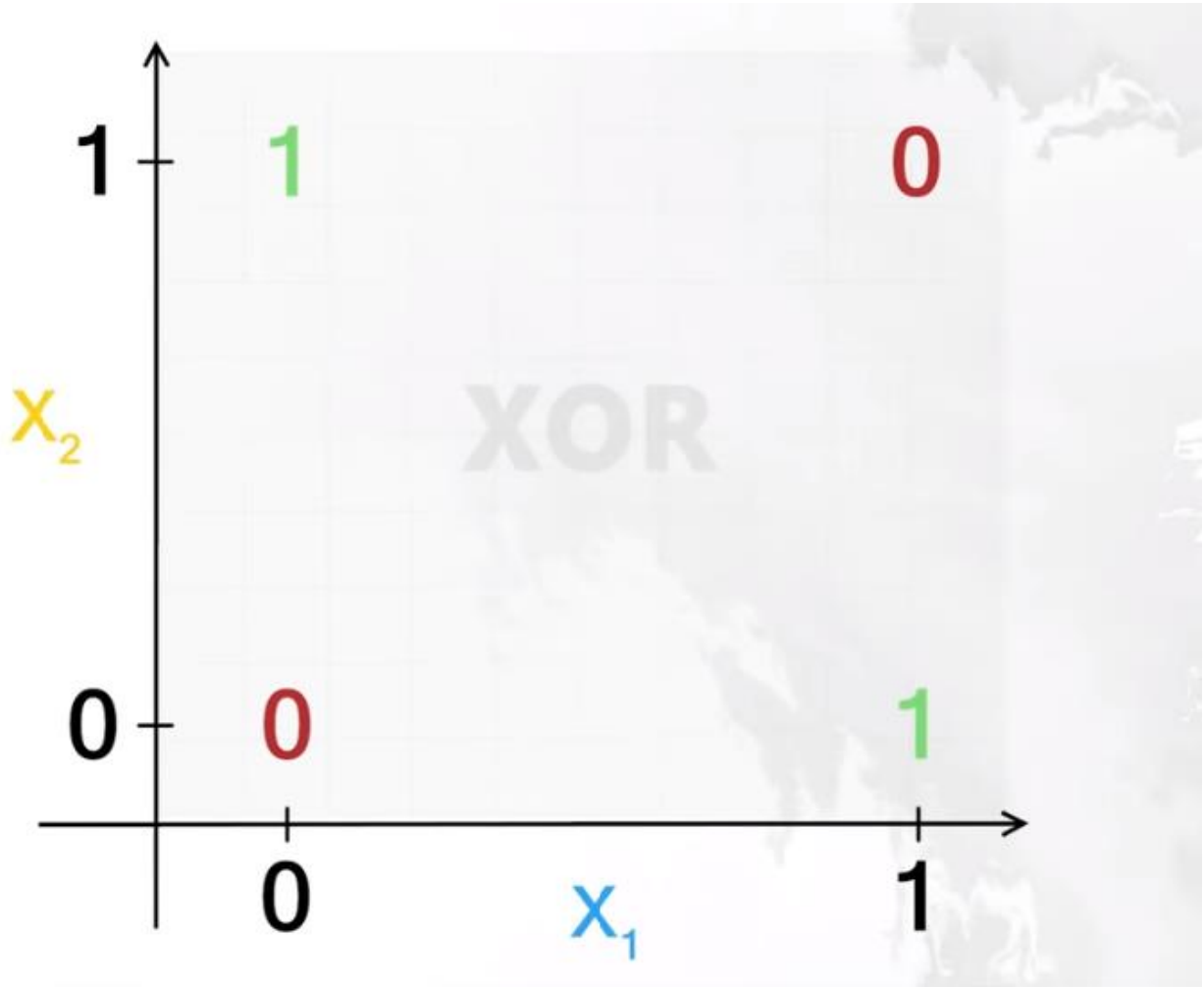


¿Qué son las redes neuronales?



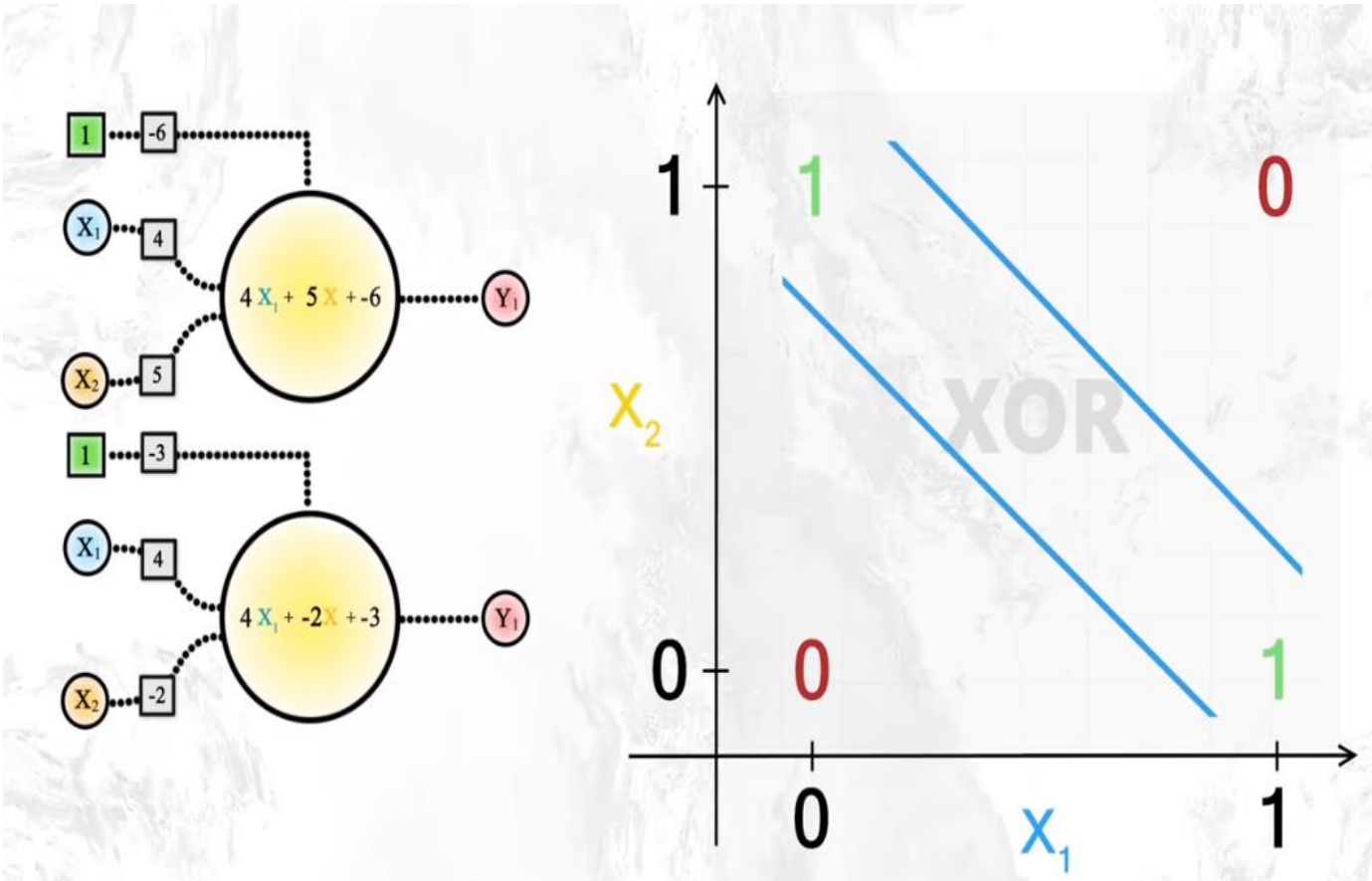
¿Qué son las redes neuronales?

¿Qué sucede cuando no es posible separar una nube de puntos con una linea?



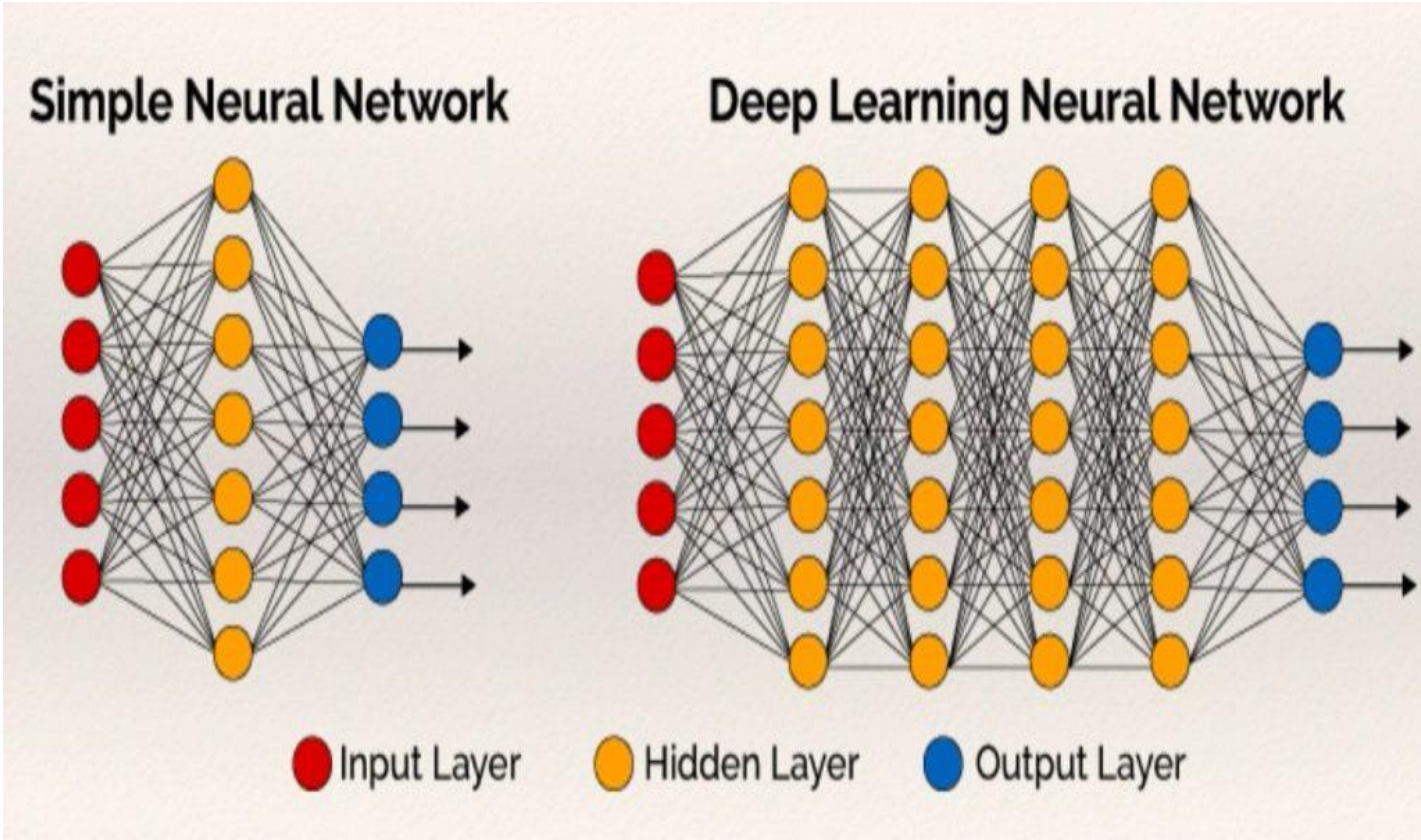
¿Qué son las redes neuronales?

Añadimos otra neurona:



¿Qué son las redes neuronales?

Si añadimos más:



¿Qué son las redes neuronales?

Una red neuronal está compuesta por neuronas o nodos de procesamiento, y estos a su vez están divididos en grupos que se llaman “capas”. Existen 3 tipos de capas: capas de entrada, capas ocultas y capas de salida; las conexiones se establecen entre los nodos de cada capa adyacente, la capa de entrada está formada por nodos de entrada, que reciben la información directamente del exterior; mientras que la capa de salida representa la respuesta de la red a una entrada dada, siendo esta información transferida al exterior.

¿Qué son las redes neuronales?

Una red neuronal también la podemos ver de la siguiente forma:

$$a_l = g(\omega_{\ell 0}^{(1)} + \sum_{j=1}^{p} \omega_{\ell j}^{(1)} X_j)$$

Donde, a_l es la conexión a la capa de entrada a través de un vector de parámetros o pesos $\omega_{\ell j}^{(1)}$ (El (1) hace referencia a el numero de la capa en este caso seria la primera y ℓj hace referencia a la $j - esima$ variable y la $\ell - esima$ unidad). Los términos $\omega_{\ell 0}^{(1)}$ son los sesgos o interceptos y la función g es llamada función de activación,usualmente se utilizan funciones sigmoides en las capas de salidas ya que estas arrojan valores entre 0 y 1 ($g = \frac{1}{(1-e^{-t})}$) y en el resto de capas se suelen utilizar funciones ReLU (Rectified Linear Unit Function).

¿Cómo funcionan?

Al poner dos neuronas de forma secuencial, una de ellas recibe la información procesada por la neurona anterior, esto trae una ventaja muy importante para el aprendizaje de la red y es que la red podrá aprender conocimiento jerarquizado, lo que sería en palabras simples, que mis neuronas en las primeras capas sean capaces de procesar cosas simples y mis neuronas en capas posteriores hagan procesos más complejos con la información suministrada por las primeras capas.

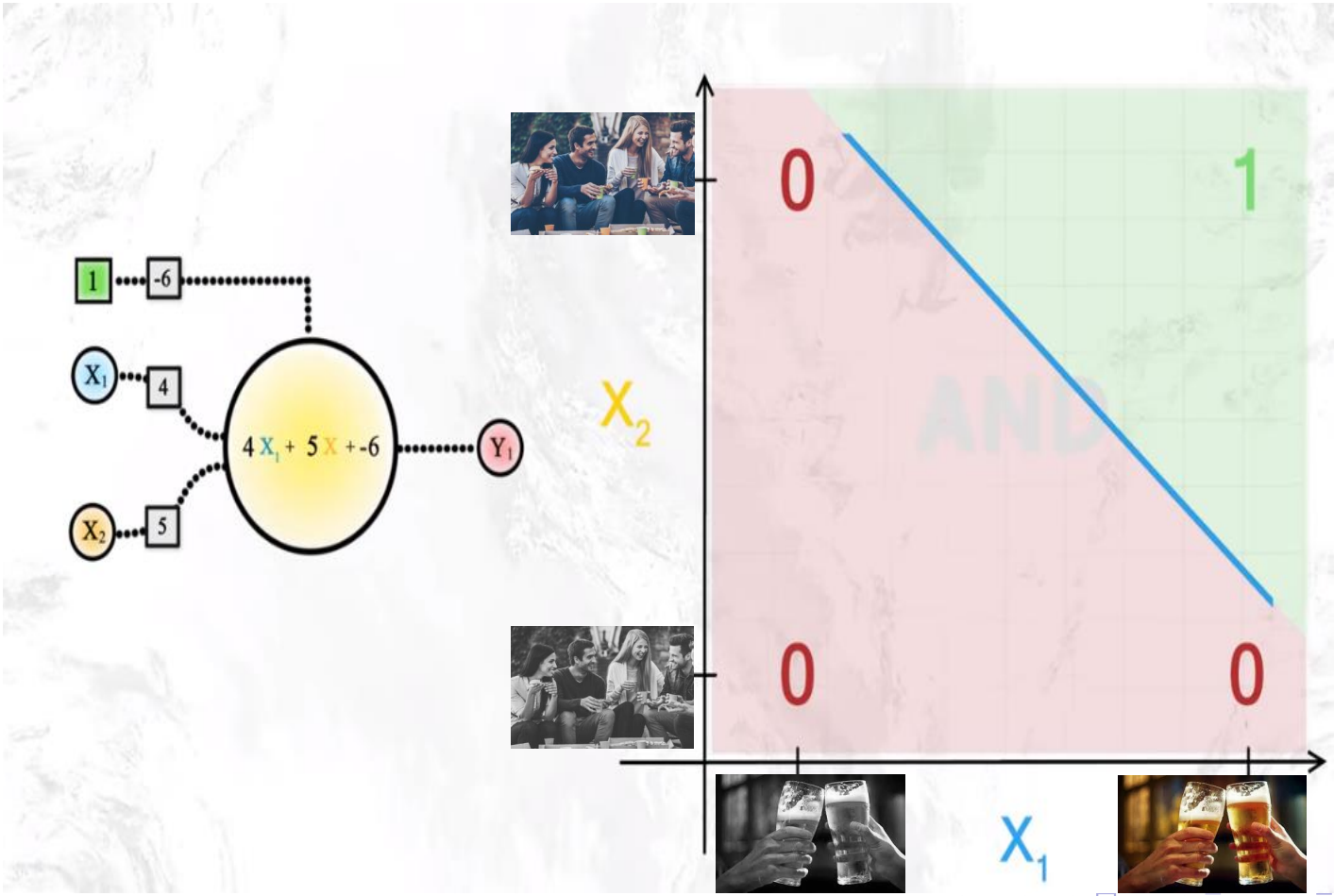
¿Cómo funcionan?

Un ejemplo, un fin de semana perfecto es:

+

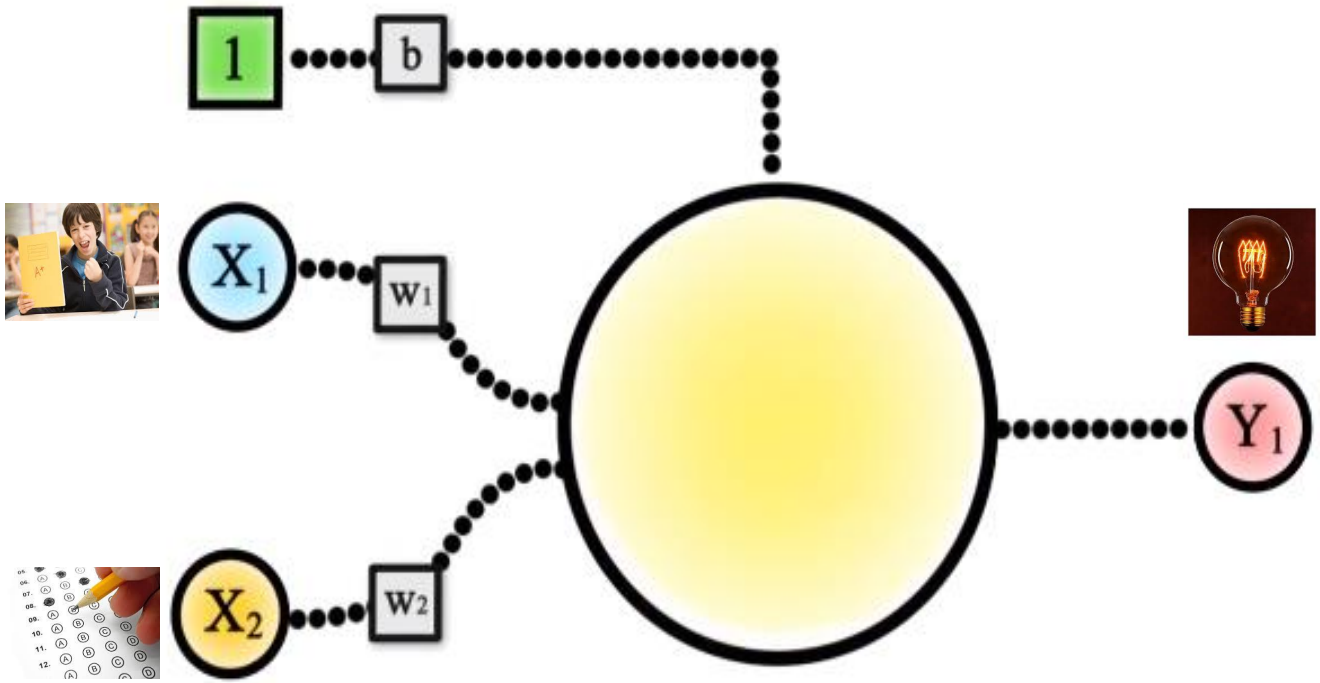
¿Cómo funcionan?

No es posible que sea perfecto si una de estas condiciones no se cumple; con una regresión simple podría modelar si mi fin de semana va a ser perfecto o no.



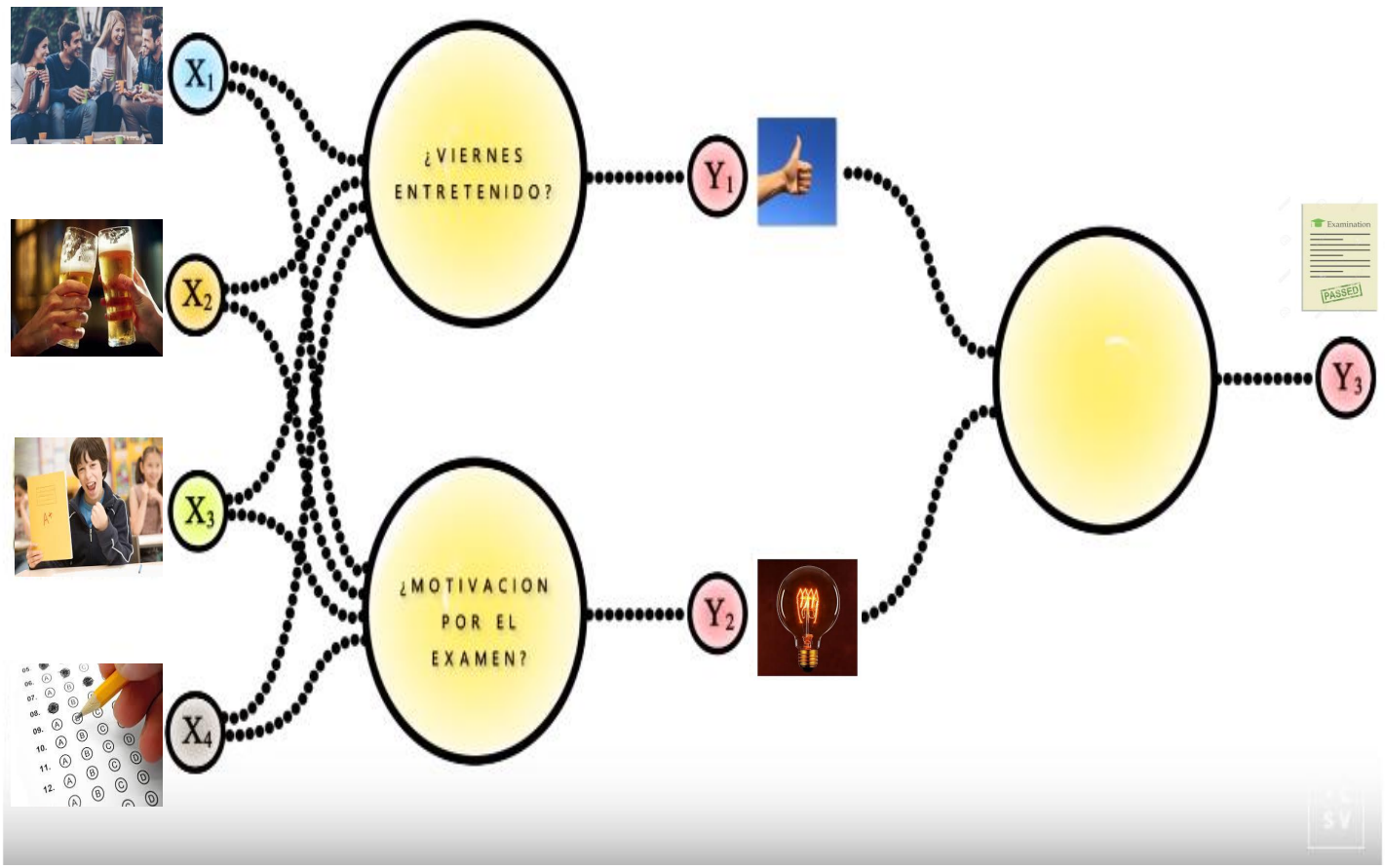
¿Cómo funcionan?

Pero si mi objetivo final es conocer algo más complejo como, ¿cuál sera mi calificación en el examen final del lunes?. A lo mejor existen otras 2 variables relacionadas, interés por la materia y dificultad del examen.



¿Cómo funcionan?

Uniendo ambas neuronas obtenemos una posible arquitectura de red neuronal:

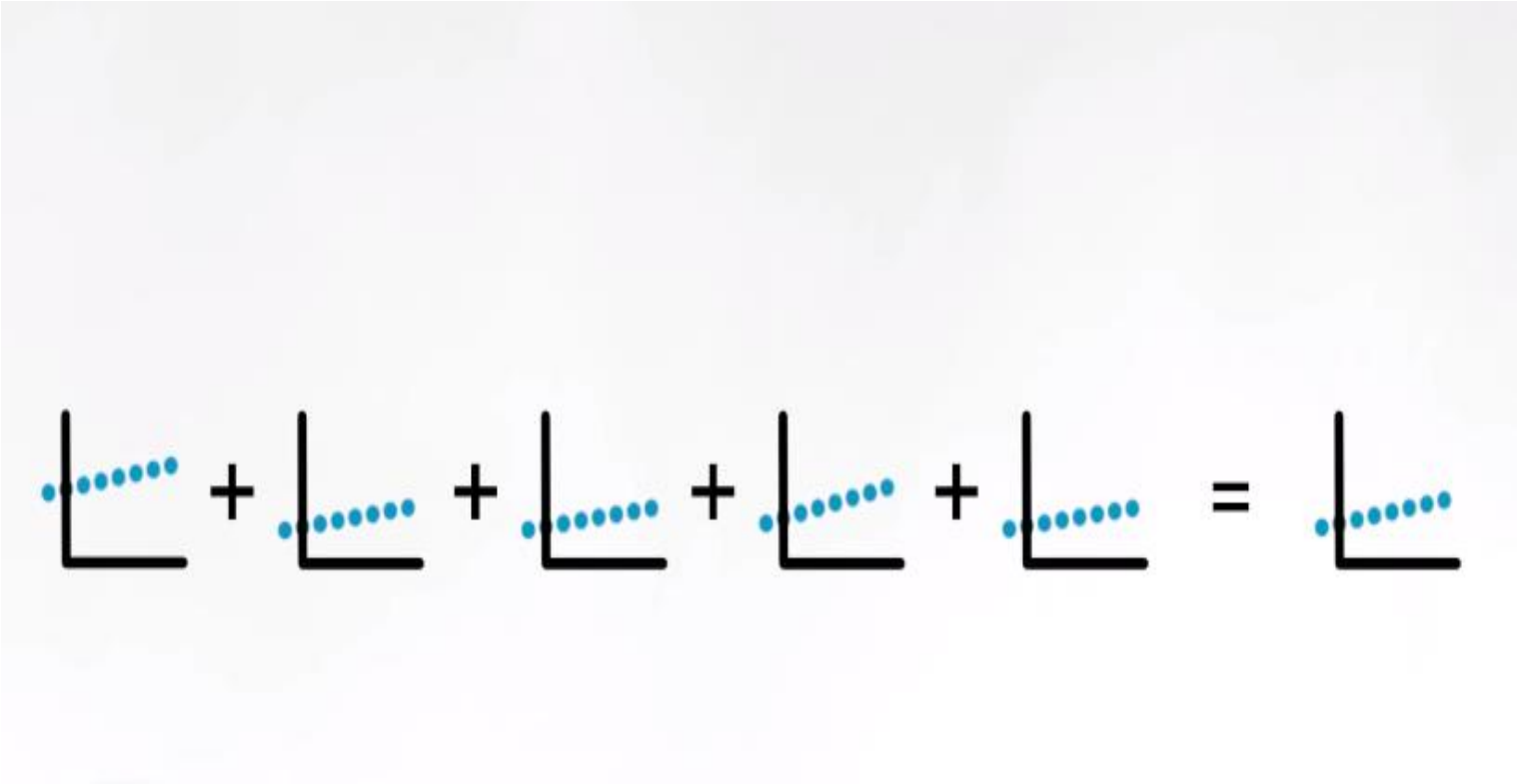


¿Cómo funcionan?

Mi red puede distinguir que estudiantes van a aprobar el examen del lunes; si su motivación por el examen es baja y su fin de semana fue perfecto, posiblemente van a estudiar poco y reprobarán el examen. De esta manera jerarquizada mi red puede generar conocimiento complejo, entre más neuronas y capas sean agregadas, más complejo será. A esta manera de aprender se le llama "Deep Learning".

¿Cómo funcionan?

Para conseguir este aprendizaje profundo en nuestra red hay que conectar múltiples neuronas de forma secuencial, como lo que realiza cada neurona es una regresión lineal, matemáticamente estamos sumando múltiples regresiones lineales.



¿Cómo funcionan?

Para que esto no suceda y nuestra red pueda aprender debemos adicionar deformaciones no lineales a nuestras regresiones; estas deformaciones se adicionan mediante la función de activación.

Funciones de activación

Una función de activación añade deformaciones no lineales, de manera que mis valores de salida pasan primero por una determinada función, esto hace que mi red no colapse en una única regresión lineal. Estas deformaciones dependen de la función de activación utilizada, existen varios tipos, las más usadas son:

Función sigmoide

La función sigmoide denotada de la forma $G(x) = \frac{1}{(1-e^{-x})}$ hace que los valores muy altos se saturen en 1 y los valores bajos en 0 por lo tanto esta función añade dicha deformación no lineal y además representa muy bien probabilidades.

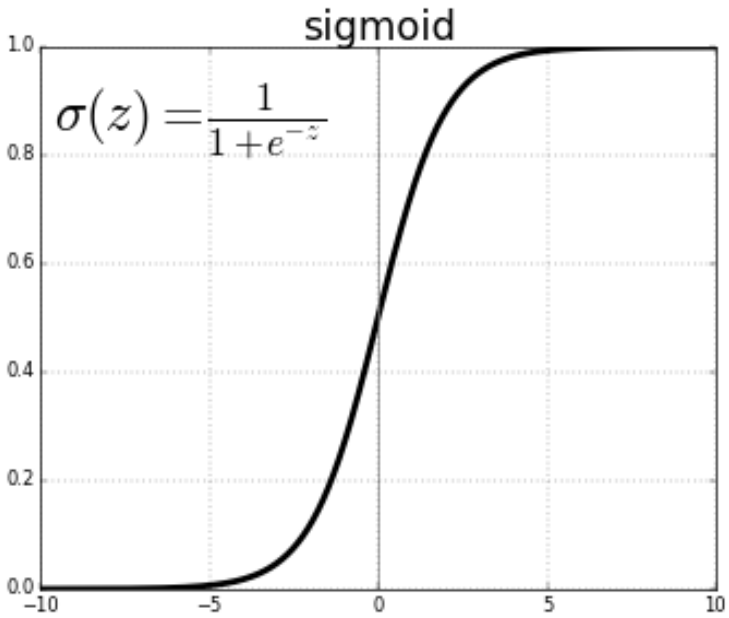


Figura: Función sigmoide

Función Tanh

Similar a la función sigmoide está la función tangente hiperbólica, su forma es similar a la sigmoide pero su rango varia entre -1 y 1 y está dada por $G(x) = tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$

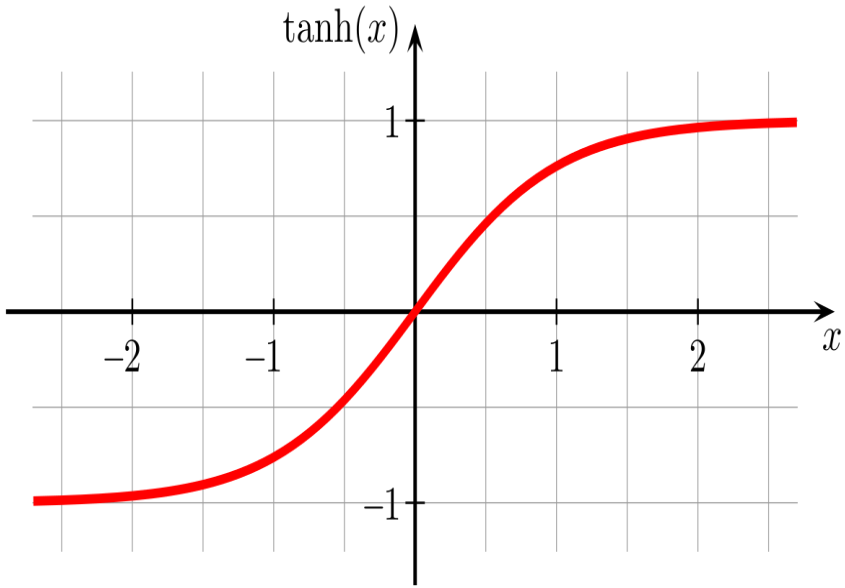


Figura: Función Tanh

Función ReLU

Esta función se comporta de forma lineal cuando es positiva y constante a 0 cuando el valor de entrada es negativo, se puede denotar como $G(x) = max(0, x)$

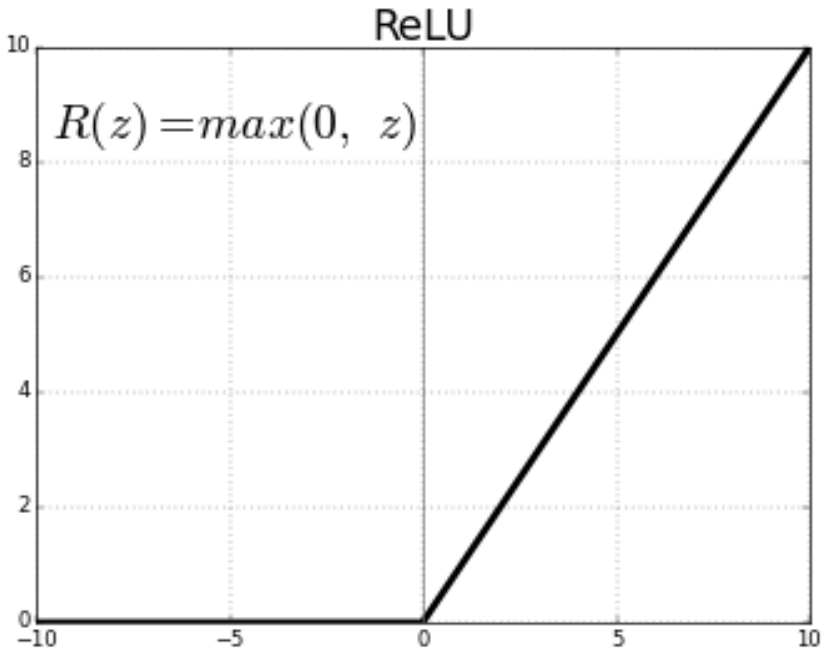


Figura: Función ReLU

Funciones de activación

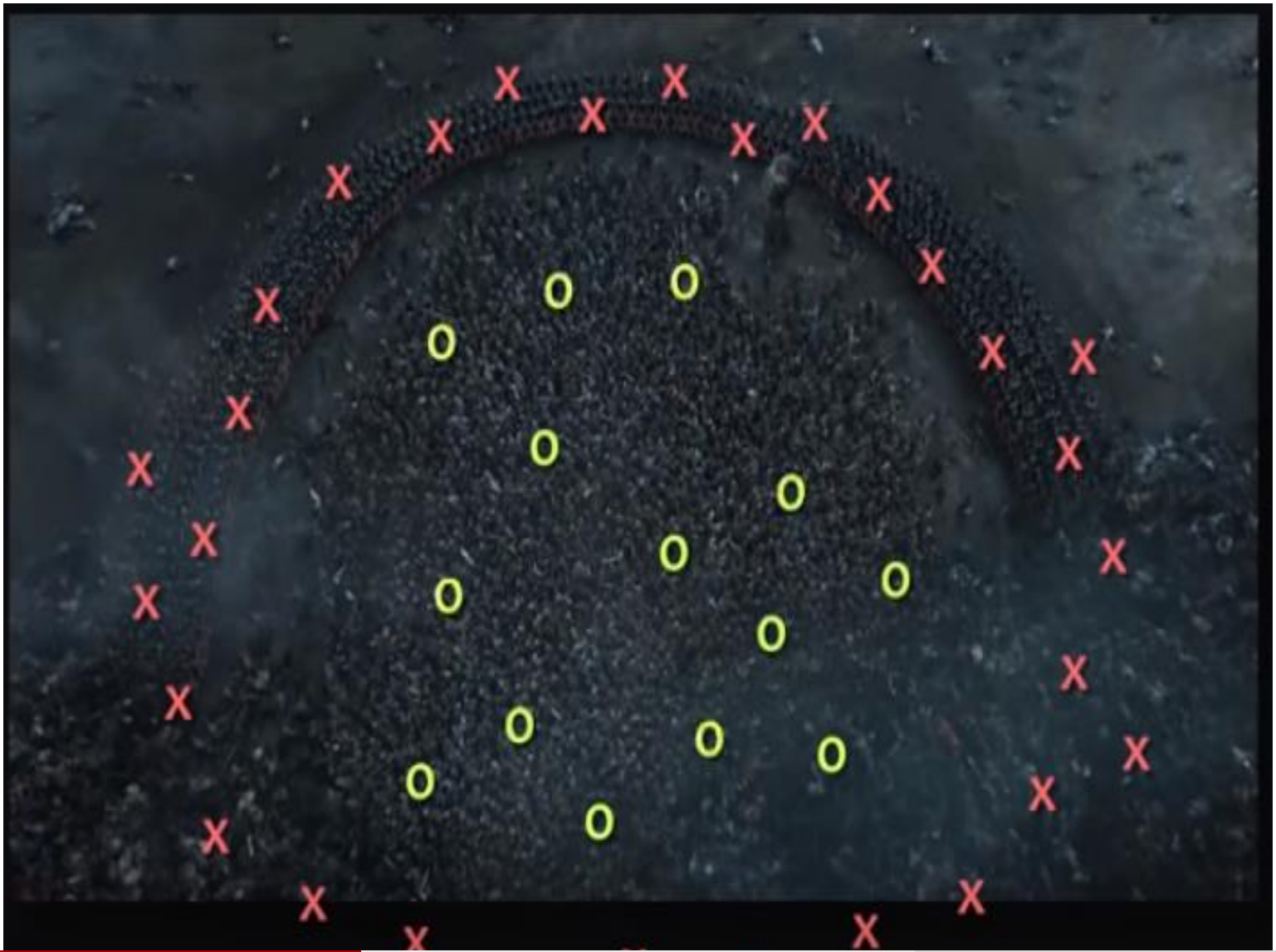
Para entenderlo mejor, veamos un ejemplo:

Funciones de activación

Los que han visto esta serie recordarán esta situación (spoiler):

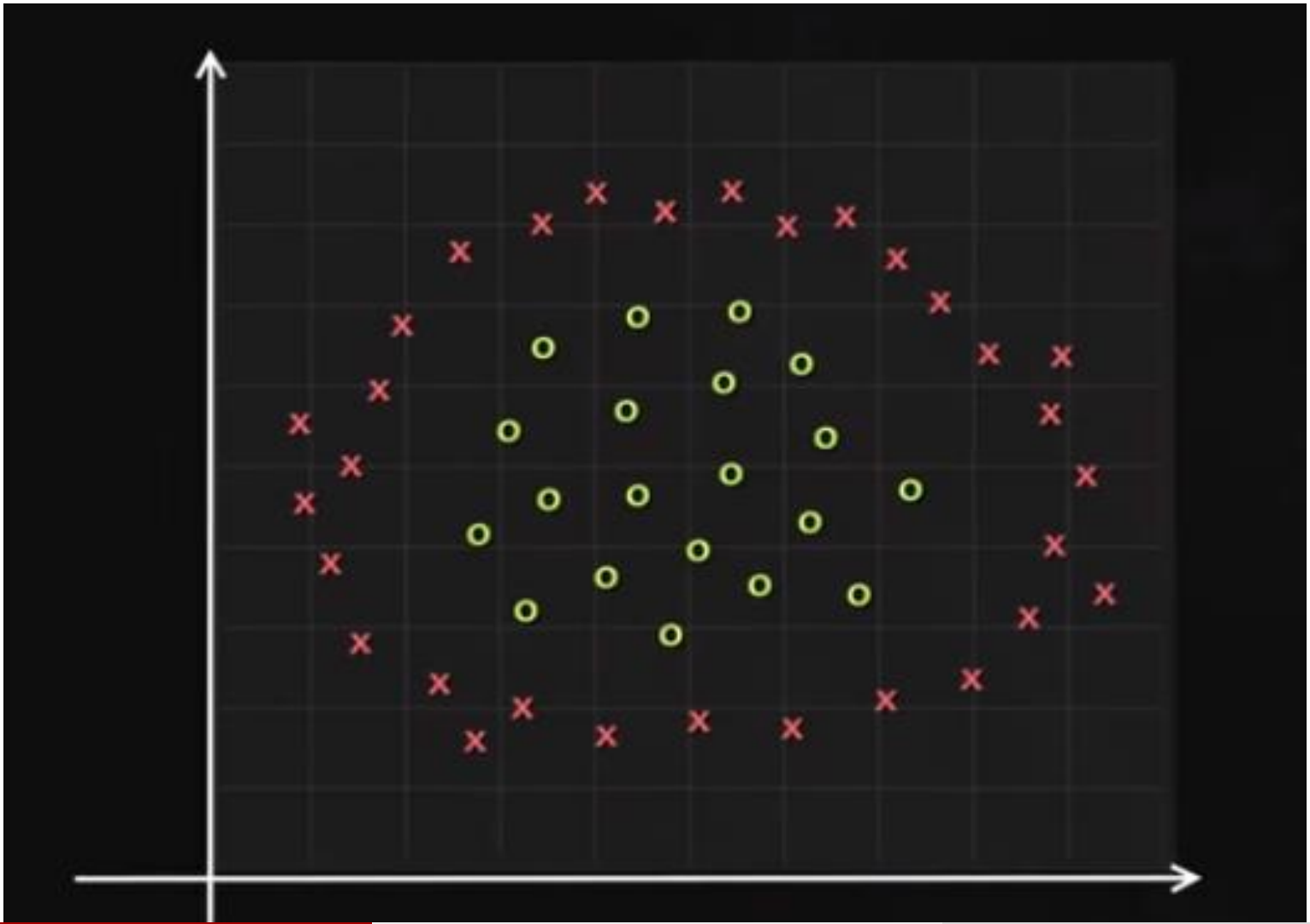
Funciones de activación

Separando los "buenos" de los "malos".



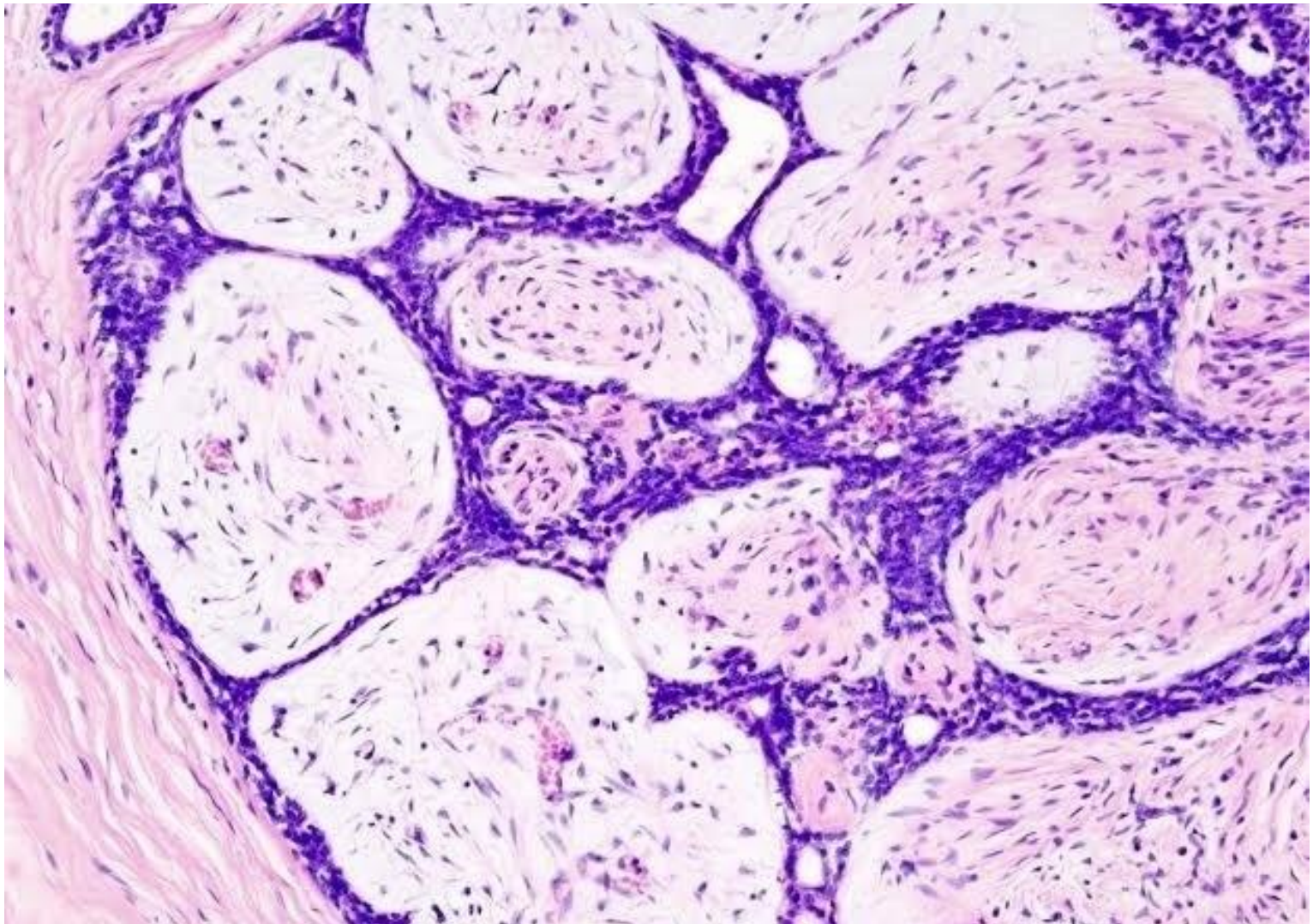
Funciones de activación

Para intentar salvar a Jon Snow y a sus amigos utilizando una red neuronal, debemos resolver el problema de separar ambas nubes de puntos.



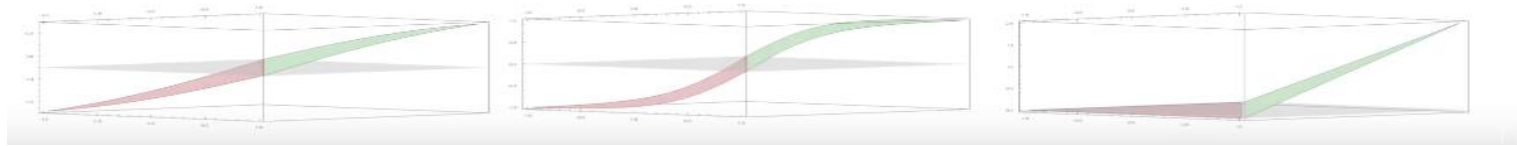
Funciones de activación

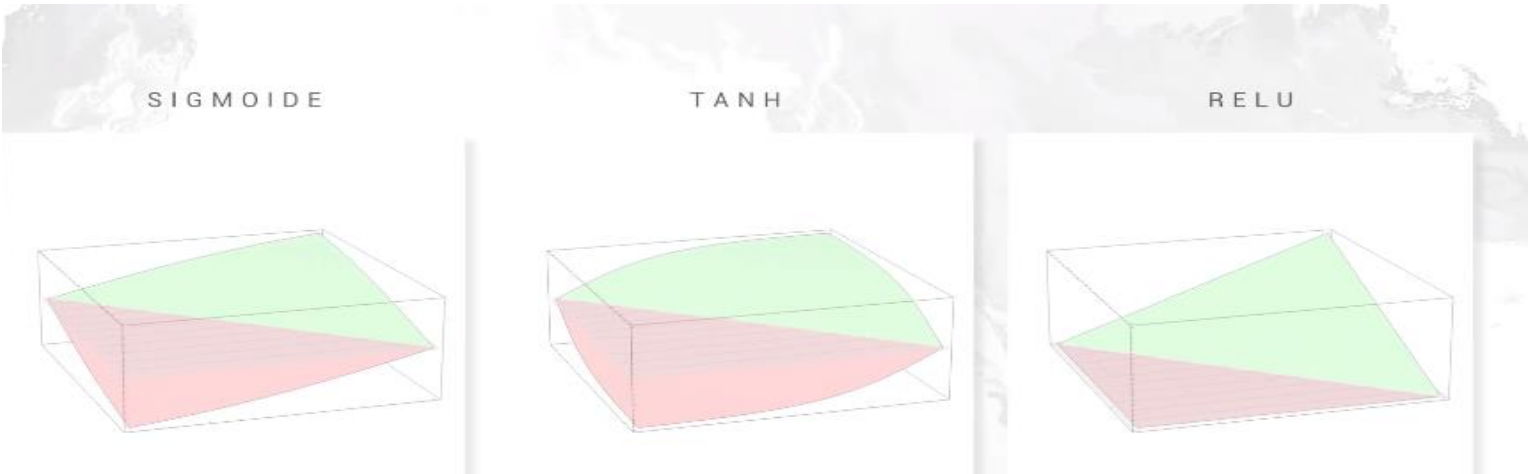
En la vida real, este problema es similar al de diferenciar en una imagen células cancerígenas de las que no lo son.



Funciones de activación

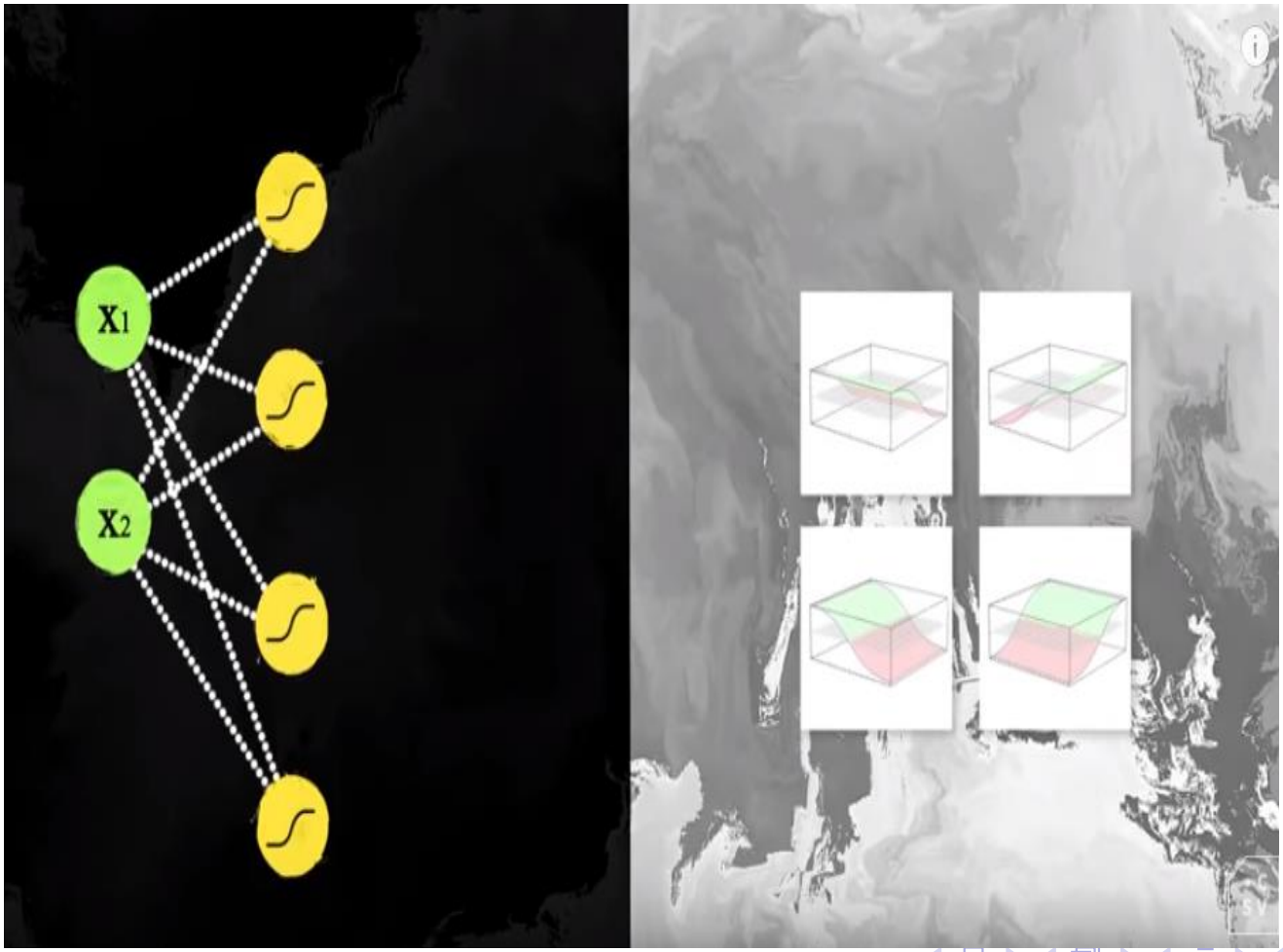
De forma geométrica, una red neuronal utiliza las funciones de activación para separar nubes de puntos.





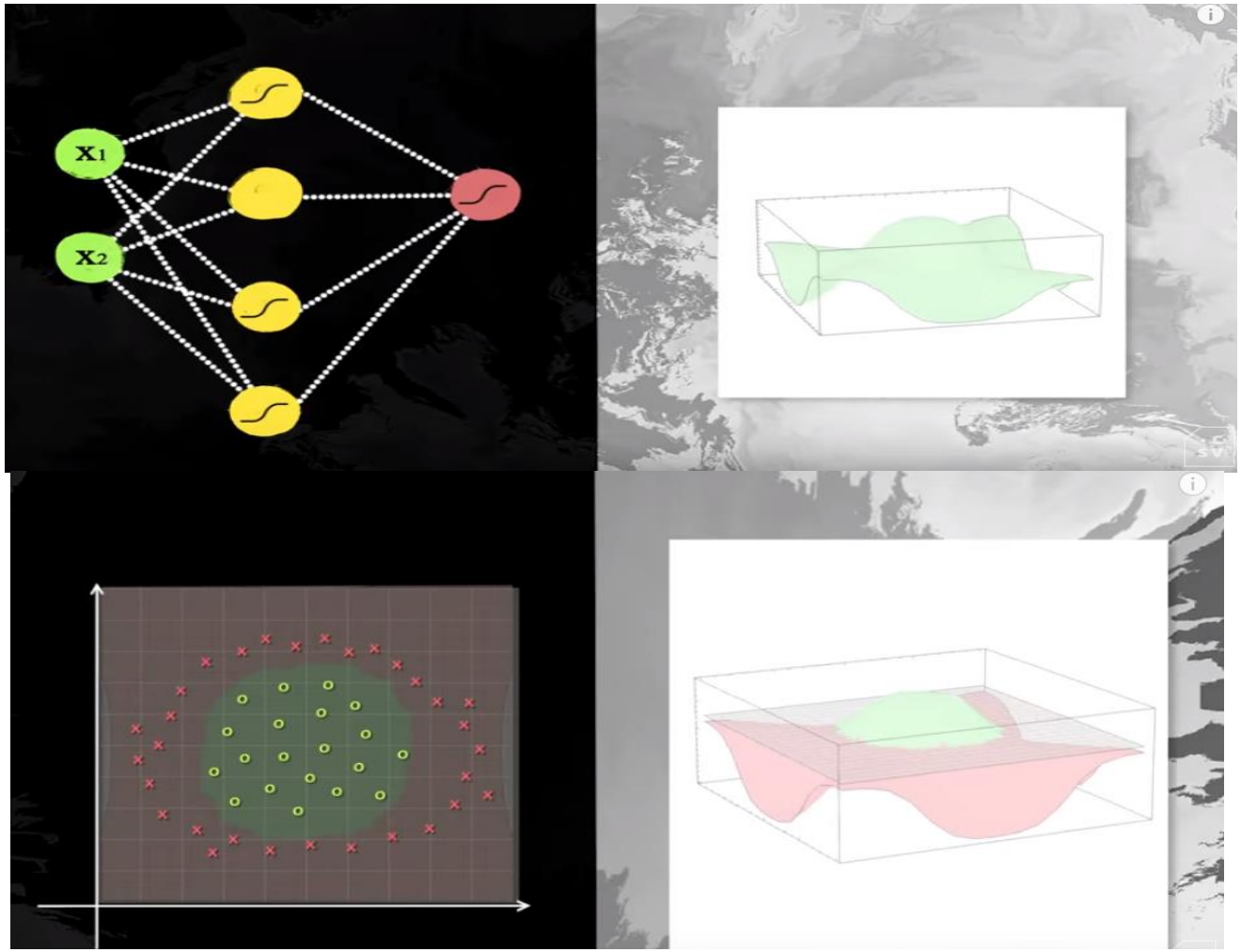
Funciones de activación

Esto se logra combinando varias neuronas; en nuestro problema particular encadenamos varias neuronas con funciones de activación sigmoide, cada una con orientación distinta.



Funciones de activación

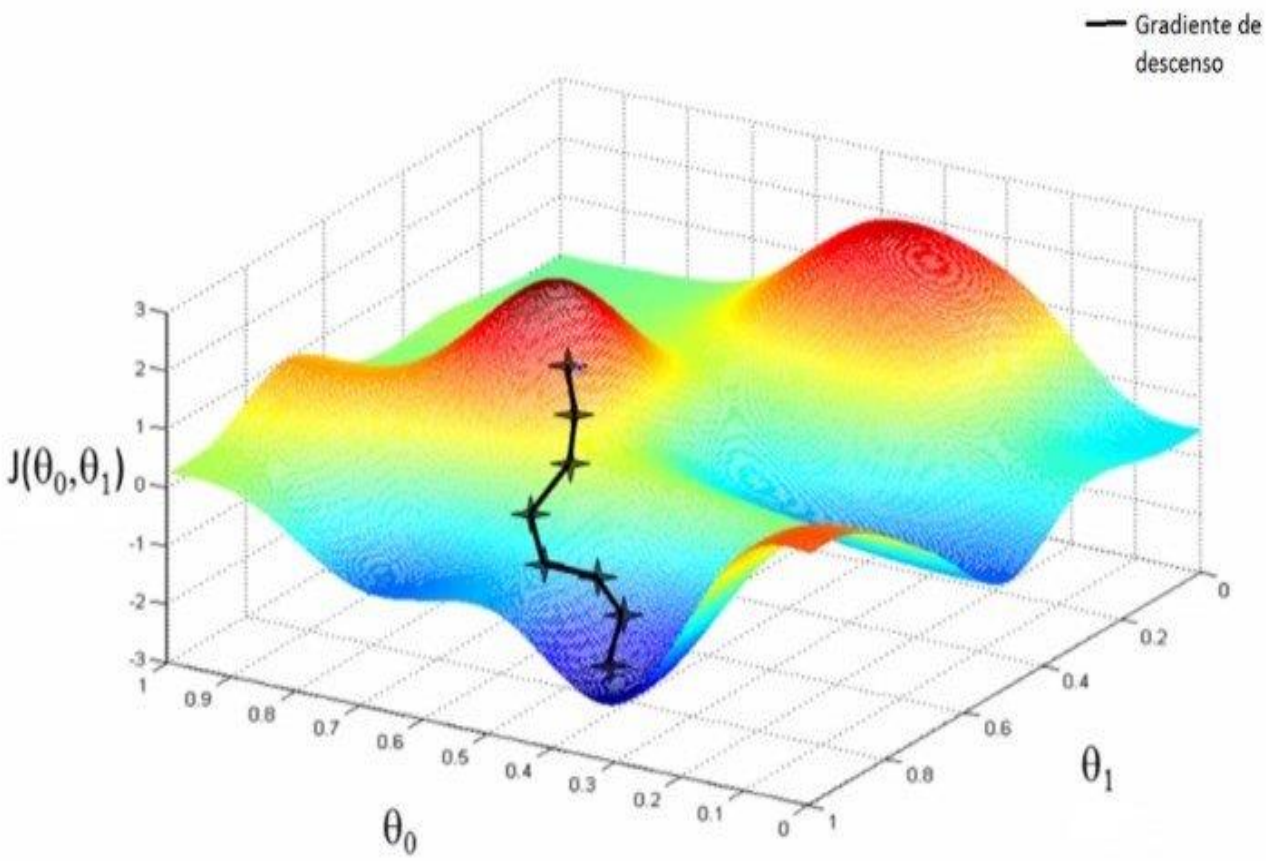
Al unir estas neuronas obtenemos la forma que separa las dos nubes de puntos, una superficie plana con un bulto en medio.



Gradient Descent

Para entender de forma intuitiva este algoritmo imaginemos que estamos en la punta de una montaña y queremos descender al punto más bajo, sin embargo vamos con los ojos vendados, lo más lógico es ir tanteando la inclinación del terreno y desplazarnos por donde la pendiente descienda con mayor intensidad, los pasos a seguir serían evaluar la inclinación para conocer la dirección con la mayor pendiente, luego caminamos una distancia en dicha dirección y nos detenemos, volvemos a repetir hasta llegar a un lugar plano.

Gradient Descent



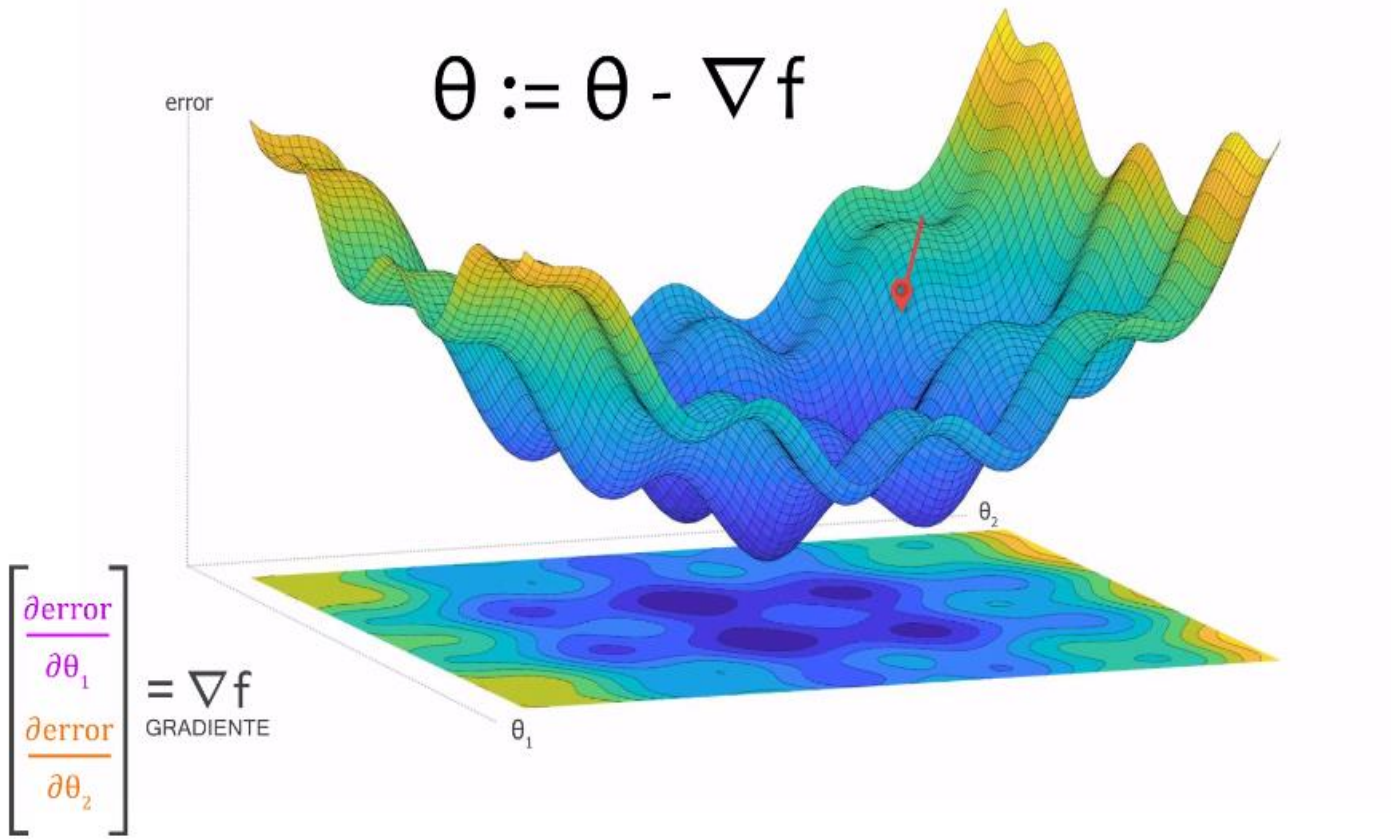
Gradient Descent

Matemáticamente, esto seria calcular las derivadas de la funcion de coste para cada parámetro en un punto aleatorio, estos nos dirá la pendiente en ese punto para cada parámetro, el vector de estas derivadas se denomina vector gradiente ∇W y nos indica la dirección hacia la que la pendiente asciende, pero como lo que queremos es descender, es lógico trabajar con $-\nabla W$ la dirección contraria, nos desplazamos en esa dirección y repetimos el proceso, calculamos las derivadas parciales para el nuevo punto y volvemos a movernos iterativamente hasta llegar a un punto donde movernos no suponga una variación notable en el coste, es decir que la pendiente sea próxima a nula.

Gradient Descent

El algoritmo del descenso del gradiente lo podemos denotar como $\theta := \theta - \alpha \nabla W$ donde θ es mi parámetro el cual se va a actualizar y α es el ratio de aprendizaje.

Computo del gradiente.

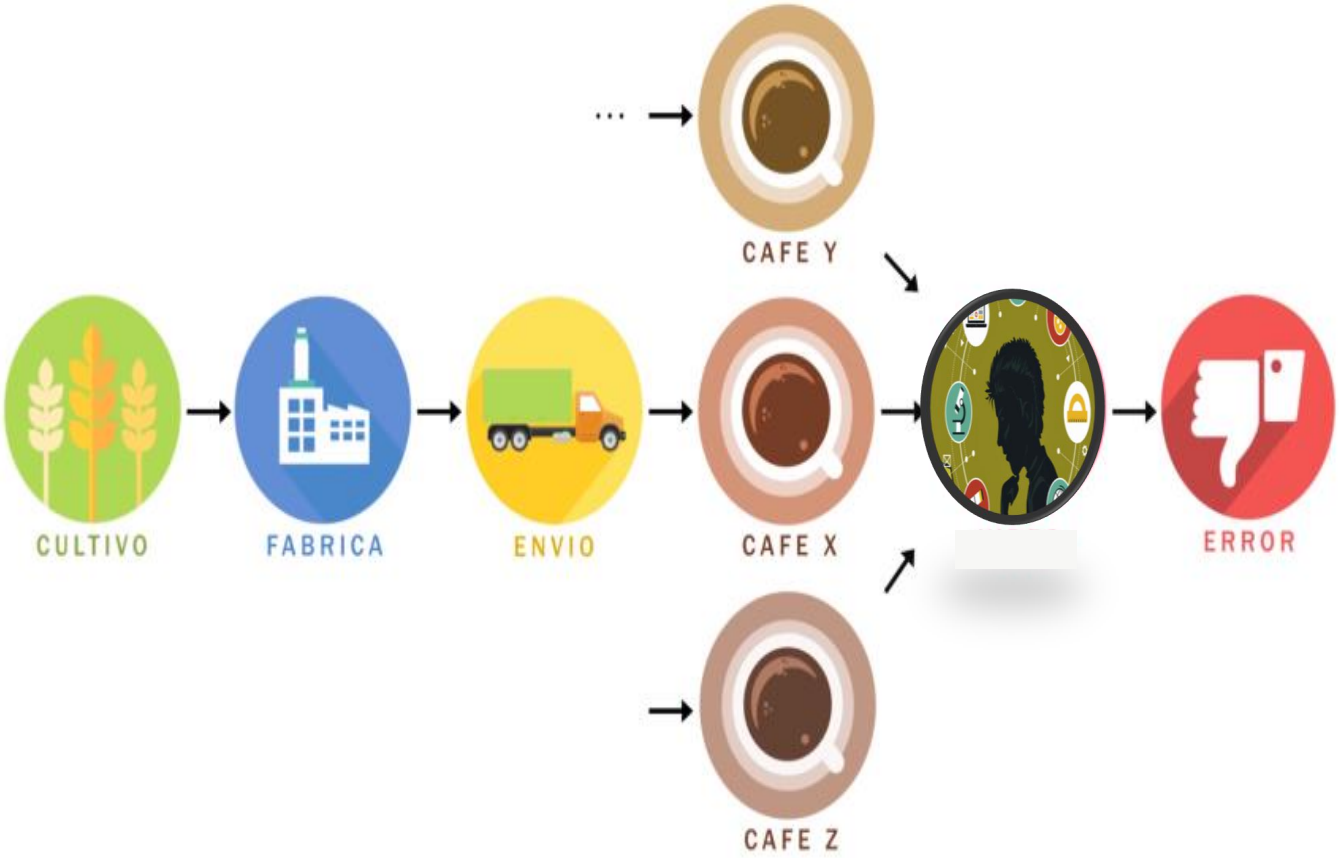


Backpropagation

El algoritmo de Backpropagation sirve para conocer qué error le corresponde a cada una de las neuronas en una red neuronal. Para hacer uso del descenso del gradiente es necesario conocer un vector gradiente y este solo se consigue conociendo la función de coste asociada.

Backpropagation

Pensemos en la siguiente cadena de responsabilidades.



Backpropagation

Esta cadena de responsabilidades se puede asemejar a una red neuronal donde cada nodo es una neurona con una tarea determinada, el examen es el resultado de salida o Output layer y la evaluación que obtenemos la podemos ver como nuestra función de coste, así, ese resultado desfavorable en el examen genera una fuerte señal de error, lo que se hace es analizar toda la cadena de responsabilidades que ha afectado el resultado y si encontramos una neurona que tuviera influencia en el error obtenido, entonces debemos responsabilizarla con parte de ese error, este análisis debe realizarse hacia atrás, desde la señal de error hacia las primeras capas y esto es así, ya que en una red neuronal el error de las capas anteriores depende del error de las capas posteriores, por ejemplo si nos damos cuenta que el resultado del examen no depende mucho de si la calidad del café es buena o mala, entonces el error en las fases anteriores, como el método de molido o el método de cosechado también afectan poco al error final.

Backpropagation

Con el algoritmo de Backpropagation obtenemos el error que tiene cada neurona en el error final, y con esto, podemos utilizar el descenso del gradiente para minimizar el error en cada una de las neuronas como si se tratara de una única regresión.

Estos algoritmos se pueden escribir de forma que:

$$\nabla W^k = \frac{1}{n} \sum_{i=1}^{n} \frac{\delta L[y_i, f(x_i; W)]}{\delta W^k}$$

$$W^k := W^k - \alpha \nabla W^k$$

Conclusiones

Podemos decir que aunque su funcionamiento es complejo y en algunos casos abstracto, las redes neuronales son útiles para resolver problemas en muchas áreas del conocimiento y la tecnología, automatizando procesos, logrando replicar comportamientos inteligentes y dando cabida a un mundo de posibilidades.

1. Las funciones de activación resuelven el problema de la linealidad dentro de las redes neuronales de manera optima, dando soluciones a problemas complejos que no son fáciles de modelar con regresiones lineales
2. EL algoritmo del descenso del gradiente y backpropagation cumplen una función importante en la anatomía de una red neuronal, ya que son el centro del aprendizaje de la misma.
3. Las redes neuronales son capaces de modelar comportamientos tan abstractos, que muchas de estas son completas cajas negras, y logran encontrar relaciones o asociaciones entre variables las cuales un humano no imaginaría.

Bibliografía


Bradley Efron and Trevor Hastie.
Computer age statistical inference. Algorithms, Evidence, and Data Science.
Cambridge, 2017.


Brett Lantz.
Machine learning with R.
Packt Publishing, 2013.


Carlos Santana Vega.
Aprendiendo inteligencia artificial.