Inteligencia Artificial II

Juan Pablo Restrepo Uribe

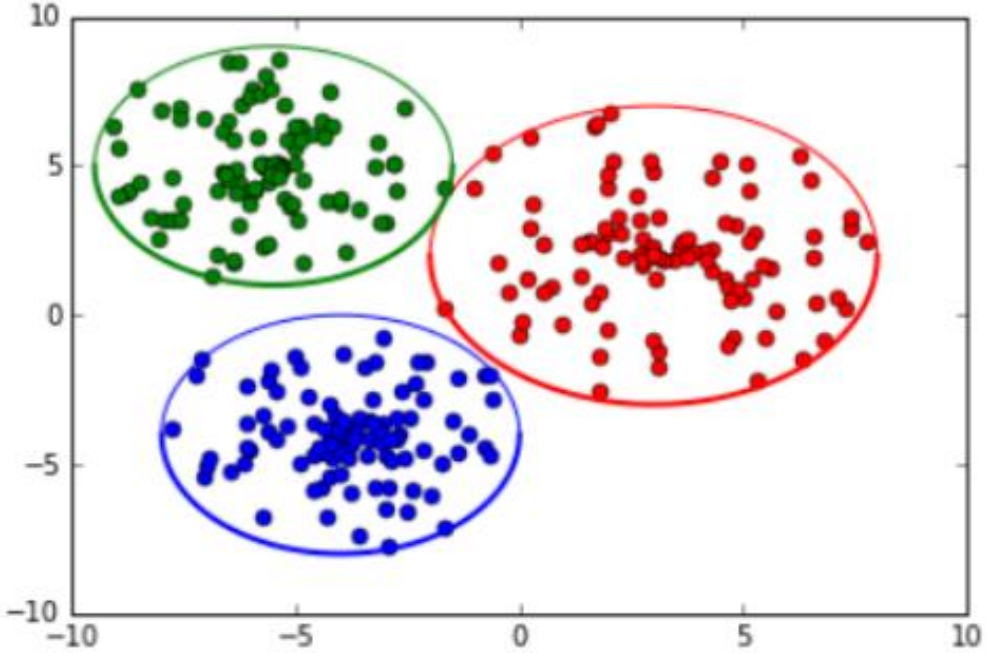
Ing. Biomedico - MSc. Automatización y Control Industrial

jprestrepo@correo.iue.edu.co

Institución Universitaria de Envigado

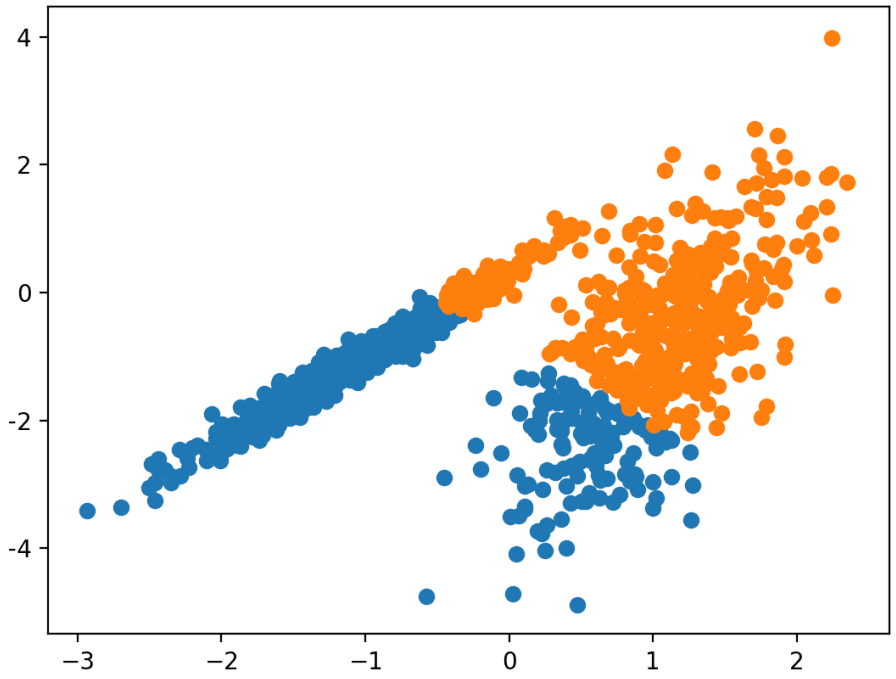
Unsupervised Learning

En el aprendizaje no supervisado no se etiquetan los datos de antemano, sino que dejamos que el algoritmo concluya. Uno de los algoritmos más comunes, y quizás el más simple de comprender es el algoritmo de clustering. Técnica que intenta separa los datos en subsets.



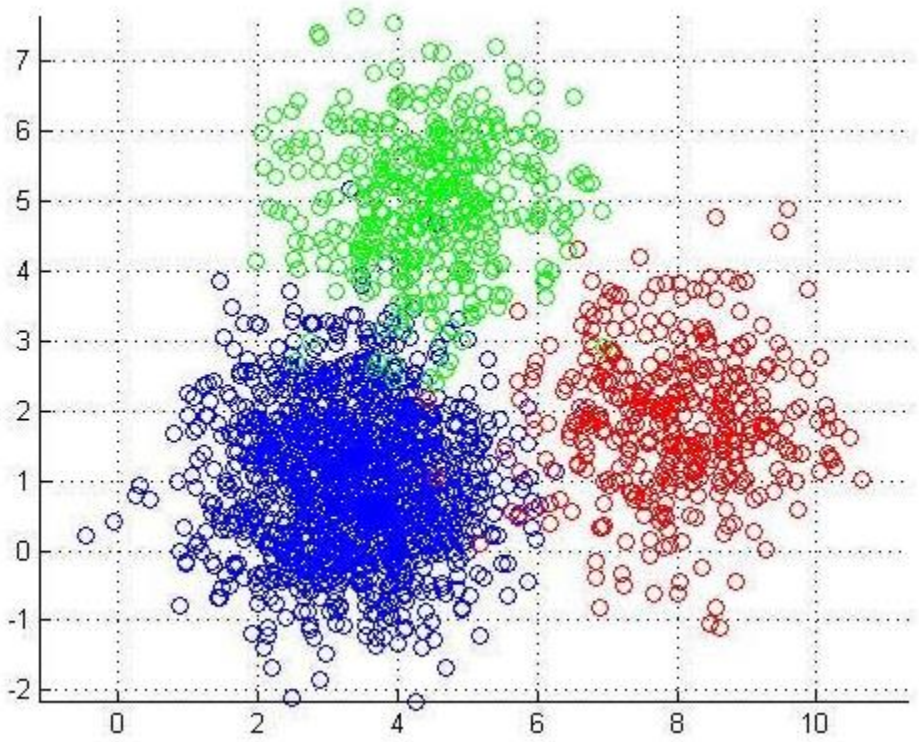
Unsupervised Learning

Si queremos clasificar corros como spam podemos hacerlo de manera no supervisada. Así, le preguntaremos al algoritmo que ubique cada muestra en uno o dos grupos separados (o clusters). El algoritmo combina las muestras en base a la similitud entre estas muestras (alta y baja).



Unsupervised Learning

Dentro de las técnicas descriptivas de Machine Learning basadas en análisis estadístico –utilizado para el análisis de datos en entornos Big Data–, encontramos el clustering, cuyo objetivo es formar grupos cerrados y homogéneos a partir de un conjunto de elementos que tienen diferentes características o propiedades, pero que comparten ciertas similitudes.

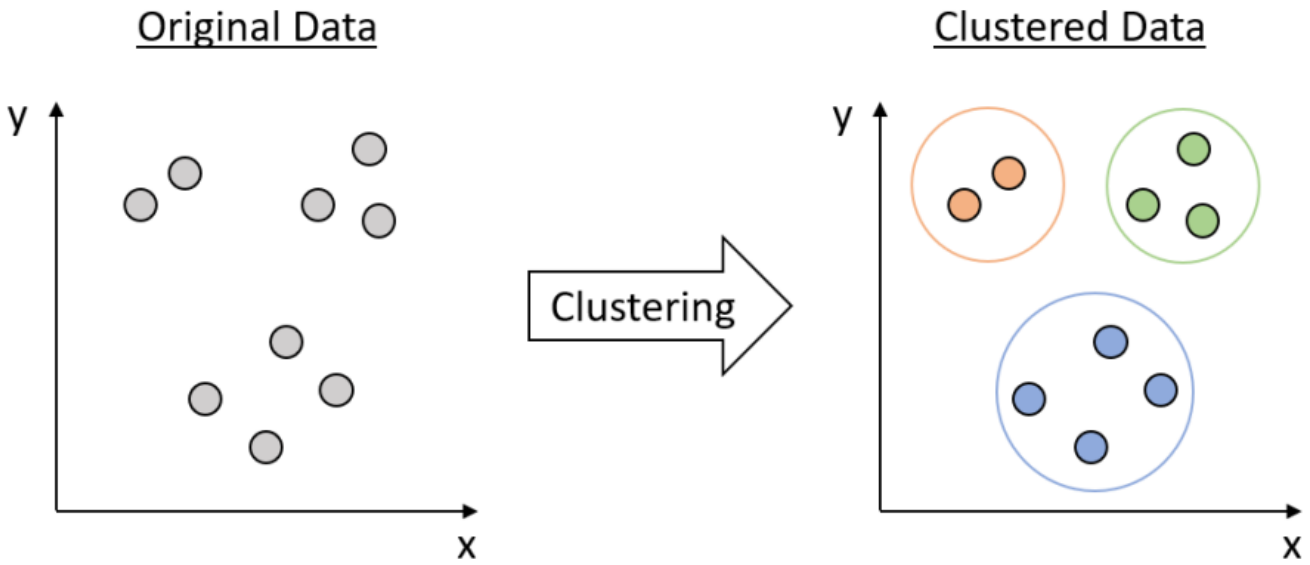


Unsupervised Learning

Algunos ejemplos de uso de las técnicas de clustering son:

- Segmentación de clientes en grupos.
- Determinar los distintos patrones climáticos de una región.
- Agrupar artículos o noticias por temas.
- Descubrir zonas con elevadas tasas de criminalidad.

K-means Clustering



K-means Clustering

SL también utilizada técnicas UL, aunque distintas del clustering. Por ejemplo, en Natural language processing (NL), o también en modelos generativas (opuestos a discriminatorio) como son las Generative Adversarial Networks (GAN).

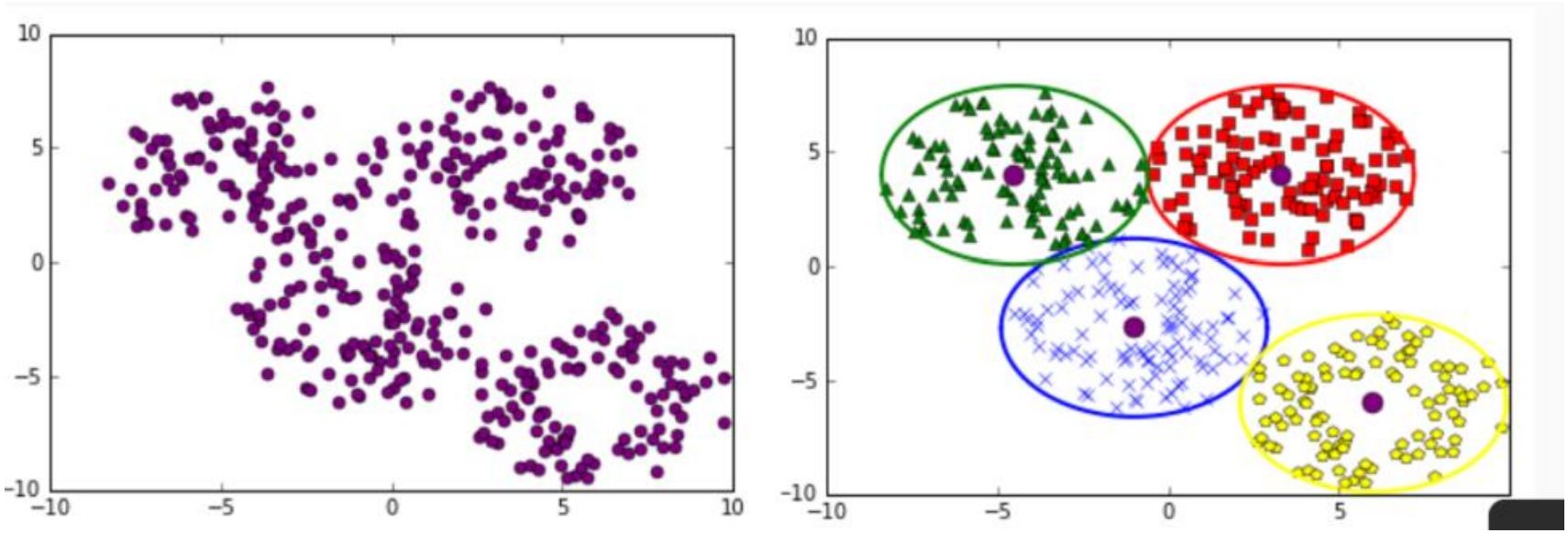
K-means Clustering

K-means es un algoritmo de clustering que agrupa los conjuntos de datos en k distintos clusters, y también es utilizado en muchos problemas prácticos.
Por ejemplo, digamos que una tienda que reparte pizza dese abrir cuatro nuevos locales en la ciudad, y ellos necesitan elegir la ubicación para estos sitios, así:

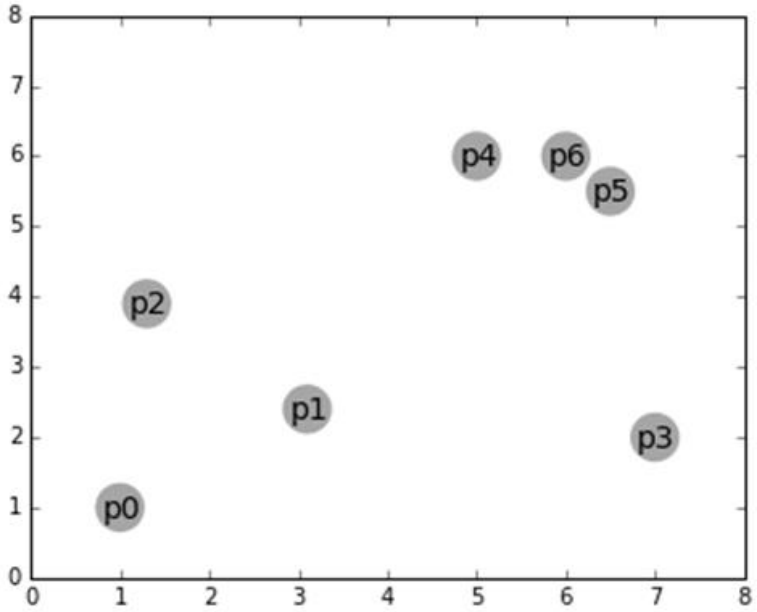
1. Encontrar la ubicación de donde las pizzas son ordenadas, por los consumidores habituales. Estos serán las entradas data.
2. Elegir cuatro puntos donde los sitios serán ubicados.
3. Usando k-means clustering, identificar las 4 mejores ubicaciones para minimizar la distancia a las entregas.

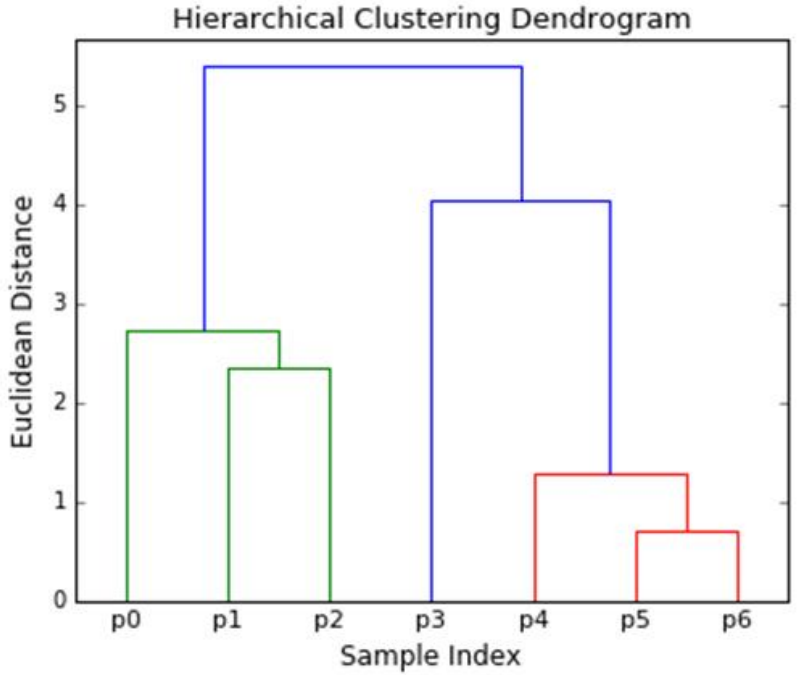
https://www.naftaliharris.com/blog/visualizing-k-means-clustering/

K-means Clustering

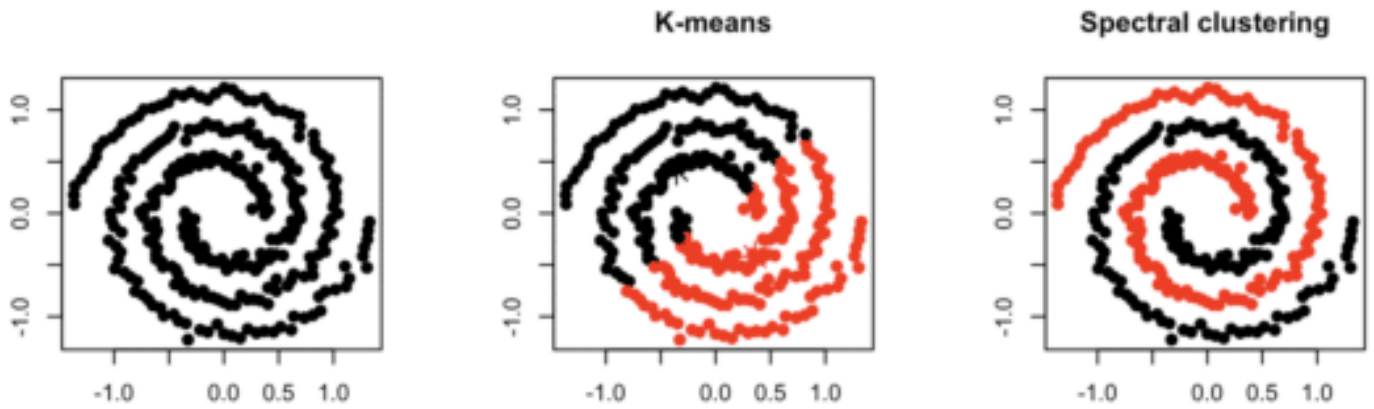


Métodos Jerárquicos





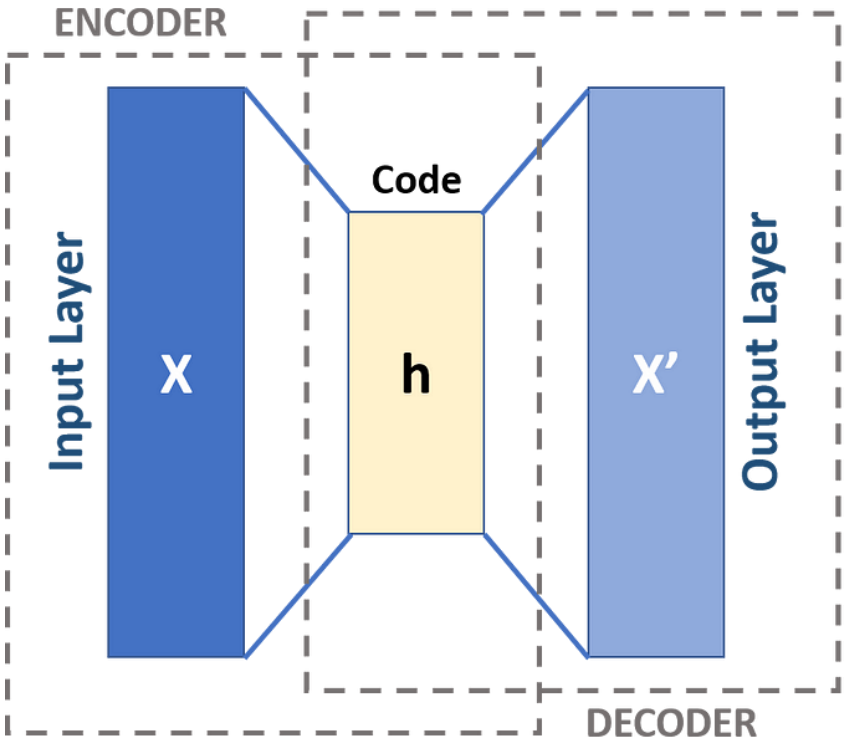
Métodos Particionales: Spectral Clustering



Evaluación de resultados

- Internos: Usa información interna del proceso de clustering, como varianza intra y entre clusters
- Externos: Compara los resultados con resultados conocidos (etiquetas)
- Relativos: comparación con sí mismo o métodos similares

Autoencoders + Clustering



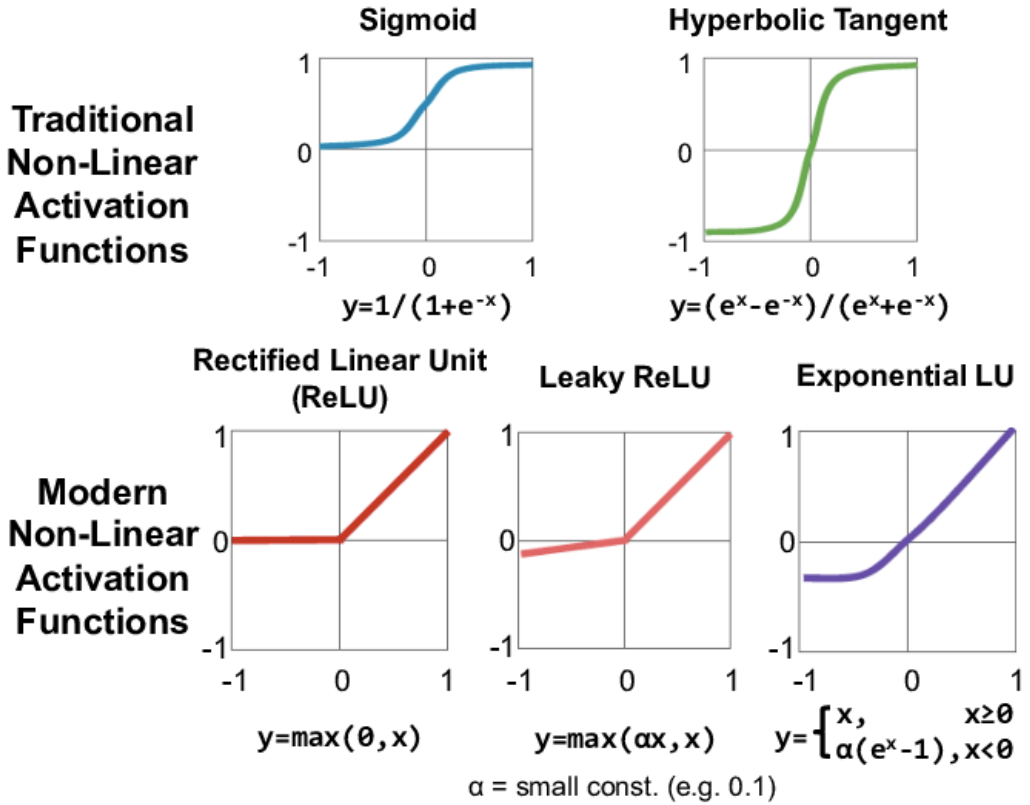
https://github.com/ZeyadZanaty/image-clustering/blob/master/cifar_image_clustering.ipynb

Función de activación

A la salida de la neurona, puede existir, un filtro, función limitadora o umbral, que modifica el valor resultado o impone un límite que se debe sobrepasar para poder proseguir a otra neurona. Esta función se conoce como función de activación. Una función de activación es, por tanto, una función que transmite la información generada por la combinación lineal de los pesos y las entradas, es decir son la manera de transmitir la información por las conexiones de salida.

$$\vec{v} = a_1\vec{v_1} + a_2\vec{v_2} + \cdots + a_n\vec{v_n}$$

Función de activación



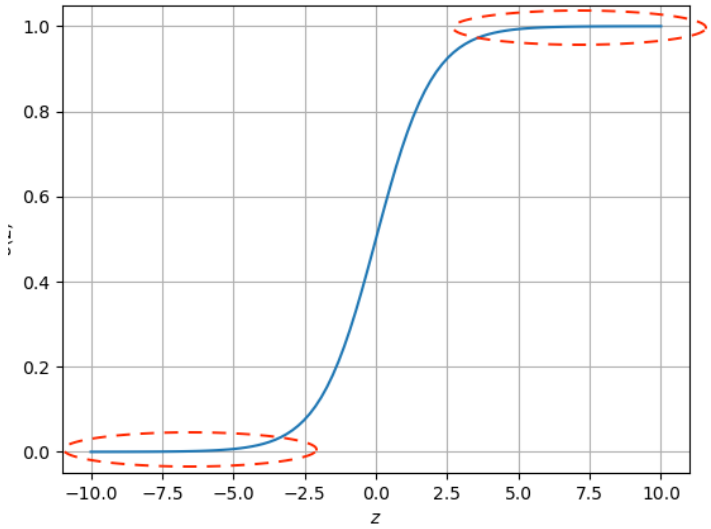
https://ignaciogavilan.com/catalogo-de-componentes-de-redes-neuronales-ii-funciones-de-activacion/

Función de activación

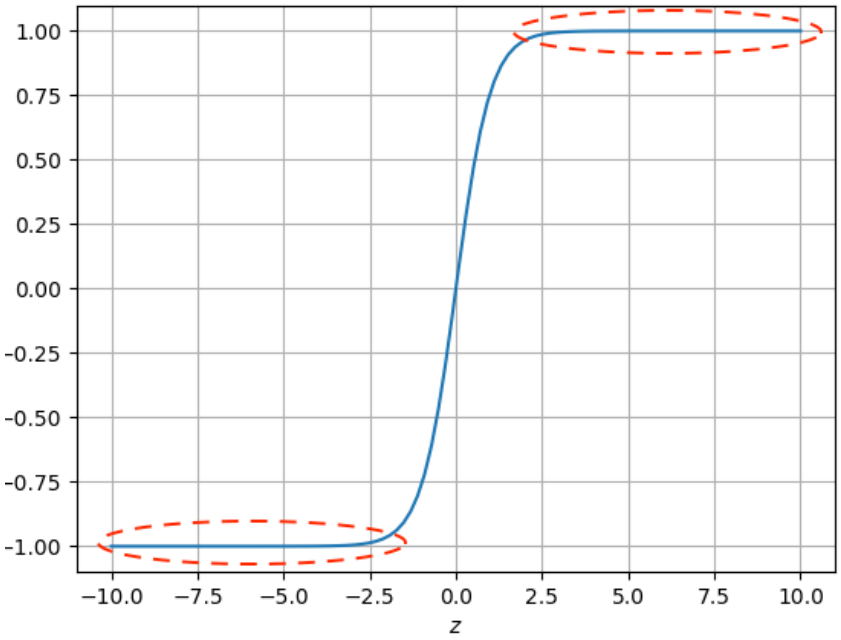
La función sigmoidal (σ)

En la actualidad esta función de activación tiene un uso limitado, y realmente su principal aplicación es la clasificación binaria.

Lo anterior se debe al problema de saturación, la función se satura a 1 cuando la entrada (z) es muy alta, y a 0 cuando es muy baja. Esto hace que durante el entrenamiento usando el método del gradiente descendente, los gradientes calculados sean muy pequeños, dificultando así la convergencia del algoritmo.



Función de activación

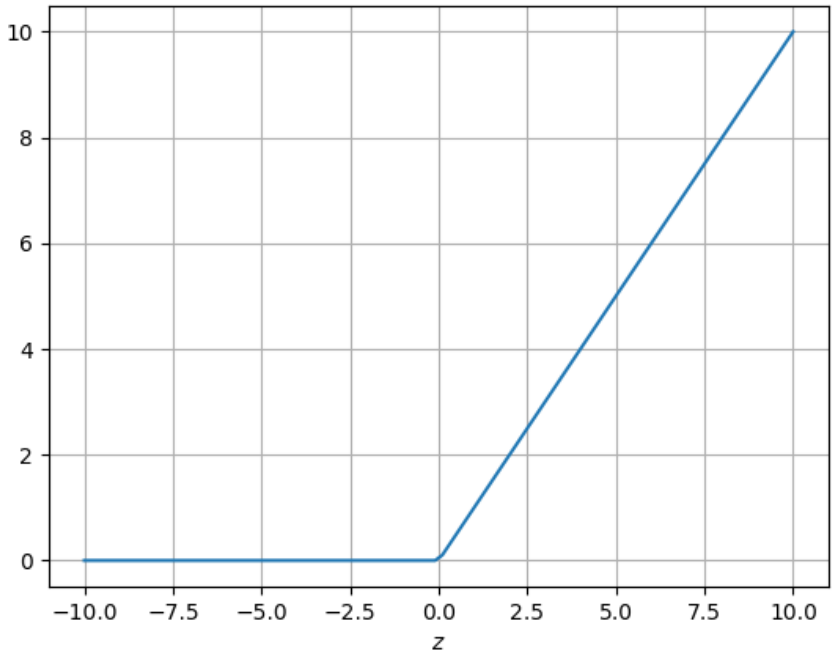


La función tangente hiperbólica (tanh)

En este caso, la función también sufre del problema de saturación, pero ofrece la ventaja de tener una salida simétrica lo cual facilita el proceso de entrenamiento.

A pesar de lo anterior es evidente que el problema de la saturación también limita la aplicabilidad de esta función de activación en Redes Neuronales.

Función de activación



La función ReLU

La función ReLU simplemente rectifica los datos (z) negativos y los vuelve cero a la salida. Las entradas con valores positivos no sufren modificación alguna a la salida

Esta función generará una salida igual a cero cuando la entrada (z) sea negativa, y una salida igual a la entrada cuando dicha esta última sea positiva.

Función de activación

La función de activación ReLU se ha convertido en la más usada en los modelos Deep Learning durante los últimos años, lo cual se debe principalmente a:

- La no existencia de saturación, como sí ocurre en las funciones sigmoidal y tanh. Lo anterior hace que el algoritmo del gradiente descendente converja mucho más rápidamente, facilitando así el entrenamiento.
- Es más fácil de implementar computacionalmente en comparación con las otras dos funciones, que requieren el cálculo de funciones matemáticas más complejas como la exponencial.

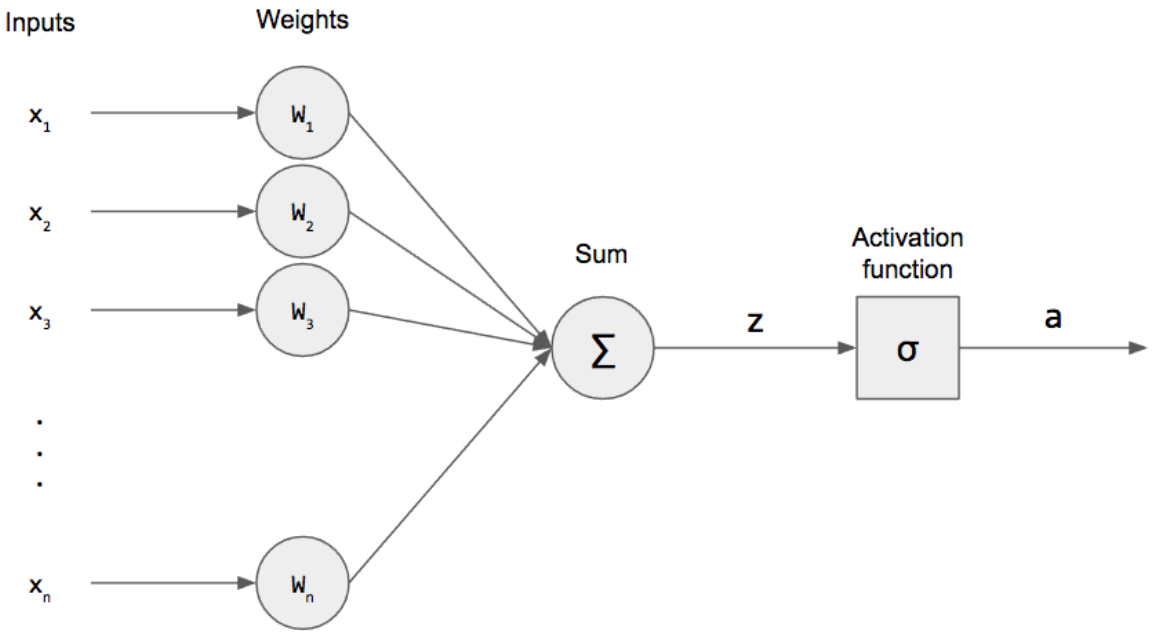
Data set

https://www.cs.toronto.edu/~kriz/cifar.html
https://es.wikipedia.org/wiki/Base_de_datos_MNIST
https://playground.tensorflow.org/#activation=tanh&batchSize=10&dataset=circle&regDataset=reg-plane&learningRate=0.03&regularizationRate=0&noise=0&networkShape=4,2&seed=0.30566&showTestData=false&discretize=false&percTrainData=50&x=true&y=true&xTimesY=false&xSquared=false&ySquared=false&cosX=false&sinX=false&cosY=false&sinY=false&collectStats=false&problem=classification&initZero=false&hideText=false

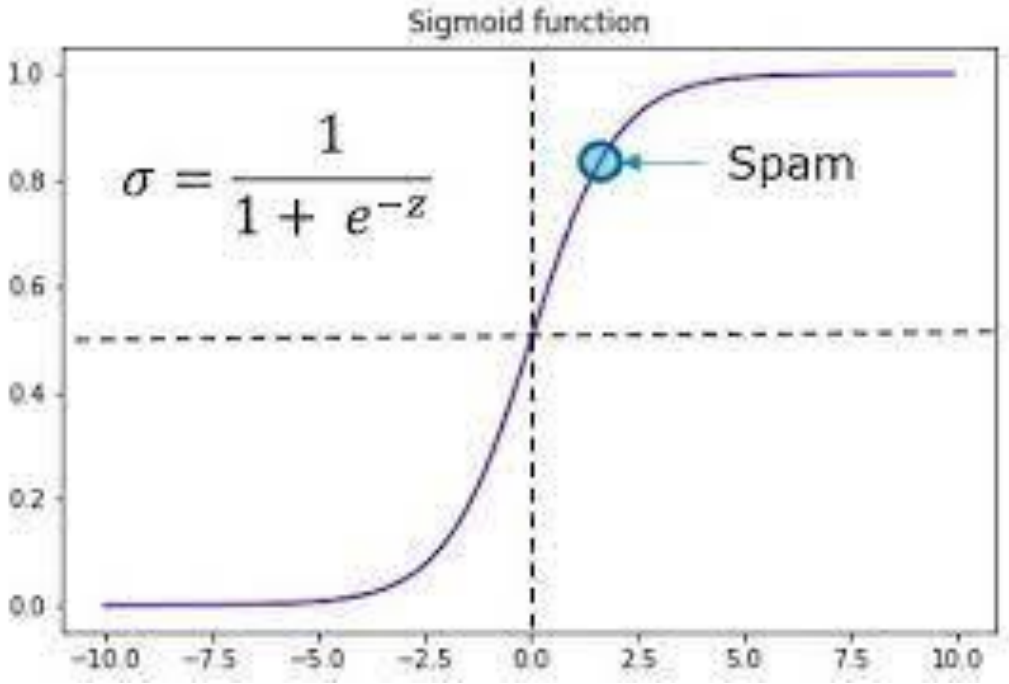
Linear and Logistic regression

Supongamos que tenemos una casa de $100\ m^2$, construida 25 años atrás, con 3 baños y 2 pisos, y se encuentra en un lugar de la ciudad con una valoración 7 (de 1 a 10). Como dato conocido, esta casa tiene un costo de $100000. Cuando deseamos parametrizar acá, lo que buscamos es una función $f(x) = 1000000$, lo que en términos de regresión lineal consiste en encontrar los pesos $\vec{w} = (w_1 + w_2 \ldots + w_n)$ de forma que $\vec{x} * \vec{w} = x_1 * w_1 + x_2 * w_2 \ldots x_n * w_n$. Si tuviéramos 5000 casas, tendríamos que repetir este proceso para todas las casas de forma de encontrar el valor $\vec{w}$ que se encuentra cercano a todas estas casas en términos de clase (precio, en este caso). La diferencia entre la regresión lineal y la logística es que la salida para esta última corresponde a la función logística $\sigma(\vec{x}, \vec{w})$. La función logística tiene definido su intervalo de salida en el rango (0,1).

Linear and Logistic regression



Linear and Logistic regression

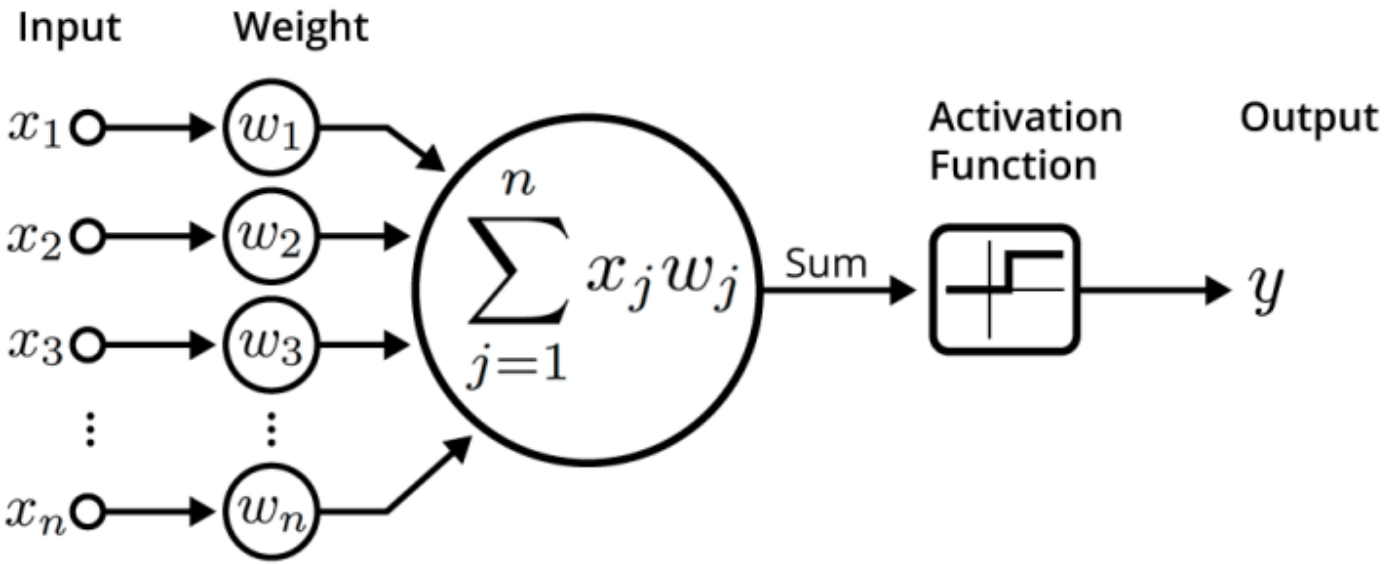


¿Qué es una red neuronal?

Podemos describir una red neuronal como un modelo matemático para procesar información. Si bien es una buena forma para describir casi cualquier algoritmo de ML, daremos un significado específico en el contexto de redes neuronales. Una red neuronal no es un programa fijo, sino más bien un modelo, un sistema que procesa información –o inputs–. Entre las características destacan:

- El proceso de información ocurre sobre objetos simples, llamados neuronas.
- Las neuronas están conectadas para intercambiar señales entre ellas.
- Cada neurona tiene un estado interno, que es determinado por toda la información recibida desde otras neuronas.
- Cada neurona tiene una función de activación distinta que es calculada sobre su estado, y determina su salida.

Una neurona

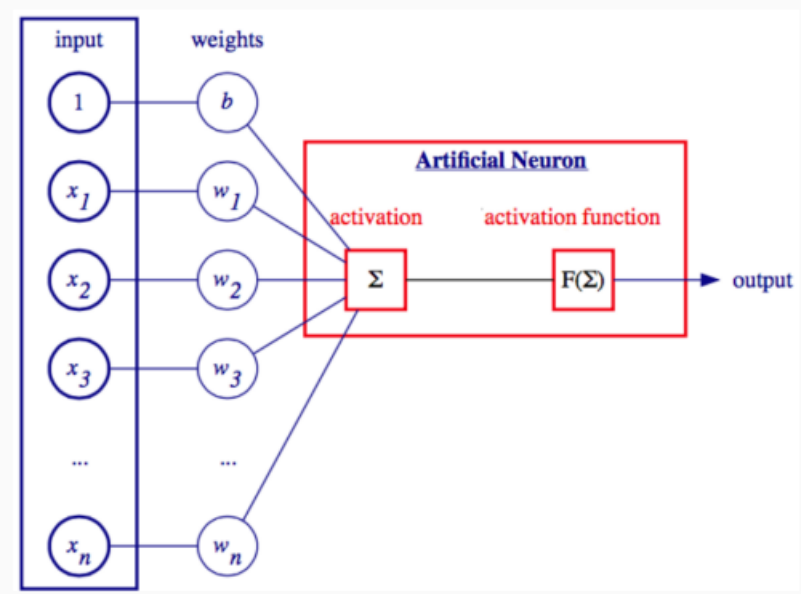

Input
Weight
x_1
x_2
x_3
x_n
w_1
w_2
w_3
w_n
$\sum_{j=1}^{n} x_j w_j$
Sum
Activation Function
Output
y

Una neurona

$$y = f\left(\sum_i x_i \omega_i + b\right)$$

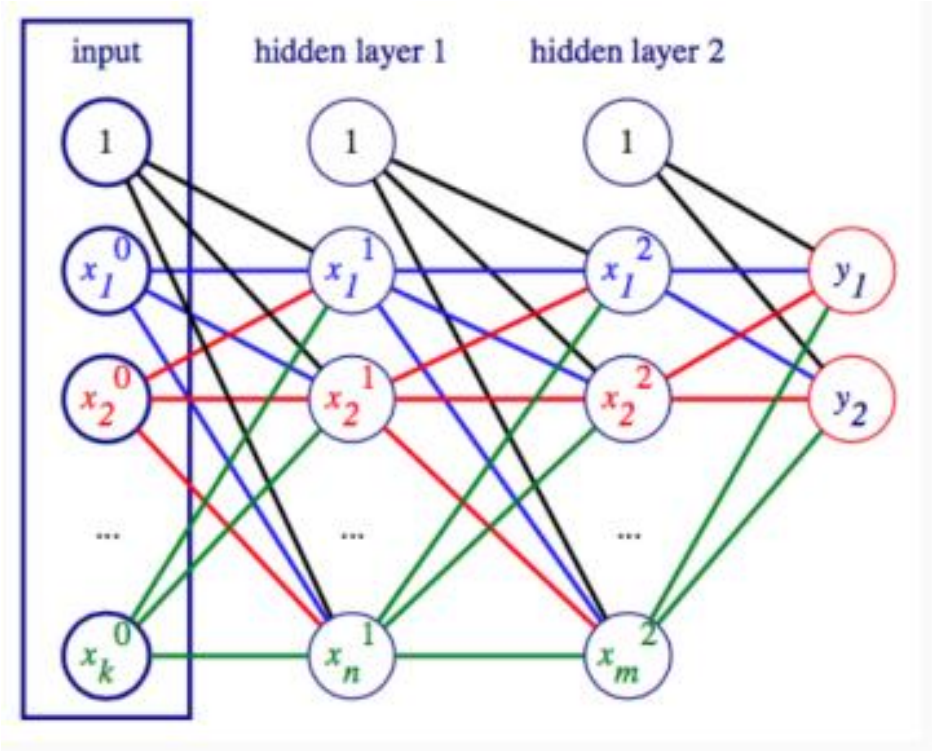
Una neurona y las capas

$$y = f\left(\sum_i x_i \omega_i + b\right)$$



RRNN de Multicapas

No existe nada que nos prohíba introducir más capas en el sistema, entre el input y el output. Así estas capas que podemos introducir son conocidas como hidden layers.



RNN de Multicapas