Apuntes 04 de abril, 2024

*Semana 8, I Semestre

Kenny Vega Obando
Tecnológico de Costa Rica
Cartago, Costa Rica
2019162050

I. Redes Neuronles - Repaso de clase pasada

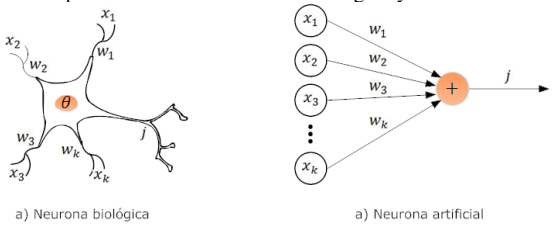
A. El Perceptron y Redes Neuronales

Los perceptrones proporcionaron un marco inicial para el aprendizaje automático, pero se descubrió que tenían limitaciones significativas. Uno de los problemas más significativos era su incapacidad para trabajar con funciones no lineales y unos requerimientos computacionales muy avanzados para la época, lo que causo que por varios años no se lograran avances significativos en el área este periodo se denomina ”Invierno de la IA”.

Para solucionar este problema, se desarrollaron perceptrones multicapa, esto consiste en múltiples capas de neuronas interconectadas, lo que les permite aprender representaciones jerárquicas de datos y resolver problemas que fueran no lineales. Las redes neuronales han demostrado ser extremadamente efectivas en una variedad de tareas de aprendizaje automático, incluyendo reconocimiento de imágenes, procesamiento de lenguaje natural, y juegos de estrategia como el ajedrez.

Este modelo se asemeja a nuestras neuronas, pues comparten diversas similitudes entre sí, fue una inspiración biológica como se observa en la figura 1.

Fig. 1. Comparación entre una neurona biológica y una neurona artificial



Estos sistemas de procesamiento de datos utilizan algoritmos informáticos para aprender, interpretar y clasificar datos sensoriales. Tratan de replican el funcionamiento de las neuronas en el cerebro, procesando datos a través de funciones de activación para generar resultados. Aprenden ajustando pesos en el algoritmo para minimizar la función de coste, lo que les permite realizar tareas como personalizar recomendaciones en comercio electrónico, procesar lenguaje natural en chatbots, predecir resultados de eventos y facilitar la conducción autónoma, entre otras aplicaciones.

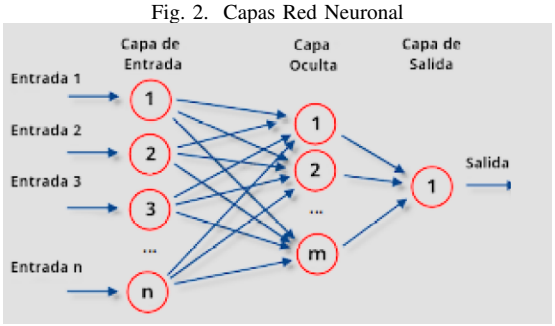
Las redes neuronales pueden tener varias capas, cada una con una función específica:

Capa de Entrada: Esta capa recibe los datos de entrada y los transmite a la red.

Capas Ocultas: Estas capas procesan la información recibida de la capa de entrada o de otras capas ocultas. Cada neurona en una capa oculta toma la salida de las neuronas de la capa anterior, realiza ciertas operaciones en ella y pasa la salida a la siguiente capa, esto permite que la red neuronal aprenda características abstractas y complejas de los datos.

Capa de Salida: Produce los resultados finales de la red neuronal después de procesar la información a través de las capas ocultas. La estructura y el número de neuronas en esta capa dependen del tipo de problema que se esté abordando.

Fig. 2. Capas Red Neuronal



La cantidad de capas necesarias nunca está definida, se usan las necesarias para resolver el problema, siempre que los recursos no sean un problema, a mayor cantidad de capas y neuronas mayor será el rendimiento obtenido pero mayor será el costo para entrenar el modelo.

B. Maldición de dimensionalidad

Se refiere a los desafíos y limitaciones que surgen al trabajar con conjuntos de datos de alta dimensionalidad. Se manifiesta cuando el rendimiento de los algoritmos de aprendizaje automático disminuye o se vuelve ineficiente a medida que aumenta el número de dimensiones de los datos, lo cual se vuelve evidente si lo pensamos, ya que no es lo mismo buscar un objeto en cuartos de una casa que buscar ese mismo objeto en un edificio con diferentes pisos y apartamentos por cada nivel.

Algunos de los problemas asociados a esto incluyen:

1) **Dispersión de los datos:** A medida que aumenta el número de dimensiones, la cantidad de datos necesarios para densificar el espacio aumenta exponencialmente, como resultado, los datos pueden estar muy dispersos, lo que dificulta encontrar patrones significativos.
2) **Mayor complejidad computacional:** El procesamiento y el almacenamiento de datos de alta dimensionalidad requieren una mayor cantidad de recursos computacionales, lo que puede hacer que los algoritmos sean más lentos y menos eficientes.
3) **Sobreajuste (overfitting):** Con conjuntos de datos de alta dimensionalidad, existe un mayor riesgo de sobreajuste, donde el modelo se adapta demasiado a los datos de entrenamiento y pierde su capacidad de generalización a nuevos datos.
4) **Mayor cantidad de características irrelevantes o redundantes:** En conjuntos de datos de alta dimensionalidad, es más probable que algunas características sean irrelevantes o redundantes para el problema que se está abordando. Esto puede dificultar la identificación de las características más importantes y relevantes para el modelo.

Comportamiento Jerárquico: El comportamiento jerárquico se refiera a entrenar las capas de la red para aprender representaciones de datos cada vez más abstractas y complejas. Las primeras capas aprenderán características simples y de bajo nivel, como bordes o texturas, mientras que las capas más profundas pueden aprender características más abstractas y de alto nivel, como formas o conceptos, este proceso de aprendizaje jerárquico permite a la red modelar relaciones complejas entre las características de entrada y producir salidas más sofisticadas y precisas.

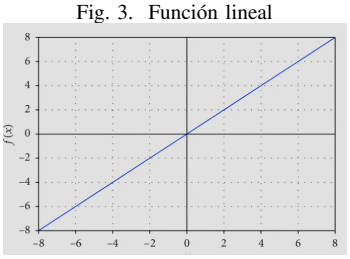
II. Funciones de activación

Las funciones de activación se colocan inmediatamente después de nuestra regresión lineal, también conocida como preactivación, y cumplen varias funciones importantes en una red neuronal:

- Permiten agregar comportamientos específicos a la red neuronal, como la no linealidad, lo que le da capacidad para aprender y modelar relaciones complejas en los datos.
- Definen un rango de salida para las neuronas, lo que permite limitar los valores que pueden ser emitidos por la red, facilitando así la interpretación y el procesamiento de los resultados.
- Controlan la activación de las neuronas, determinando cuándo y en qué medida una neurona debe ”activarse” y transmitir su señal a las neuronas de la capa siguiente.

Función lineal: La función más básica que se utiliza comúnmente es la función lineal, sin embargo, esta función no resulta muy útil en la mayoría de los casos, ya que no permite definir un rango de salida y, debido a que su derivada siempre es constante, el gradiente de la red durante el entrenamiento no depende de la entrada, lo que significa que esta no aprendera nada de los dato.

Fig. 3. Función lineal



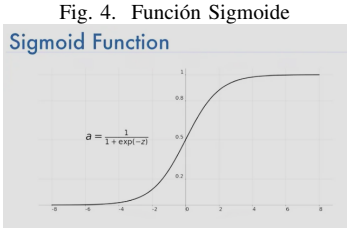
Sigmoide:

- Sus valores van del 0 al 1.
- Siempre es positiva debido a su definición.
- Está acotada dentro de su rango de valores.
- Es una función estrictamente creciente, a medida que nos desplazamos en el eje x de izquierda a derecha, siempre encontraremos un valor mayor que el de la izquierda.

A pesar de sus ventajas, la función sigmoide puede presentar un problema con su pendiente, ya que puede ser muy cercana a cero en regiones lejanas de cero, lo que puede provocar que:

- El entrenamiento puede ser muy lento o en el peor de los casos este no converger
- Vanishing gradient, lo cual es la principal por la que el entramiento se estanque.
- Gradiente pequeño al final de la función.

Fig. 4. Función Sigmoide

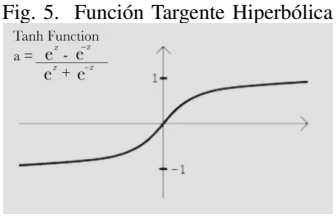


Targente Hiperbólica:

- Similar a la función sigmoide conservando su propiedad de estrictamente creciente, pero con un rango de salida entre -1 y 1.
- Puede tener valores negativos como positivos
- Es una función simétrica alrededor del origen
- Usada en modelos de lenguaje (LSTM)

La tangente hiperbólica comparte similitudes con la función sigmoide y esta también puede sufrir problemas similares, como la desaparición del gradiente en regiones lejanas de cero. Esto puede afectar el proceso de entrenamiento al igual que el caso anterior.

Fig. 5. Función Targente Hiperbólica



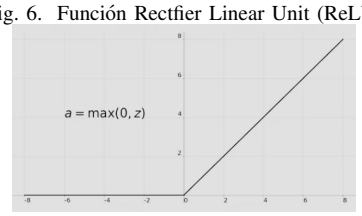
Rectfier Linear Unit (ReLU):

- Se encuentra acotada únicamente con valores debajo del cero
- Estrictamente creciente
- Es una de las funciones mas utilizadas en el deep learning y capas ocultas
- Es la más rápida de computar y normalmente se empieza usando esta
- Función: g(x) = max(x,0)

Un problema de la función ReLU es que no es derivable en el punto x = 0, aunque esta propiedad podría plantear problemas en términos de cálculo de gradientes durante el entrenamiento de la red neuronal, en la práctica no suele ser un inconveniente significativo. Esto se debe a que las probabilidades de que la entrada sea exactamente 0 son mínimas en la mayoría de los conjuntos de datos reales. Por lo tanto, en la práctica, se considera aceptable retornar un valor de 1 o 0 para la derivada. Esta simplificación no suele generar conflictos y permite que la función ReLU se utilice eficazmente en redes neuronales en una variedad de aplicaciones.

Otra característica a tener en cuenta es que puede generarse neuronas muertas, lo que significa que la neurona no está contribuyendo efectivamente al proceso de aprendizaje de la red neuronal y ya no cambiara.

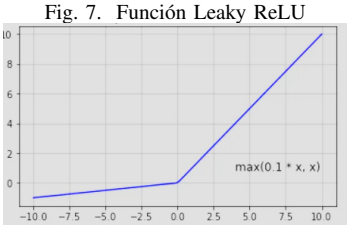
Fig. 6. Función Rectfier Linear Unit (ReLU)



Leaky ReLU:

- RTrata de resolver el problema de las neuronas muertas: Multiplicando x por una constante para resolver así el problema de las neuronas inactivas.
- Permite un pequeño gradiente para valores negativos, lo que evita las ”neuronas muertas” y permite un flujo de información incluso para entradas negativas.
- Aunque es un intento de solución, puede no ser la mejor opción en todos los casos, ya que el valor de la constante de inclinación podría ser subóptimo para algunos problemas.

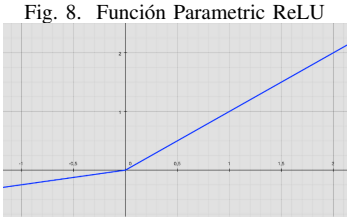
Fig. 7. Función Leaky ReLU



Parametric ReLU:

- Al igual que en el caso anterior, el Parametric ReLU viene a resolver el problema de las neuronas muertas.
- Permite aprender un parámetro que controla la pendiente de la parte negativa de la función de activación, lo que permite que la señal fluya.
- El parámetro se aprende durante el entrenamiento de la red, junto con otros parámetros, lo que le otorga mayor flexibilidad y adaptabilidad.
- Esta flexibilidad puede conducir a un mejor rendimiento en comparación con otras funciones de activación.

Fig. 8. Función Parametric ReLU

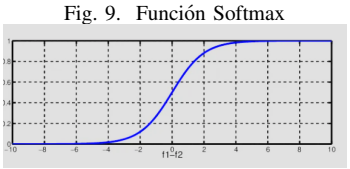


Softmax:

- Función utilizada para convertir las salidas de las neuronas en probabilidades.
- Principalmente empleada en problemas de clasificación multiclase, donde asigna una probabilidad a cada clase.
- Garantiza que la suma de todas las probabilidades sea igual a 1, lo que la convierte en una distribución de probabilidad válida.
- La función de pérdida comúnmente utilizada con Softmax es la pérdida de entropía cruzada (cross-entropy loss), que compara las distribuciones de probabilidad predichas con las distribuciones reales de las clases.

Se usa en la parte final y hace que la clase con mayor probabilidad sea la seleccionada.

Fig. 9. Función Softmax



A. *Logits:*

Los logits son los valores de entrada antes de una función de activación, y son utilizados en la etapa final de la red neuronal para calcular las probabilidades de pertenencia a cada clase en un problema de clasificación.

REFERENCES

[1] PERCEPTRON, QUÉ ES Y CÓMO SE USA EN MACHINE LEARNING. [Online]. https://ciberseguridad.com/guias/nuevas-tecnologias/machine-learning/perceptron/#Origen

[2] Jesús. (2022, Abril 27). Componentes de una Red Neuronal. [Online]. https://www.datasmarts.net/componentes-de-una-red-neuronal/

[3] ¿QUÉ SON LAS REDES NEURONALES ARTIFICIALES Y CÓMO FUNCIONAN?. [Online]. https://ciberseguridad.com/guias/nuevas-tecnologias/inteligencia-artificial/redes-neuronales-artificiales/#Funcion_de_paso_binario