Universidad Autónoma de Aguascalientes

Centro de Ciencias Básicas

Departamento de Ciencias de la Computación

Optativa Profesionalizante II: Machine Learning y Deep Learning

10° "A"

Actividad 5: Función ReLU

Docente: Dr. Francisco Javier Luna Rosas

Alumno: Joel Alejandro Espinoza Sánchez (211800)

Fecha de Entrega: Aguascalientes, Ags., 20 de marzo del 2023.

Actividad 5: Función ReLU

La Función SoftMax

El alumno deberá elaborar un documento `*.pdf` donde implemente la función ReLU.

El documento debe contener lo siguiente:

- Análisis de la función ReLU.
- Implementación de la función ReLU en Python o en el lenguaje de programación de preferencia.
- Evaluación de la función ReLU en Python o en el lenguaje de programación de preferencia.

El alumno deberá subir a la plataforma el archivo (`*.pdf`) y un documento auto-reproducible (`*.html`) que deberá contener:

- Portada.
- Evidencias de la actividad.
- Conclusiones.
- Referencias (formato APA).

La función de activación ReLU (Rectified Linear Unit) es una de las más utilizadas en redes neuronales artificiales. Esta función es simple pero efectiva, ya que básicamente toma un valor de entrada y devuelve cero si es negativo o el valor de entrada si es positivo. Es decir, la función

ReLU activa solo los valores positivos y desactiva los valores negativos, lo que la hace muy útil para la clasificación binaria. Además, es muy rápida y fácil de calcular, lo que la hace ideal para ser utilizada en redes profundas con grandes cantidades de datos.

La función ReLU se utiliza en la capa de activación de una red neuronal para introducir la no-linealidad en el modelo y para ayudar a evitar el problema de la desaparición del gradiente que ocurre con funciones de activación sigmoideas. Al ser una función no lineal, permite que las redes neuronales se ajusten a relaciones complejas y no lineales entre los datos de entrada y salida. En general, la función ReLU es muy popular en la comunidad de aprendizaje automático debido a su simplicidad, rapidez y eficacia.

Su implementación en código se realiza a continuación, primeramente importando las librerías necesarias:

In [5]:
```
import numpy as np
import matplotlib.pyplot as plt
```

Después se define la función ReLU:

In [10]:
```
def relu(x):
    return np.maximum(0, x)
```

Podemos visualizar los resultados de esta función de la siguiente forma:

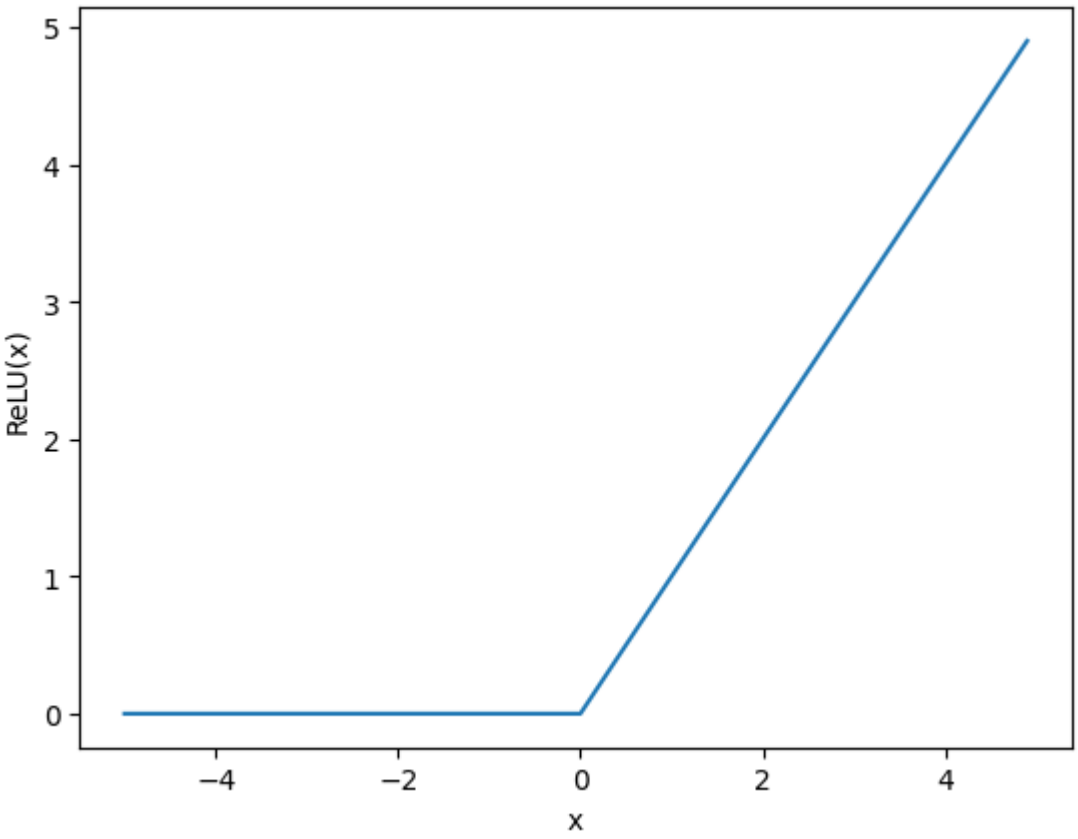
In [11]:
```
# Definimos el intervalo en el que queremos graficar la función SoftMax
x = np.arange(-5, 5, 0.1)

# Calculamos los valores de la función SoftMax para el intervalo dado
y = relu(x)

# Graficamos la función SoftMax
plt.plot(x, y)

# Etiquetamos los ejes
plt.xlabel('x')
plt.ylabel('ReLU(x)')

# Mostramos la gráfica
plt.show()
```



Conclusiones

Es interesante e importante poder implementar las bases de estos temas para entenderlos en un futuro, pues, posteriormente no basta con sólo importar librerías que realicen el trabajo pesado, ya que, implementar manualmente las funciones matemáticas en las que recaen estos algoritmos nos enseña a qué hay detrás del algoritmo, cómo funciona y poder comprender realmente qué está ocurriendo como la base de una red neuronal y la forma en la que ésta aprende realizando el descenso del gradiente. Es muy útil la implementación de estos algoritmos en estas tareas para las futuras tareas de la materia y aplicaciones de Machine Learning en la vida personal.

Referencias

- Anónimo (s.f.) “Red neuronal artificial”. Obtenido de Wikipedia: https://es.wikipedia.org/wiki/Red_neuronal_artificial.
- Data Scientest (2021) “Perceptrón. ¿Qué es y para qué sirve?”. Obtenido de Data Scientest: https://datascientest.com/es/perceptron-que-es-y-para-que-sirve.
- Luna, F. (2023) “El Modelo de McCulloch – Pitts”. Apuntes de ICI 10°.