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 4: Función SoftMax

Docente: Dr. Francisco Javier Luna Rosas

Alumno: Joel Alejandro Espinoza Sánchez (211800)

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

Actividad 4: Función SoftMax

La Función SoftMax

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

El documento debe contener lo siguiente:

- Análisis de la función SoftMax.
- Implementación de la función SoftMax en Python o en el lenguaje de programación de preferencia.
- Evaluación de la función SoftMax 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 SoftMax es una función matemática que se utiliza comúnmente en aprendizaje automático y redes neuronales para normalizar un conjunto de valores en un rango de 0 a 1.

Esta función es especialmente útil en problemas de clasificación multiclase, donde se necesita convertir un conjunto de valores en una distribución de probabilidad que sume 1. La función SoftMax toma un vector de números y devuelve otro vector del mismo tamaño, donde cada elemento del vector de salida se calcula dividiendo el exponente del elemento correspondiente en el vector de entrada por la suma de los exponentes de todos los elementos del vector de entrada. En otras palabras, la función SoftMax transforma una entrada en un vector de valores normalizados que representan la probabilidad de cada clase.

La función SoftMax se define matemáticamente como:

$$SoftMax(x_i) = \frac{e^{x_i}}{sum(e^{x_j})}$$

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

In [3]:
```
import numpy as np
import matplotlib.pyplot as plt
```

Después se define la función SoftMax:

In [4]:
```
def softmax(x):
    return np.exp(x) / np.sum(np.exp(x))
```

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

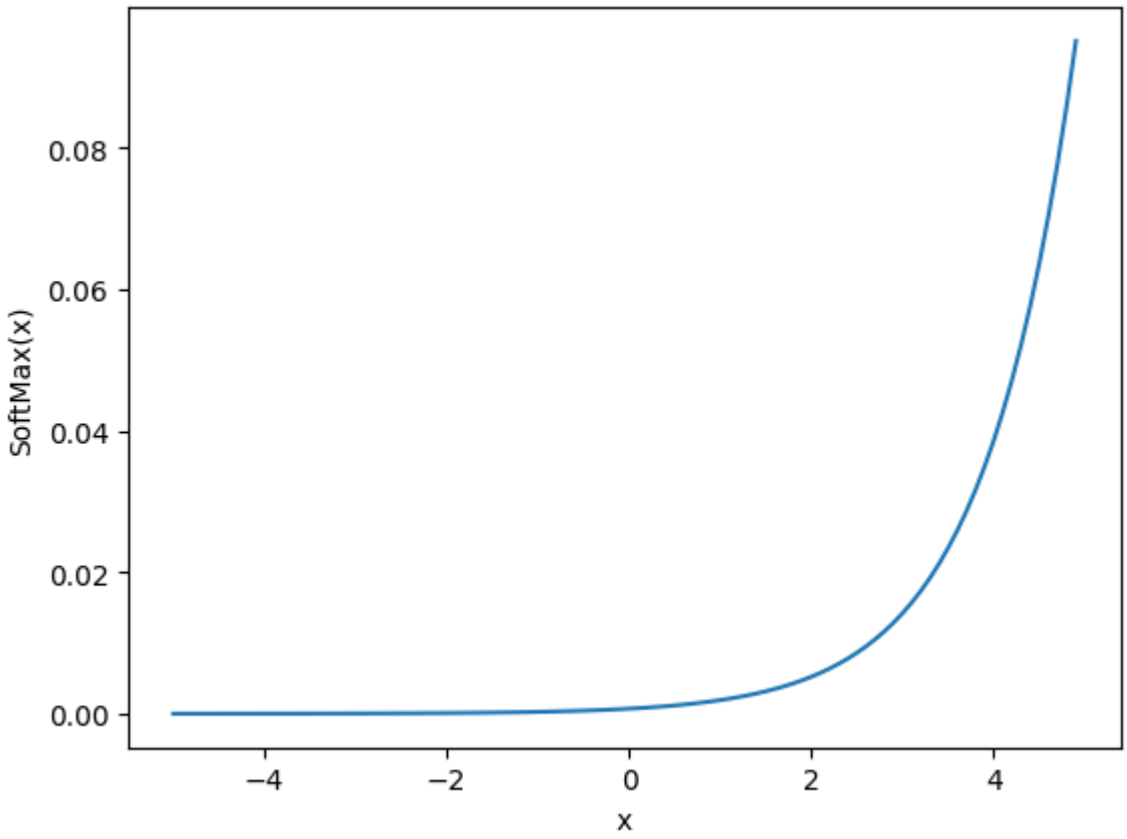
In [5]:
```
# Definimos el intervalo en el que queremos graficar la función SoftMax
x = np.arange(-5, 5, 0.1)

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

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

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