SSP de inteligencia Artificial II

Práctica 1. Ejercicio 2

Diego Arturo Herrera Álvarez | 217450735

Profesor: Oliva Navarro, Diego Alberto

NRC: 145999

Clave: I7041

Sección: D05

índice

Contents

1

Introducción

Métodos de partición

Cuando se entrena a un perceptrón, se utiliza una serie de datos con sus respectivos resultados para la realización de el algoritmo de aprendizaje. Por lo regular estos datos se encuentran en un sólo apartado, por ejemplo un archivo csv, base de datos, archivo de excel, etc.

Para esto hay que dividir dichos datos en entrenamiento y prueba, es decir que parte de esos datos se utilizarán para que el perceptrón entrene y los demás se utilizarán para probarlo y verificar los resultados obtenidos. Para esta actividad se idearon 5 métdods para la separación de en este caso, un archivo csv.

2

Primero entrenamiento, luego pruebas.

Este método es básicamente primero sacar los datos de entrenamiento y luego los de prueba, por ejemplo, si se requiere un 80 % de datos de entrenamiento, se tomará ese 80% en el orden como están los datos en el archivo, sin ninguna modificación. Los datos restantes serán desinados a pruebas.

```
numEntrenamiento = int(len(datos) * int(porcentajeEntrenamiento) / 100)
num    (variable) case: str    mEntrenamiento

if case == '1':
    newDatos = datos[0:numEntrenamiento]
    newY = y[0:numEntrenamiento]
    datosPrueba = datos[numEntrenamiento: len(datos)]
    resultadosPrueba = y[numEntrenamiento: len(datos)]

    x = np.array(newDatos)
```

1- Ajuste de pesos:

3

Primero pruebas, luego entrenamiento

Este método es el mismo que el anterior, sólo cambia el orden. En el ejemplo anterios hay un 20 % de pruebas, entonces en el orden del archivo se tomará ese 20% y los datos restantes serán destinados a entrenamiento.

```
elif case == '2':
    datosPrueba = datos[0: numPrueba]
    resultadosPrueba = y[0:numPrueba]
    newDatos = datos[numPrueba:len(datos)]
    newY = y[numPrueba:len(datos)]

    x = np.array(newDatos)
```

4

Al azar

Aquí primero se toman los índices de los datos y se revuelven aleatoriamente y después se asignan primero los datos de entrenamiento y en segundo los datos de pruebas. Gracias a este método, tenemos datos aleatorios de el archivo csv.

```
elif case == '3':
    indices = list(range(len(datos)))
    random.shuffle(indices)

    # Seleccionar los índices para entrenamiento y prueba
    indices_entrenamiento = indices[:numEntrenamiento]
    indices_prueba = indices[numEntrenamiento:]

    # Crear conjuntos de entrenamiento y prueba usando los índices
    newDatos = [datos[i] for i in indices_entrenamiento]
    newY = [y[i] for i in indices_entrenamiento]

    datosPrueba = [datos[i] for i in indices_prueba]
    resultadosPrueba = [y[i] for i in indices_prueba]

    x = np.array(newDatos)
```

5

Por bloques

Aquí se dividirá sobre 10 el número de datos de entrenamiento y de prueba. Por ejemplo, en caso de tener 200 datos y querer un 80% de pruebas, es decir 160 datos, entonces se harán bloques de 16 datos de entrenamiento y 4 datos de prueba. Después de realizar esto, en orden del archivo csv se toman 16 datos de entrenamiento, después lo 4 de prueba y se repite el proceso hasta terminar los bloques.

```
elif case == '5':
    bloqueEntrenamiento = numEntrenamiento/10
    bloquePrueba = numPrueba/10


    iBloqueEntrenamiento = 0
    iBloquePrueba = -1

    for indice in range(len(datos)):
        if(iBloqueEntrenamiento < bloqueEntrenamiento):
            newDatos.append(datos[indice])
            newY.append(datos[indice])
            iBloqueEntrenamiento = iBloqueEntrenamiento + 1

        elif(iBloqueEntrenamiento == bloqueEntrenamiento):
            iBloquePrueba = 0
            iBloqueEntrenamiento = iBloqueEntrenamiento + 1

        if(iBloquePrueba>=0 and iBloquePrueba<bloquePrueba):
            datosPrueba.append(datos[indice])
            resultadosPrueba.append(datos[indice])
            iBloquePrueba = iBloquePrueba + 1

        elif(iBloquePrueba>= bloquePrueba):
            iBloqueEntrenamiento = 0
            iBloquePrueba = -1

    x = np.array(newDatos)
```

6

Pruebas

Técnica 1:

La técnica 1 será probada con el data set 1 (spheres2d10.csv), estos son los datos ingresados:

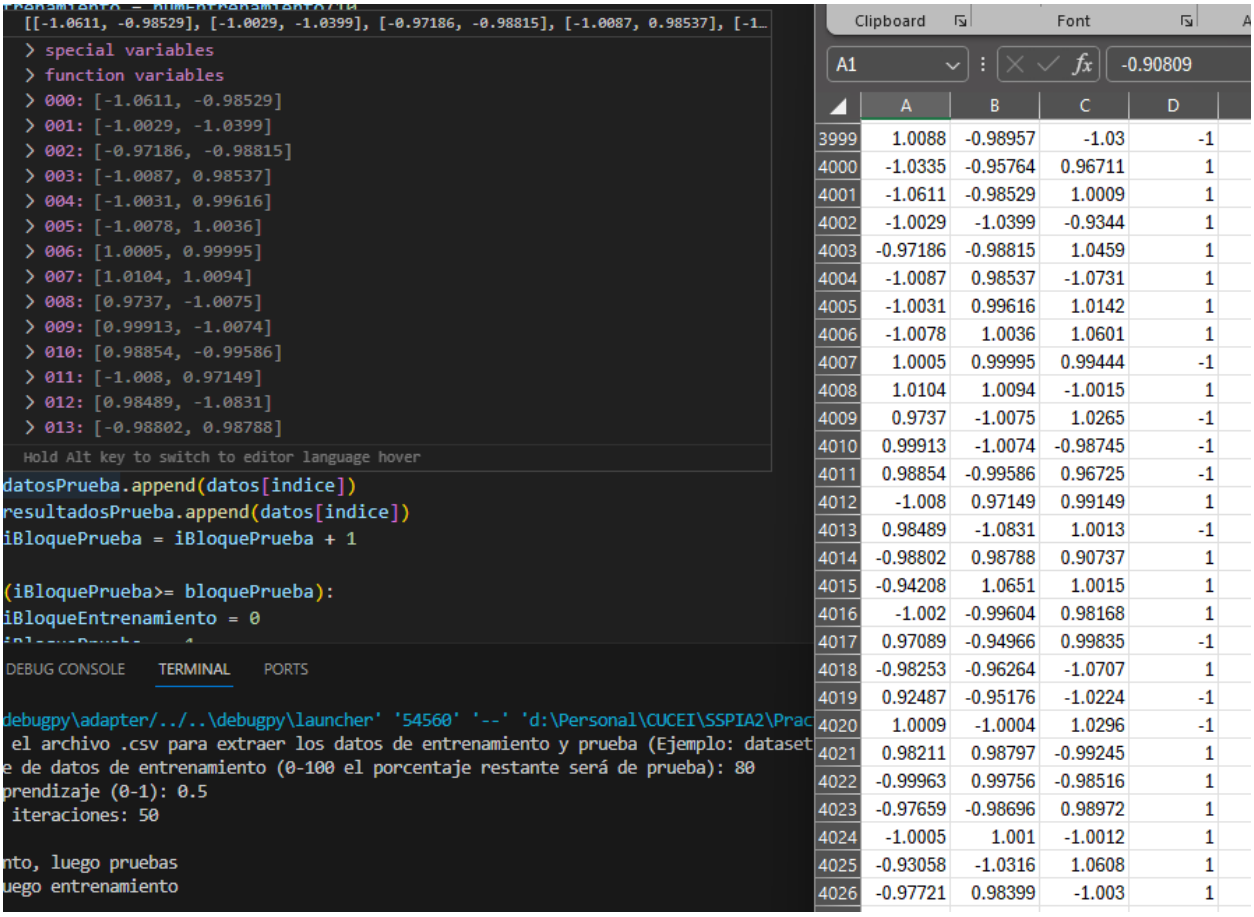
```
Ingrese el nombre de el archivo .csv para extraer los datos de entrenamiento y prueba (Ejemplo: dataset1.csv): spheres2d10.csv
Ingrese el porcentaje de datos de entrenamiento (0-100 el porcentaje restante será de prueba): 80
Ingrese la tasa de aprendizaje (0-1): 0.5
Ingrese el número de iteraciones: 50

1)Primero entrenamiento, luego pruebas
2)Primero pruebas, luego entrenamiento
3)Al azar
4)Uno y uno
5)Por bloques
Ingrese el modo de partición: 1
```

Este archivo tiene 5000 elementos, por ende el 80% sería de 4000.

Datos de entrenamiento después de la partición:

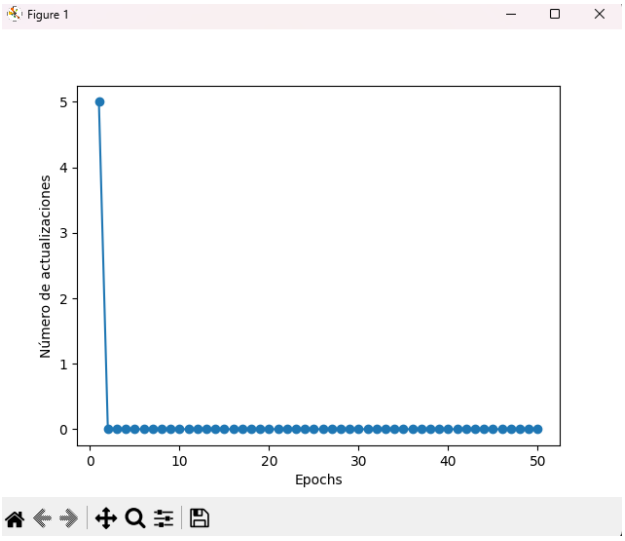
```
newDatos.append(datos[indice])
  [[-0.90809, 1.0026], [1.0399, 0.94302], [-0.98341, 0.95374], [-0.9966, 0.92313], [1.0, …
  > 0088: [-0.98067, 1.0069]
  > 0089: [1.0114, -0.9937]
  > 0090: [0.98184, -1.005]
  > 0091: [-1.0055, 1.0394]
  > 0092: [-0.99055, -1.0111]
  > 0093: [1.0089, 0.9691]
  > 0094: [1.0116, 1.0027]
  > 0095: [0.95897, -0.98455]
  > 0096: [-0.99831, -1.0254]
  > 0097: [0.98887, 1.0227]
  > 0098: [-1.0116, 1.0113]
  > 0099: [1.0, -1.0078]
  > more: ...
    len(): 4000
```

Datos de prueba después de la partición:



Aquí se puede ver que los datos de prueba empiezan a partir de el elemento 4001.

Resultados:



Podemos ver que el error se fue a cero muy rápido.

Para calcular la precisión y la sensibilidad se obtienen varias variables, la cantidad de positivos en los casos de prueba, la cantidad de aciertos y el número de casos de prueba.

```
# Hacer predicciones usando el perceptrón entrenado
for dato in datosPrueba:
    prediccion = ppn.predice(dato)
    resultado = resultadosPrueba[i]
    i = i+1

    print(f'Entrada: {dato}, Predicción: {prediccion}, Resultado: {resultado}')

     # Calcular aciertos
    if prediccion == resultado:
        aciertos += 1

# Calcular precisión y sensibilidad
precision = aciertos / len(datosPrueba)
sensibilidad = aciertos / positivos_reales

print(f'Precisión: {precision:.2f}')
print(f'Sensibilidad: {sensibilidad:.2f}')
```

Resultados:

```
Entrada: [1.0215, 1.0691, 0.99915], Predicción: -1, Resultado: -1.0
Entrada: [0.95583, 0.97188, -1.0246], Predicción: 1, Resultado: 1.0
Entrada: [0.99861, 1.001, -0.9872], Predicción: 1, Resultado: 1.0
Entrada: [-0.99623, -1.0029, -0.94489], Predicción: 1, Resultado: 1.0
Precisión: 1.00
Sensibilidad: 4.13
```

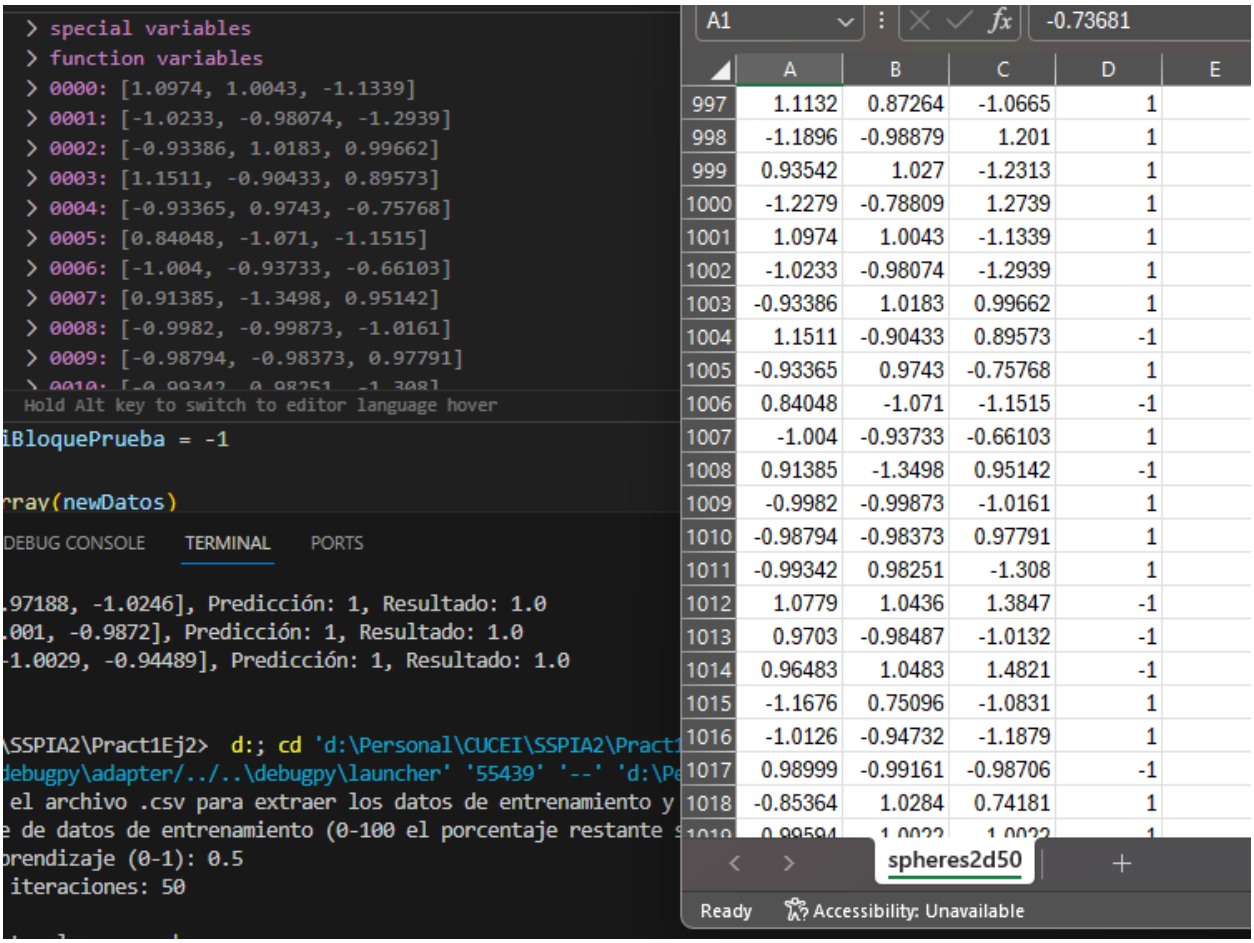
Tenemos una precisión completa y una sensibilidad alta.

Técnica 2:

Entradas:

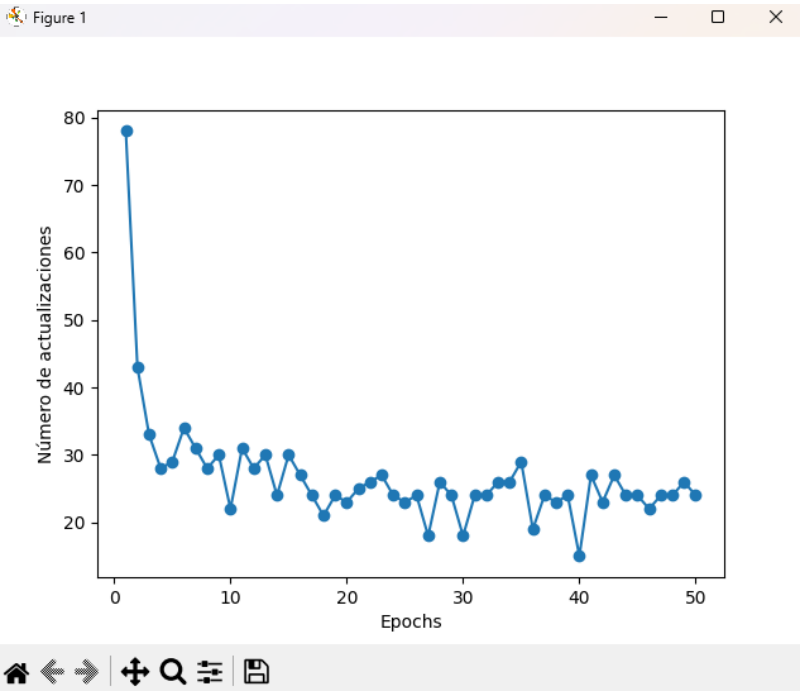
```
Ingrese el nombre de el archivo .csv para extraer los datos de entrenamiento y prueba (Ejemplo: dataset1.csv): spheres2d50.csv
Ingrese el porcentaje de datos de entrenamiento (0-100 el porcentaje restante será de prueba): 80
Ingrese la tasa de aprendizaje (0-1): 0.5
Ingrese el número de iteraciones: 50

1)Primero entrenamiento, luego pruebas
2)Primero pruebas, luego entrenamiento
3)Al azar
4)Uno y uno
5)Por bloques
Ingrese el modo de partición: 2
```

División:



Aquí vemos que caso contrario a la técnica 1 ahora los datos de entrenamiento empiezan desde el dato 1001.

Resultados:



Vemos que el error se mantiene bajo pero no se pone en 0 en ningún momento. La precisión sigue siendo de 1 pero la sensibilidad bajó n poco.

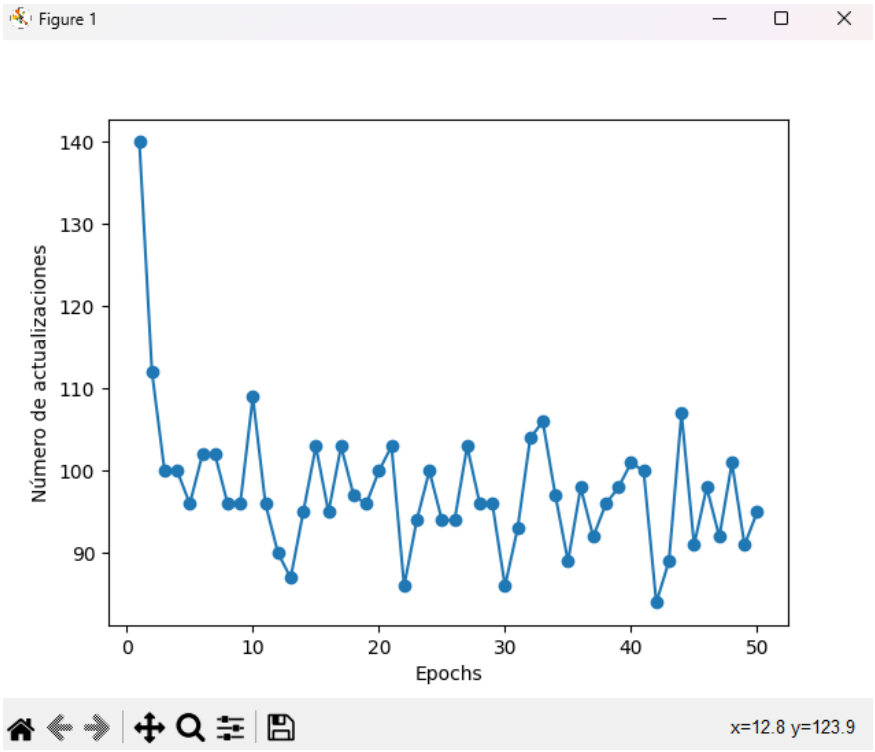
```
Entrada: [1.1132, 0.87264, -1.0665], Predicción: 1, Resultado: 1.0
Entrada: [-1.1896, -0.98879, 1.201], Predicción: 1, Resultado: 1.0
Entrada: [0.93542, 1.027, -1.2313], Predicción: 1, Resultado: 1.0
Entrada: [-1.2279, -0.78809, 1.2739], Predicción: 1, Resultado: 1.0
Precisión: 1.00
Sensibilidad: 3.53
```

Técnica 3:

Entradas:

```
Ingrese el nombre de el archivo .csv para extraer los datos de entrenamiento y prueba (Ejemplo: dataset1.csv): spheres2d70.csv
Ingrese el porcentaje de datos de entrenamiento (0-100 el porcentaje restante será de prueba): 80
Ingrese la tasa de aprendizaje (0-1): 0.5
Ingrese el número de iteraciones: 50

1)Primero entrenamiento, luego pruebas
2)Primero pruebas, luego entrenamiento
3)Al azar
4)Uno y uno
5)Por bloques
Ingrese el modo de partición: 3
```

Resultados:



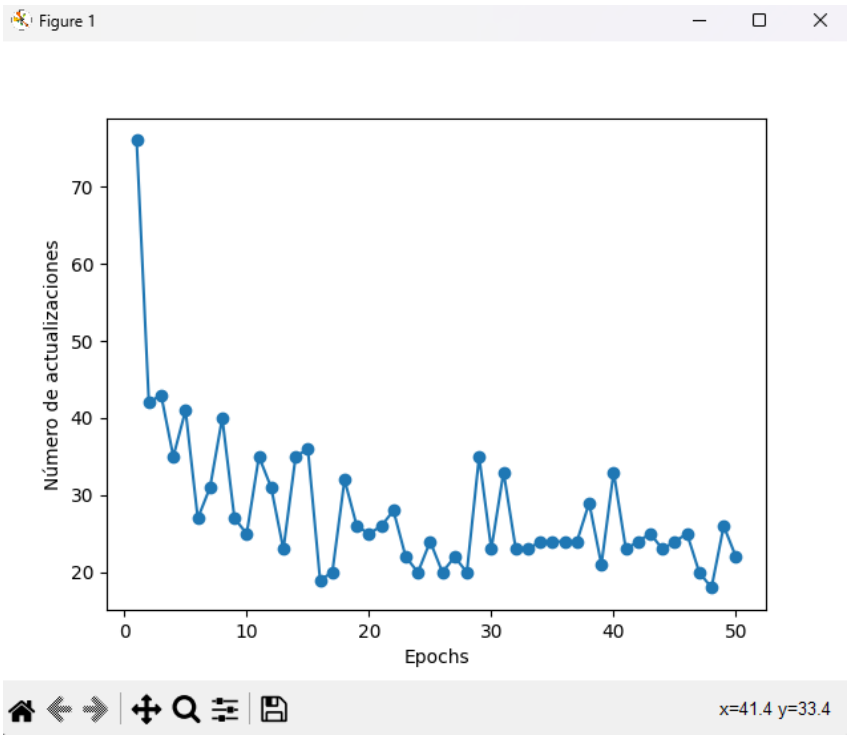

```
Entrada: [-1.4817, 1.3757, -1.1159], Predicción: 1, Resultado: 1.0
Entrada: [0.91259, 1.2434, 0.48593], Predicción: -1, Resultado: -1.0
Entrada: [-0.98319, 1.1261, -0.49204], Predicción: 1, Resultado: 1.0
Entrada: [-0.86893, 1.0314, -0.85223], Predicción: 1, Resultado: 1.0
Entrada: [-0.97114, 0.95925, -1.0552], Predicción: 1, Resultado: 1.0
Entrada: [-0.93033, 0.5938, 1.0293], Predicción: 1, Resultado: 1.0
Entrada: [1.2027, -1.026, 1.0134], Predicción: -1, Resultado: -1.0
Entrada: [0.80255, -0.96856, 1.66], Predicción: -1, Resultado: -1.0
Entrada: [1.3202, -1.1737, 1.1085], Predicción: -1, Resultado: -1.0
Precisión: 0.98
Sensibilidad: 3.54
```

Técnica 4:

Entradas:

```
Ingrese el nombre de el archivo .csv para extraer los datos de entrenamiento y prueba (Ejemplo: dataset1.csv): spheres2d50.csv
Ingrese el porcentaje de datos de entrenamiento (0-100 el porcentaje restante será de prueba): 80
Ingrese la tasa de aprendizaje (0-1): 0.5
Ingrese el número de iteraciones: 50

1)Primero entrenamiento, luego pruebas
2)Primero pruebas, luego entrenamiento
3)Al azar
4)Uno y uno
5)Por bloques
Ingrese el modo de partición: 4
```

Resultados:



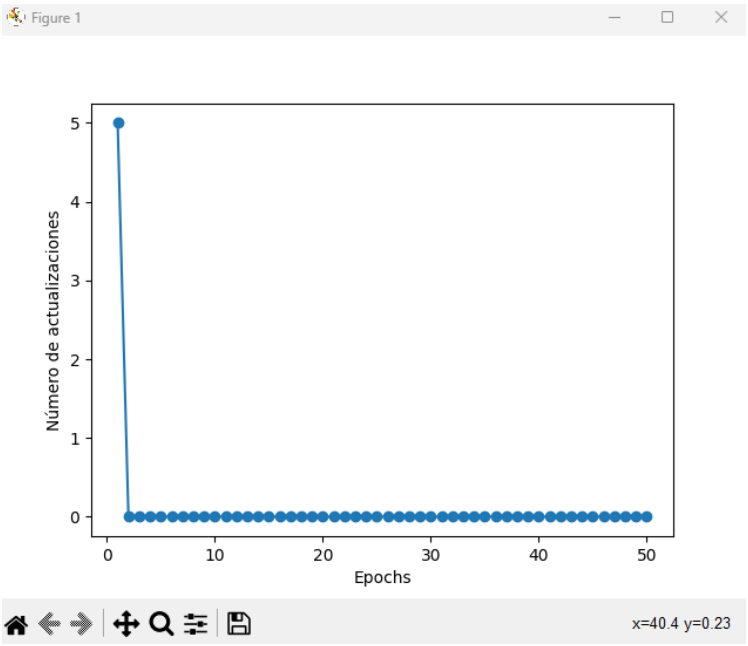

```
Entrada: [-1.1498, 1.0642, -0.74701], Predicción: 1, Resultado: 1.0
Entrada: [-0.82436, 1.0281, 0.79855], Predicción: 1, Resultado: 1.0
Entrada: [-1.0046, -1.0025, 1.1039], Predicción: 1, Resultado: 1.0
Entrada: [-1.1431, 0.76926, 1.0466], Predicción: 1, Resultado: 1.0
Entrada: [0.99216, -1.0143, 0.54184], Predicción: -1, Resultado: -1.0
Entrada: [1.0043, 0.99702, 0.95382], Predicción: -1, Resultado: -1.0
Entrada: [-1.0236, -0.99042, 1.0054], Predicción: 1, Resultado: 1.0
Entrada: [0.83091, 0.80439, -0.81739], Predicción: 1, Resultado: 1.0
Precisión: 1.00
Sensibilidad: 3.69
```

Técnica 5:

Entradas:

```
Ingrese el nombre de el archivo .csv para extraer los datos de entrenamiento y prueba (Ejemplo: dataset1.csv): spheres2d10.csv
Ingrese el porcentaje de datos de entrenamiento (0-100 el porcentaje restante será de prueba): 80
Ingrese la tasa de aprendizaje (0-1): 0.5
Ingrese el número de iteraciones: 50

1)Primero entrenamiento, luego pruebas
2)Primero pruebas, luego entrenamiento
3)Al azar
4)Uno y uno
5)Por bloques
Ingrese el modo de partición: 5
```

Resultados:




```
Entrada: [1.001, 0.99948, 1.0792], Predicción: -1, Resultado: -1.0
Entrada: [-0.98584, -0.98566, -0.91359], Predicción: 1, Resultado: 1.0
Entrada: [-1.0677, -1.0032, -0.95853], Predicción: 1, Resultado: 1.0
Entrada: [1.0215, 1.0691, 0.99915], Predicción: -1, Resultado: -1.0
Entrada: [0.95583, 0.97188, -1.0246], Predicción: 1, Resultado: 1.0
Entrada: [0.99861, 1.001, -0.9872], Predicción: 1, Resultado: 1.0
Entrada: [-0.99623, -1.0029, -0.94489], Predicción: 1, Resultado: 1.0
Precisión: 1.00
Sensibilidad: 4.29
```

7

Conclusión

Se puede apreciar una diferencia poco significante entre los métodos, por ejemplo, el método 1 y 5 que compartieron set de datos, se comportaron de forma muy similar. La técnica 2 y 4 también compartieron set de datos y puedo ver que la técnica 2 es un poco mejor que la técnica 4, ya que tiene el error más controlado y una sensibilidad más alta. Pero esto va a depender mucho de el acomodo de los datos en el archivo. Yo creo que si tuviera que elegir uno sería la técnica 1, ya que es la más fácil y lógica de usar, aunque habría que ver cómo se acomodan de inicio los datos, saber qué datos le convienen más al perceptrón.