Universidad de Guadalajara
Centro Universitario de Ciencias Exactas e Ingenierías
Seminario de Solución de Problemas de Inteligencia Artificial II
Dr. Diego Oliva
Depto. De Ciencias Computacionales
Practica 1
Perceptrón Simple y Multicapa

Ejercicio 2: dataset

Mercado Manzo Luis Alfonso

Código:212559704

Objetivo:

Ejercicio 2: Realizar un programa que permita generar un conjunto de particiones de entrenamiento

considerando un dataset. El programa debe permitir seleccionar la cantidad e particiones y el

porcentaje de patrones de entrenamiento y prueba. Para verificar su funcionamiento se debe realizar

lo siguiente:

1. Usar el archivo spheres1d10.csv que contiene datos generados en base a la Tabla 1. Estos

datos consideran alteraciones aleatorias (<10%), tal como se muestra en la Figura 1(a).

Usando el perceptrón simple, crear cinco particiones de entrenamiento usando 80% de los

datos y 20% para la generalización.

2. Considerando la Tabla 1, modificar el punto x=[-1, +1, -1] → yd = 1. Con esto se genera un

nuevo dataset. Los archivos spheres2d10.csv, spheres2d50.csv y spheres2d70.csv contienen

los datos perturbados en un 10%, 50% y 70% y se presentan en las Figuras 1 (b), (c), (d).

mediante el perceptrón simple realizar una clasificación con 10 particiones usando 80% de

los datos y 20% para la generalización.

Codigo:

```
# -*- coding: utf-8 -*-
"""
Created on Thu Oct 12 14:30:48 2023

@author: luis mercado
"""

import numpy as np
import pandas as pd
```

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Perceptron

# Lectura del archivo de datos
datos = pd.read_csv('spheres1d10.csv', header=None)

# Extracción de las entradas y las salidas de los datos
entradas = datos.iloc[:, :-1].values
salidas = datos.iloc[:, -1].values

# Definición de parámetros de particionamiento
num_particiones = 5  # Número de particiones
porcentaje_entrenamiento = 0.8  # Porcentaje de patrones de entrenamiento
porcentaje_prueba = 1 - porcentaje_entrenamiento  # Porcentaje de patrones de prueba

# Creación de las particiones de entrenamiento y prueba
particiones_entrenamiento = []
particiones_prueba = []

for _ in range(num_particiones):
    entradas_entrenamiento, entradas_prueba, salidas_entrenamiento, salidas_prueba =
train_test_split(
        entradas, salidas, train_size=porcentaje_entrenamiento, test_size=porcentaje_prueba
    )
    particiones_entrenamiento.append((entradas_entrenamiento, salidas_entrenamiento))
    particiones_prueba.append((entradas_prueba, salidas_prueba))

# Creación y entrenamiento del perceptrón para cada partición
for i in range(num_particiones):
```

```
entradas_entrenamiento, salidas_entrenamiento = particiones_entrenamiento[i]
perceptron = Perceptron()
perceptron.fit(entradas_entrenamiento, salidas_entrenamiento)
# Realiza las operaciones que desees con el perceptrón entrenado en cada partición
# ...

# Ejemplo: Mostrar el porcentaje de acierto en la partición de prueba
entradas_prueba, salidas_prueba = particiones_prueba[i]
salidas_predichas = perceptron.predict(entradas_prueba)
porcentaje_acierto = (salidas_prueba == salidas_predichas).mean()
print(f"Partición {i+1}: Porcentaje de acierto en la prueba: {porcentaje_acierto * 100}%")
```

resultados:

spheres1d10.csv

```
Partición 1: Porcentaje de acierto en la prueba: 81.0%
Partición 2: Porcentaje de acierto en la prueba: 78.5%
Partición 3: Porcentaje de acierto en la prueba: 74.0%
Partición 4: Porcentaje de acierto en la prueba: 87.0%
Partición 5: Porcentaje de acierto en la prueba: 54.50000000000001%
```

Spheres2d10.csv

```
Partición 1: Porcentaje de acierto en la prueba: 100.0%
Partición 2: Porcentaje de acierto en la prueba: 100.0%
Partición 3: Porcentaje de acierto en la prueba: 100.0%
Partición 4: Porcentaje de acierto en la prueba: 100.0%
Partición 5: Porcentaje de acierto en la prueba: 100.0%
```

Spheres2d50.csv

```
Partición 1: Porcentaje de acierto en la prueba: 99.1%
Partición 2: Porcentaje de acierto en la prueba: 99.2%
Partición 3: Porcentaje de acierto en la prueba: 99.1%
Partición 4: Porcentaje de acierto en la prueba: 99.6%
Partición 5: Porcentaje de acierto en la prueba: 99.4%
```

Spheres2d70.csv

```
Partición 1: Porcentaje de acierto en la prueba: 98.5%
Partición 2: Porcentaje de acierto en la prueba: 97.8%
Partición 3: Porcentaje de acierto en la prueba: 97.6%
```

```
Partición 4: Porcentaje de acierto en la prueba: 97.89999999999999%
Partición 5: Porcentaje de acierto en la prueba: 97.7%
```

Conclusión:

las particiones de entrenamiento, validación y prueba son esenciales en el desarrollo de modelos de aprendizaje automático para asegurarse de que el modelo sea capaz de generalizar y funcionar bien en datos no vistos. El enfoque preciso de estas particiones puede variar según el problema y los datos disponibles, pero es importante tener en cuenta estas divisiones al desarrollar modelos de aprendizaje automático.