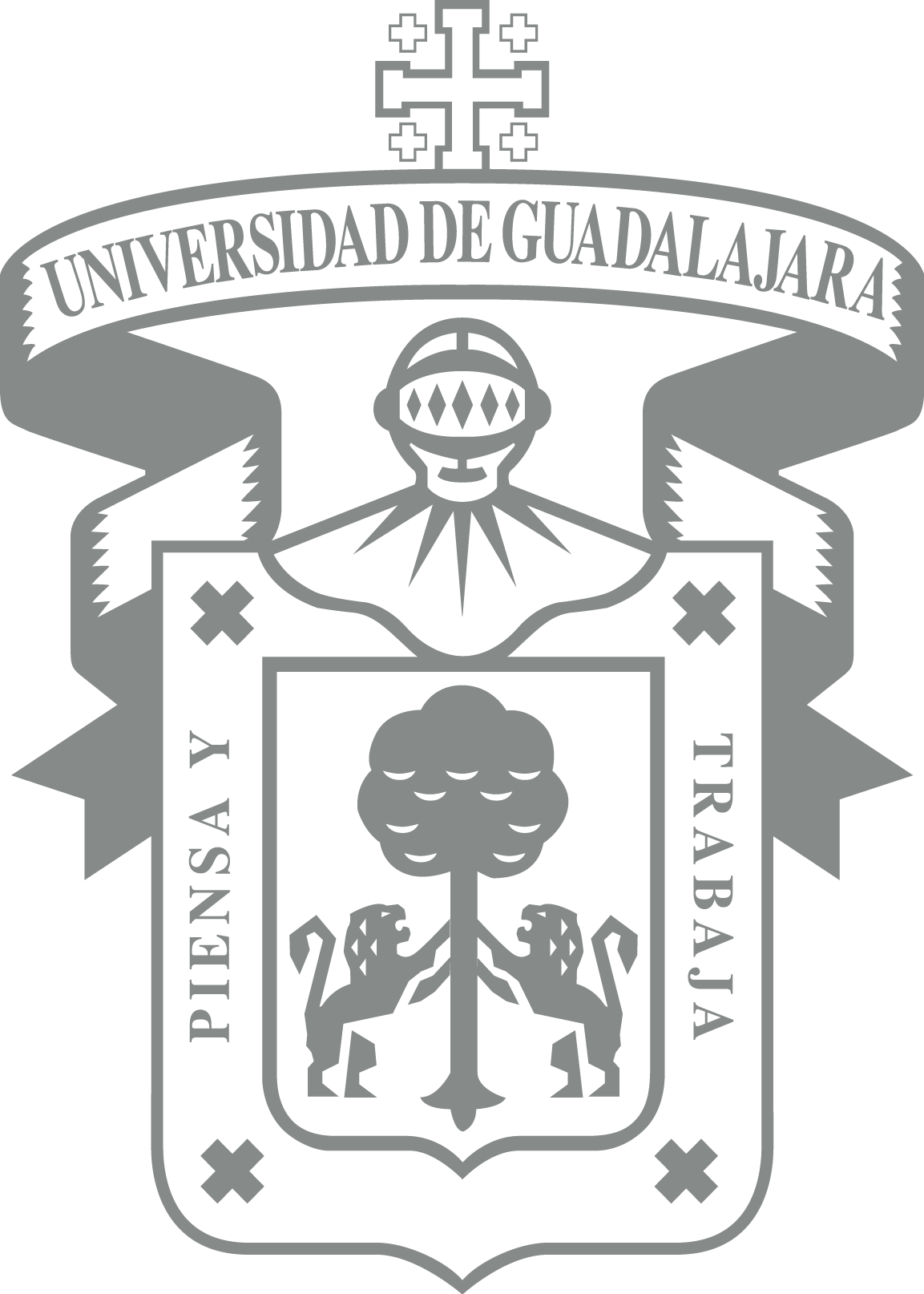
**Universidad de Guadalajara  
Centro Universitario de Ciencias e Ingenierías**



Departamento de Ciencias Computacionales  
Seminario de Solución de Problemas de Inteligencia Artificial ll

Profesor: OLIVA NAVARRO, DIEGO ALBERTO  
Alumno: Torres Hernández David

Código: 215428899 Carrera: INCO Sección: D05 Fecha: 04/03/2024

**Practica 1. Ejercicio 2**

# **Introducción**

El código proporcionado implementa un modelo de Perceptrón Simple y utiliza datos de archivos CSV para generar gráficas en 3D de los datos originales y los datos perturbados con diferentes niveles de perturbación. Se utiliza la biblioteca matplotlib para la visualización de datos en 3D y pandas para cargar los datos desde los archivos CSV.

# Instrucciones

Realizar un programa que permita generar un conjunto de particiones de entrenamiento considerando un dataset. El programa debe permitir seleccionar la cantidad de 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.

Imagen que contiene Tabla

Descripción generada automáticamente

1. Considerando la Tabla 1, modificar el punto . 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.

Gráfico, Diagrama, Gráfico de dispersión

Descripción generada automáticamente

# **Desarrollo**

1. **Implementación del Perceptrón Simple:** Se define una clase PerceptronSimple que representa el modelo de perceptrón. Este modelo tiene métodos para inicializar los pesos, predecir las etiquetas, entrenar el modelo y evaluar su rendimiento.
2. **Carga de Datos desde Archivos CSV:** Se proporciona una función load\_csv para cargar los datos desde archivos CSV. Esta función utiliza la biblioteca pandas para cargar los datos en un DataFrame y luego los convierte en un array de numpy para su posterior manipulación.
3. **Generación de Datos Perturbados:** Se define una función perturb\_data para introducir perturbaciones en los datos. Esta función añade ruido gaussiano a los datos originales para simular diferentes niveles de perturbación.
4. **Generación de Gráficas en 3D:** Se utilizan las funcionalidades de matplotlib y mpl\_toolkits para generar gráficas en 3D de los datos originales y los datos perturbados con diferentes niveles de perturbación. Se crea una figura con cuatro subgráficas, cada una representando un conjunto de datos diferente.

**Código:**

import numpy as np

import matplotlib.pyplot as plt

from mpl\_toolkits.mplot3d import Axes3D

import pandas as pd

class PerceptronSimple:

    def \_\_init\_\_(self, num\_features, learning\_rate=0.01, epochs=100):

        self.learning\_rate = learning\_rate

        self.epochs = epochs

        self.weights = np.zeros(num\_features + 1)  # +1 for the bias

    def predict(self, x):

        activation = np.dot(self.weights[1:], x) + self.weights[0]

        return 1 if activation >= 0 else -1

    def train(self, X, y):

        for \_ in range(self.epochs):

            for i in range(X.shape[0]):

                prediction = self.predict(X[i])

                self.weights[1:] += self.learning\_rate \* (y[i] - prediction) \* X[i]

                self.weights[0] += self.learning\_rate \* (y[i] - prediction)

    def evaluate(self, X\_test, y\_test):

        correct = 0

        for i in range(X\_test.shape[0]):

            if self.predict(X\_test[i]) == y\_test[i]:

                correct += 1

        return correct / X\_test.shape[0]

# Función para cargar datos desde un archivo CSV

def load\_csv(filename):

    data = pd.read\_csv(filename, header=None)

    return data.values

# Cargar datos originales

original\_data = load\_csv('spheres1d10.csv')

# Función para perturbar los datos con un cierto porcentaje

def perturb\_data(data, percentage):

    noise = np.random.normal(0, 0.1, size=data.shape)

    return data + percentage \* noise

# Cargar datos perturbados con diferentes porcentajes

perturbed\_10 = perturb\_data(load\_csv('spheres2d10.csv'), 0.1)

perturbed\_50 = perturb\_data(load\_csv('spheres2d50.csv'), 0.5)

perturbed\_70 = perturb\_data(load\_csv('spheres2d70.csv'), 0.7)

# Gráficas en 3D

fig = plt.figure(figsize=(15, 15))

# Datos originales

ax1 = fig.add\_subplot(221, projection='3d')

ax1.scatter(original\_data[:,0], original\_data[:,1], original\_data[:,2])

ax1.set\_title('Datos originales')

# Datos perturbados < 10%

ax2 = fig.add\_subplot(222, projection='3d')

ax2.scatter(perturbed\_10[:,0], perturbed\_10[:,1], perturbed\_10[:,2])

ax2.set\_title('Datos perturbados < 10%')

# Datos perturbados < 50%

ax3 = fig.add\_subplot(223, projection='3d')

ax3.scatter(perturbed\_50[:,0], perturbed\_50[:,1], perturbed\_50[:,2])

ax3.set\_title('Datos perturbados < 50%')

# Datos perturbados < 70%

ax4 = fig.add\_subplot(224, projection='3d')

ax4.scatter(perturbed\_70[:,0], perturbed\_70[:,1], perturbed\_70[:,2])

ax4.set\_title('Datos perturbados < 70%')

plt.show()

# **Resultados**

* **Datos Originales:** Los datos originales representan una distribución uniforme en un espacio tridimensional. No muestran ningún patrón específico y están bien dispersos.
* **Datos Perturbados < 10%:** Los datos perturbados con un nivel de perturbación del 10% muestran una ligera distorsión respecto a los datos originales debido al ruido gaussiano añadido. Sin embargo, la estructura general de los datos se mantiene.
* **Datos Perturbados < 50%:** Con un nivel de perturbación del 50%, los datos perturbados muestran una mayor distorsión y dispersión en comparación con los datos originales. La presencia de ruido gaussiano es más notable en este caso.
* **Datos Perturbados < 70%:** Con un nivel de perturbación del 70%, los datos perturbados muestran una distorsión significativa y una dispersión mucho mayor en comparación con los datos originales. La estructura de los datos se ve más afectada por el ruido gaussiano, lo que dificulta la identificación de patrones claros.

Gráfico, Gráfico radial

Descripción generada automáticamente

# **Conclusión**

El código proporciona una implementación completa para cargar datos desde archivos CSV, aplicar perturbaciones a los datos y visualizarlos en gráficas en 3D. El uso de la clase Perceptrón Simple y la generación de gráficas proporciona una base sólida para el análisis y la experimentación con modelos de aprendizaje automático en un entorno tridimensional.

Este enfoque facilita la comprensión y exploración de los datos, así como la evaluación del rendimiento de los modelos en conjuntos de datos modificados.

Principio del formulario