Entrega 6: diseño de redes neuronales

Aprendizaje Automatico y Big Data- Alejandro Barrachina Argudo

Introducción

En este documento se explicará el código del entregable 6B y el proceso de diseño de redes neuronales con *pytorch*.

Para esta práctica se usarán los siguientes *imports* vistos en la figura 0.1. Parte del código se reutiliza de la práctica anterior.

```
from sklearn.datasets import make_blobs
from sklearn.model_selection import train_test_split
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
from ComplexModel import ComplexModel
from SimpleModel import SimpleModel
import torch
import commandline
import os
import sys
```

Figura 0.1: Código de las bibliotecas usadas

También usaremos una serie de constantes para todo el programa (figura 0.2).

```
plot_folder = './memoria/images'

# Slice the Paired colormap into two segments for each dataset
paired_cmap = plt.cm.get_cmap('Paired')
# "Pastel" colors
cmap_dataset1 = ListedColormap(
    [paired_cmap(2*i) for i in range(6)])
# "Vibrant" colors
cmap_dataset2 = ListedColormap(
    [paired_cmap(2*i+1) for i in range(6)])
```

Figura 0.2: Constantes del programa

El *dataset* para esta práctica lo generamos aleatoriamente con la función *generate_data* (figura 0.4). El dataset se compone de una linea de datos "ideales" y datos con ruido para comprobar la eficacia de la red neuronal.

Para dibujar estos datos usaremos la función *plot_data* (figura 0.5).

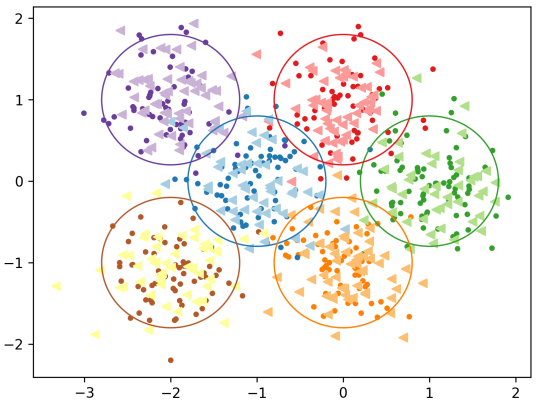

Figura 0.3: Ejemplo del *dataset*

```python
def generate_data() -> tuple[np.ndarray, np.ndarray, np.ndarray]:
    """Generates an artificial set of data with 6 classes

    Returns:
        tuple[np.ndarray, np.ndarray, np.ndarray]: X, y, centers of each class
    """
    classes: int = 6
    m: int = 800
    std: float = 0.4
    center: np.ndarray = np.array(
        [[-1, 0], [1, 0], [0, 1], [0, -1], [-2, 1], [-2, -1]])

    X, y = make_blobs(n_samples=m, centers=center,
                      cluster_std=std, random_state=2, n_features=2)
    return X, y, center
```

Figura 0.4: Función *generate_data*

```python
def plot_data(X_train: np.ndarray, y_train: np.ndarray, X_cv: np.ndarray, y_cv: np.ndarray,
    centers: np.ndarray, radius: float, name: str) -> None:
    """Plots the data with the training and cross validation data
    Args:
        X_train (np.ndarray): Training data
        y_train (np.ndarray): Training labels
        X_cv (np.ndarray): Cross validation data
        y_cv (np.ndarray): Cross validation labels
        centers (np.ndarray): centers of the classes
        radius (float): radius of the classes
        name (str): name of the file inside the plot folder
    """

    plt.scatter(X_train[:, 0], X_train[:, 1],
                c=y_train, marker=".", cmap=cmap_dataset2)
    plt.scatter(X_cv[:, 0], X_cv[:, 1], c=y_cv, marker='<', cmap=cmap_dataset1)
    circles = [plt.Circle(centers[i], radius * 2, color=cmap_dataset2(i), fill=False)
               for i in range(6)]
    for circle in circles:
        plt.gca().add_artist(circle)

    plt.savefig(f'{plot_folder}/{name}.png', dpi=300)
```

Figura 0.5: Función *plot_data*