UNIVERSIDAD AUTÓNOMA METROPOLITANA
División de Ciencias Naturales e Ingeniería
Departamento de Matemáticas Aplicadas y Sistemas
`$ Licenciatura en Ingeniería en Computación`

Neurociencias Computacionales y Aprendizaje Automático

Práctica 3: introducción a las redes neuronales

Cortes Lopez Alan Yair

García Núñez Rodrigo

Actividad 1

Seguir las instrucciones de la presentación para hacer la clasificación de **las tres variantes de flores de Iris**.

Como aparece en la presentación (*slide* 100), deben instanciar un Perceptrón multicapa de la biblioteca *SciKit-Learn* usando el conjunto de datos completo (las 150 muestras de flores).

Para que la clasificación tenga menos errores al clasificar deben de probar diferentes valores para el número de capas ocultas, el método de optimización para encontrar los pesos, y el número de iteraciones máximo.

En el reporte deben incluir:

1. El valor de los siguientes parámetros que hayan usado: *i*) número de capas ocultas y el número de neuronas de cada una (*hidden_layer_sizes*), *ii*) el método de optimización (*solver*), *iii*) el número de iteraciones máximo.

    1. *i) Número de capas ocultas y el número de neuronas de cada una (hidden_layer_sizes):* ***Hay una sola capa oculta con 1 neurona.***
    2. *ii) Método de optimización (solver):* ***Se utiliza el método de optimización 'adam'.***
    3. *Número de iteraciones máximo:* ***El número máximo de iteraciones se establece en 1000000.***

2. Mostrar la cuenta del total de puntos usado para probar el clasificador y el número de puntos mal clasificados.
    a. Recuerden que al igual que en el ejercicio donde solamente usamos 2 variantes de Iris, deben dividir el conjunto total de entrenamiento en dos partes: un conjunto para entrenar (`train_data`, `train_labels`) y otro para probar el desempeño (`test_data`, `test_labels`).

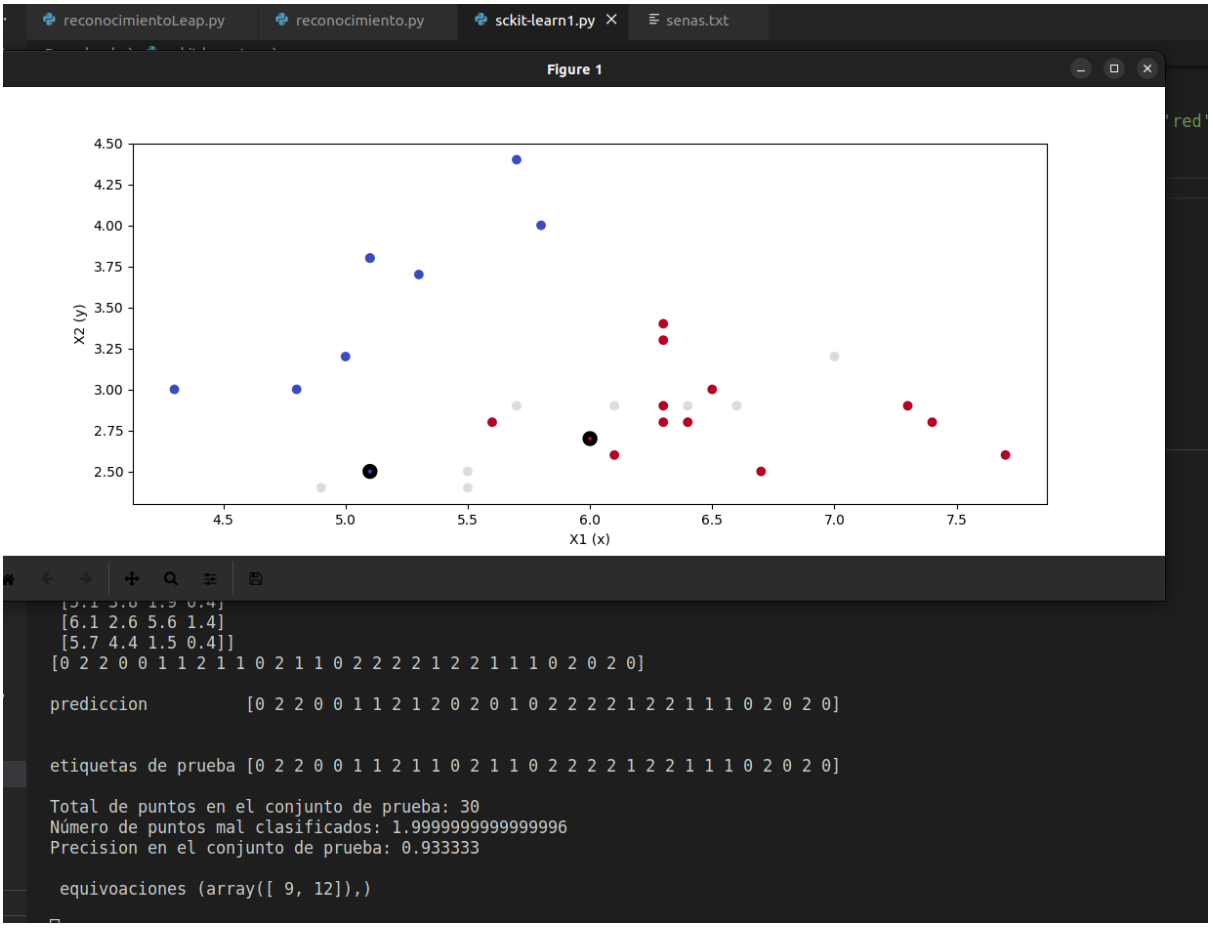
```
# Divide nuestros puntos totales, de manera que el 20% serán
# puntos de prueba, y el 80% puntos de entrenamiento.
datasets = train_test_split(datos,
                            etiquetas,
                            test_size=0.2)

train_data, test_data, train_labels, test_labels = datasets
```

3. Una gráfica donde se muestren los puntos de entrenamiento con su clase correcta usando tres colores diferentes (tip en el *slide* 109), y encerrados en un círculo, los puntos mal clasificados:

    a. Para graficar en 2D solamente tomen las dos primeras componentes de cada punto de `test_data`.
    b. Para saber cuáles puntos están mal clasificados comparen las clases correctas del conjunto de prueba, `test_label`, con la clase estimada que se obtiene mediante:

Ejecución del código sckit-learn.py:



En la imagen observamos que se usaron 30 puntos de prueba, de los cuales dos estuvieron mal clasificados -el número 9 y el 12-. Esto se puede corroborar comparando el vector de la predicción y el de las etiquetas de prueba. En el grafico, los puntos mal clasificados se encuentran remarcadoo por un circulo negro.

Actividad 2

Hacer clasificación de gestos con la mano para jugar a piedra, papel o tijeras usando el *Leap Motion*.

En el reporte deben incluir (de manera similar al ejercicio de la flores):

1. El valor de los siguientes parámetros que hayan usado: *i*) número de capas ocultas y el número de neuronas de cada una (*hidden_layer_sizes*), *ii*) el método de optimización (*solver*), *iii*) el número de iteraciones máximo.

    1) *El parámetro hidden_layer_sizes se establece en (20, 20), lo que significa que hay dos capas ocultas en la red y cada capa oculta tiene 20 neuronas.*
    2) *El parámetro solver se establece en 'adam'. Adam es un algoritmo de optimización basado en el descenso de gradiente estocástico. Se utiliza para ajustar los pesos de la red neuronal durante el proceso de entrenamiento.*
    3) *El parámetro max_iter se establece en 1000000. Indica el número máximo de iteraciones que el algoritmo de optimización (Adam en este caso) realizará durante el entrenamiento antes de detenerse, incluso si el modelo no ha convergido completamente.*

2. Mostrar la cuenta del total de puntos usado para probar el clasificador y el número de puntos mal clasificados.
    1. Como en la actividad 1 de las flores de Iris, deben dividir el conjunto total de entrenamiento en dos partes: un conjunto para entrenar y otro para probar el desempeño de la red.

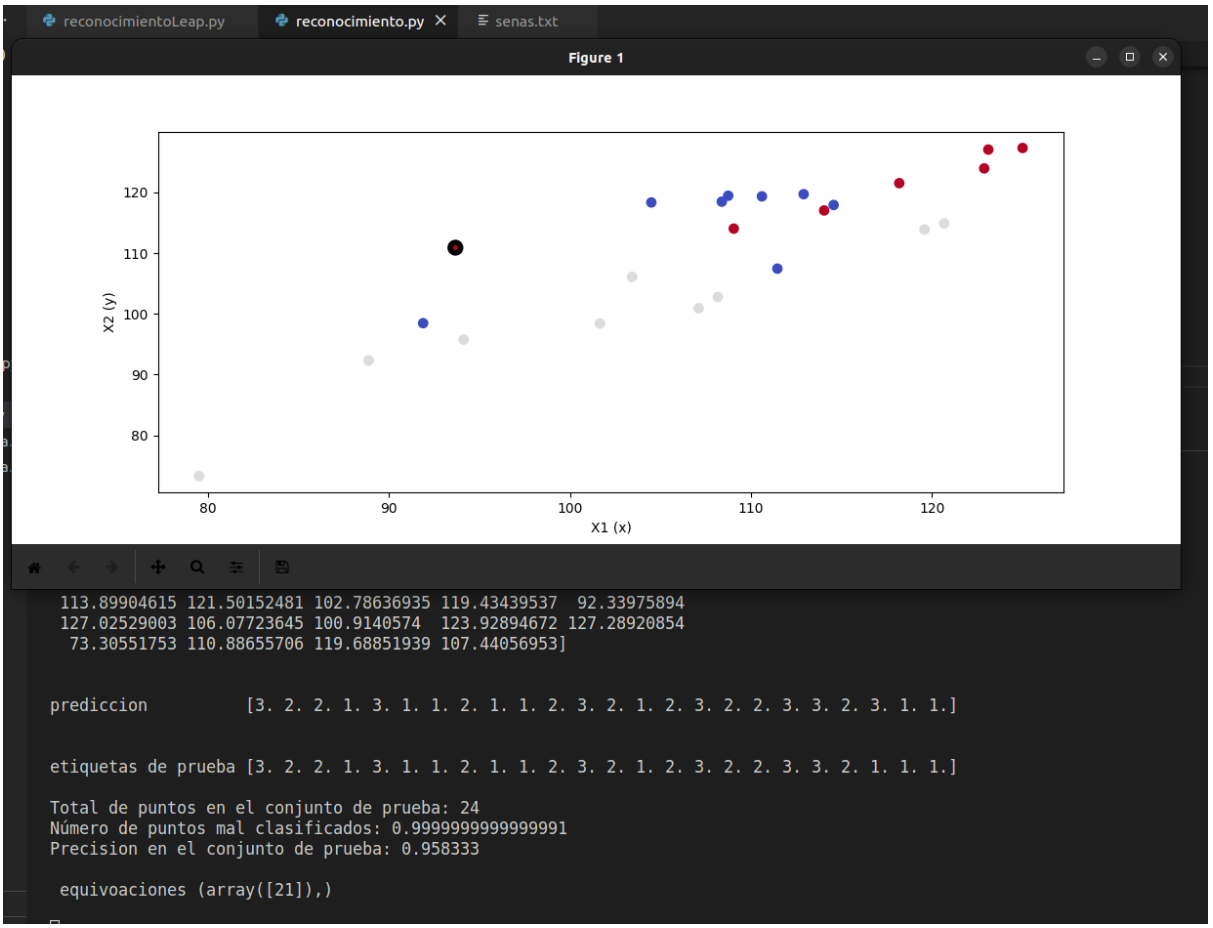
```
datasets = train_test_split(datos,
                            etiquetas,
                            test_size=0.2)

train_data, test_data, train_labels, test_labels = datasets
```

3. Una gráfica donde se muestran los puntos de entrenamiento con su clase correcta usando tres colores diferentes (tip en el *slide* 109), y encerrados en un círculo, los puntos mal clasificados:

1. Para graficar en 2D solamente tomen las dos primeras componentes de cada punto de entrada.
2. Para saber cuáles puntos están mal clasificados comparen las clases correctas del conjunto de prueba contra la clase estimada

Ejecución del código reconocimiento.py:



En la imagen observamos que se usaron 24 puntos de prueba, de los cuales solo uno estuvo mal clasificado -el número 21-. Esto se puede corroborar comparando el vector de la predicción y el de las etiquetas de prueba. En el grafico, el punto mal clasificado se encuentra remarcado por un circulo negro.