Universidad de Guadalajara
Centro Universitario de Ciencias Exactas e Ingenierías

Ingeniería en Computación
Seminario de Solución de Problemas de Inteligencia Artificial II
Profesor: Campos Peña Diego

Lepiz Hernández Manuel Alejandro
Código: 214797998

Práctica 1.

Ejercicio 4.

Introducción

Se realizará en Python la implementación del Algoritmo de retropropagación, usando un archivo .csv para la entrada de datos y un perceptrón multicapa para la resolución del algoritmo.

Iris es el género de una planta herbácea con flores que se utiliza en decoración. Dentro de este género existen muy diversas especies entre las que se han estudiado la Iris setosa, la Iris versicolor y la Iris virginica

Desarrollo

Las tres especies se pueden diferenciar en base a las dimensiones de sus pétalos y sépalos. Se ha recopilado la información de 50 plantas de cada especie y se han almacenado en el archivo irisbin.csv.

Dichas mediciones están en centímetros junto con un código binario que indica la especie a la que pertenece [-1, -1, 1] = setosa, [-1, 1, -1] = versicolor, [1, -1, -1] = virginica.

Se debe crear un programa capaz de clasificar automáticamente los datos de 150 patrones usando un perceptrón multicapa. Es recomendable considerar 80% de los datos para entrenamiento y 20% para generalización.

Se hizo la implementación de una Clase para representar el perceptrón, usando el implementado en las prácticas anteriores.

```
[55] import numpy as np
     import matplotlib.pyplot as plt

     class PerceptronMulticapa:
         def __init__(self, nEntradas, capasOcultas, nSalidas, tazaAprendizaje, epochs):
             self.nEntradas = nEntradas
             self.capasOcultas = capasOcultas
             self.nSalidas = nSalidas
             self.tazaAprendizaje = tazaAprendizaje
             self.epochs = epochs
             self.pesos = [np.random.rand(capasOcultas[0], nEntradas)]
             self.sesgos = [np.zeros(capasOcultas[0])]

             for i in range(1, len(capasOcultas)):
                 self.pesos.append(np.random.rand(capasOcultas[i], capasOcultas[i-1]))
                 self.sesgos.append(np.zeros(capasOcultas[i]))

             self.pesos.append(np.random.rand(nSalidas, capasOcultas[-1]))
             self.sesgos.append(np.zeros(nSalidas))
```

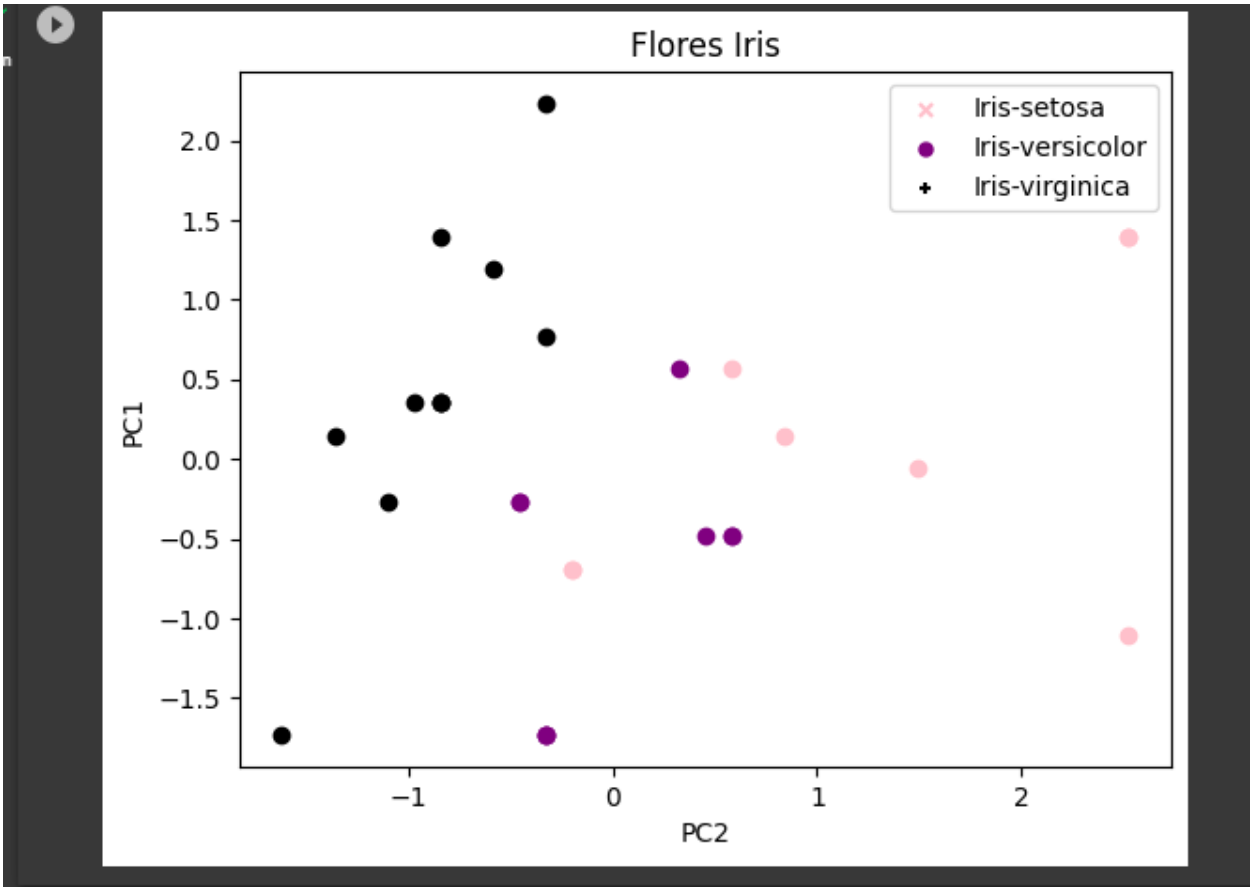
Se realizaron distintas pruebas usando el mismo set de datos, pero con distintos valores dados al perceptrón

```
            if self.TipoFlor(prediccion, capa) != "None":
                plt.scatter(entradas[0], entradas[1], color=coloresFlores[np.argmax(capa)])
                prediccionesCorrectas += 1
        precision = prediccionesCorrectas / prediccionesTotal
        self.leaveOneOut()
        #self.leaveOneOut(2)
        for i, capa in enumerate(tiposFlores):
            plt.scatter([], [], color=coloresFlores[i], label=capa, marker=marcadores[i], s=25)
        plt.legend()
        plt.xlabel('PC2')
        plt.ylabel('PC1')
        plt.title('Flores Iris')
        plt.show()
```

```
floresIris = Separador(nEntradas = 4, capasOcultas = [8], nSalidas = 3, tazaAprendizaje = 0.1, epochs = 100)
floresIris.leerArchivoDatos()
floresIris.Entrenamiento()
floresIris.ProbarFlores()
```

```
Precision prueba: 83.33%
Desviacion Estándar: 37.27%
Error: 16.666666666666664%
```

Se grafican los datos obtenidos al usar el perceptrón



Resultado esperado

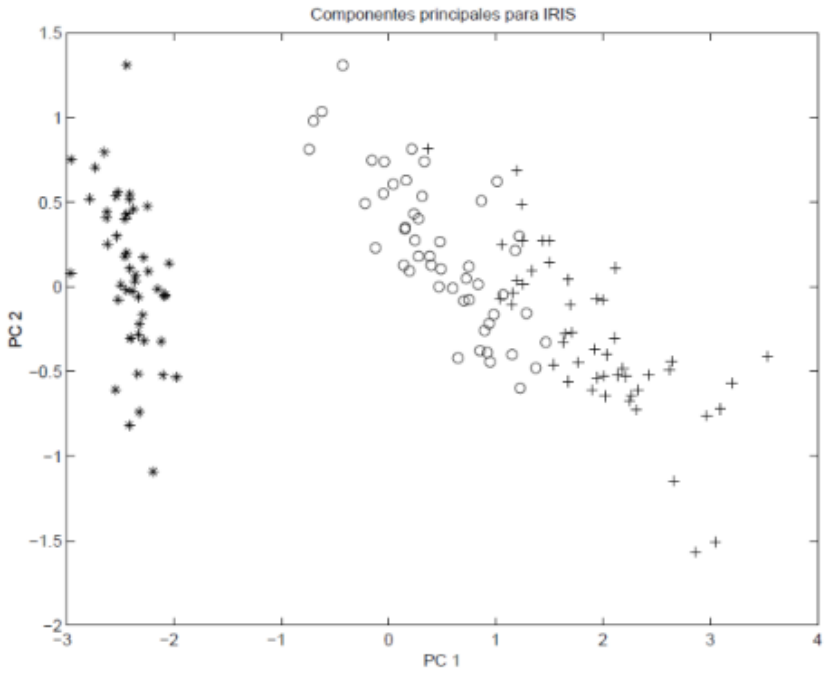


Figura 4. Proyección en dos dimensiones de la distribución de clases para el dataset Iris

Conclusión

Los resultados fueron acercados a los esperados, de manera general.

Al hacer la prueba con el algoritmo k-Out el perceptrón tardaba mucho tiempo pero hacia la clasificación.

No se pudo adjuntar de nuevo ya que el tiempo era aproximadamente de media hora, pero no daban errores los resultados, para mejor clasificación se optó por usar una librería externa.

Conclusión General

Hubo algunos problemas al inicio de la materia para la implementación de los algoritmos, pero con el paso del tiempo se fue comprendiendo mejor cómo realizarlos.

Todo el trabajo fue realizado en Python, usando código implementado personalmente y al final usando algunas librerías para validación de datos

El ploteo de datos ha sido una parte costosa de implementar ya que no se habían usado muy a fondo esas librerías, pero leyendo la documentación se resolvió este problema, aunque de manera muy básica.