Universidad de Guadalajara
Centro Universitario de Ciencias e Ingenierías

Departamento de Ciencias Computacionales

Seminario de Solución de Problemas de Inteligencia Artificial II

Profesor: OLIVA NAVARRO, DIEGO ALBERTO

Alumnos: Torres Hernández David
Sandoval Gil Sunem

Código: 215428899 Carrera: INCO Sección: D05 Fecha: 22/04/2024

Practica 1. Ejercicio 3

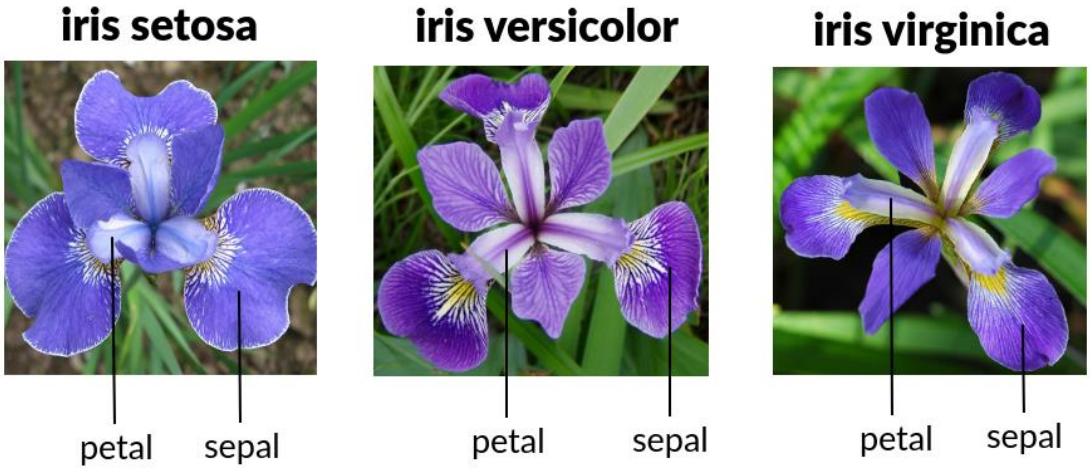
Introducción

Iris es el género de una planta herbácea con flores que se utilizan 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 (ver Figura 3).

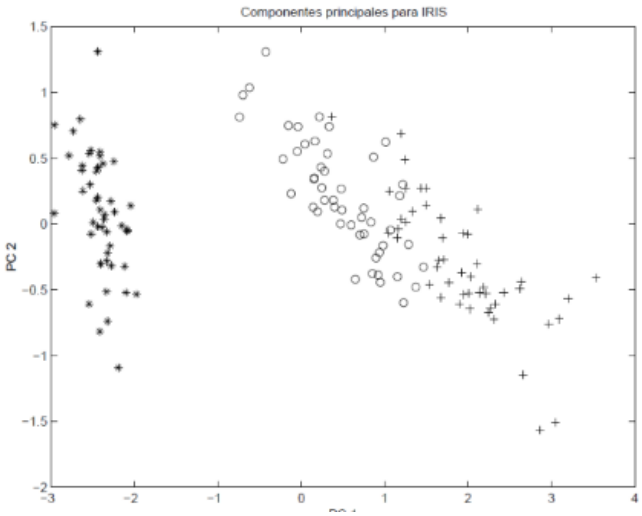
Figura 3. Muestra de la especie Iris virginica.

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, la Figura 4 muestra la distribución de los datos contenidos en el archivo. 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.



Con la estructura optima de la red, se deben validar los resultados usando lo métodos leave-k-out y leave-one-out con un perceptrón multicapa como clasificador. Se debe estimar el error esperado de clasificación, el promedio y la desviación estándar de ambos métodos



Código

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold, LeaveOneOut
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score
from sklearn.decomposition import PCA

class PerceptronMulticapa:
    def __init__(self, hidden_layer_sizes=(100,), activation='relu',
solver='adam', max_iter=2000, random_state=None):
        self.model = MLPClassifier(hidden_layer_sizes=hidden_layer_sizes,
activation=activation,
                                   solver=solver, max_iter=max_iter,
random_state=random_state)

    def train(self, X_train, y_train):
        self.model.fit(X_train, y_train)

    def evaluate(self, X_test, y_test):
        y_pred = self.model.predict(X_test)
        return accuracy_score(y_test, y_pred)
```

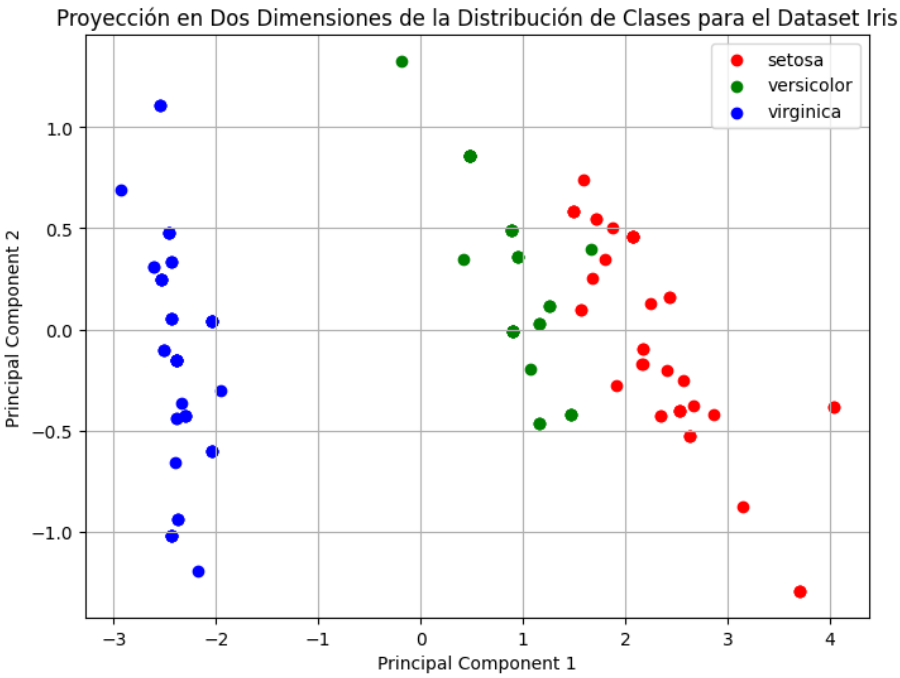
```
def leave_k_out(X, y, k):
    kf = KFold(n_splits=k)
    errors = []
    for train_index, test_index in kf.split(X):
        X_train, X_test = X[train_index], X[test_index]
        y_train, y_test = y[train_index], y[test_index]
        perceptron_multicapa = PerceptronMulticapa(hidden_layer_sizes=(10,),
activation='relu', solver='adam', max_iter=2000, random_state=42)
        perceptron_multicapa.train(X_train, y_train)
        errors.append(1 - perceptron_multicapa.evaluate(X_test, y_test))
    return errors

def leave_one_out(X, y):
    loo = LeaveOneOut()
    errors = []
    for train_index, test_index in loo.split(X):
        X_train, X_test = X[train_index], X[test_index]
        y_train, y_test = y[train_index], y[test_index]
        perceptron_multicapa = PerceptronMulticapa(hidden_layer_sizes=(10,),
activation='relu', solver='adam', max_iter=2000, random_state=42)
        perceptron_multicapa.train(X_train, y_train)
        errors.append(1 - perceptron_multicapa.evaluate(X_test, y_test))
    return errors

# Función para cargar y preprocesar los datos del archivo irisbin.csv
def load_iris_data(file_path):
    data = pd.read_csv(file_path)
    species = []
    for i in range(len(data)):
        if (data.iloc[i][-3:] == [-1, -1, 1]).all():
            species.append('setosa')
        elif (data.iloc[i][-3:] == [-1, 1, -1]).all():
            species.append('versicolor')
        elif (data.iloc[i][-3:] == [1, -1, -1]).all():
            species.append('virginica')
    X = data.iloc[:, :-3].values
    y = np.array(species)
    return X, y

# Cargar datos desde el archivo irisbin.csv
X_iris, y_iris = load_iris_data('irisbin.csv')

# Leave-k-out
k_out_errors = leave_k_out(X_iris, y_iris, k=10)
mean_k_out = np.mean(k_out_errors)
```

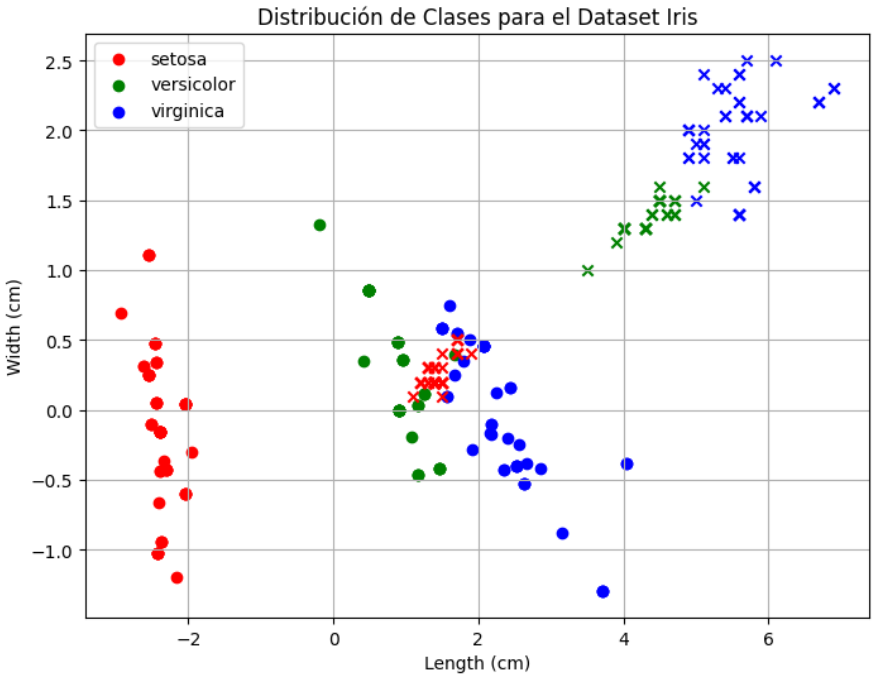
```
std_k_out = np.std(k_out_errors)

# Leave-one-out
one_out_errors = leave_one_out(X_iris, y_iris)
mean_one_out = np.mean(one_out_errors)
std_one_out = np.std(one_out_errors)

print("Leave-k-out: Mean Error =", mean_k_out, "Standard Deviation =", std_k_out)
print("Leave-one-out: Mean Error =", mean_one_out, "Standard Deviation =",
std_one_out)

# Reducción de dimensionalidad con PCA para la proyección en dos dimensiones
pca = PCA(n_components=2)
X_2d = pca.fit_transform(X_iris)

# Dividir los datos en entrenamiento y prueba (80% entrenamiento, 20% prueba)
X_train_iris, X_test_iris, y_train_iris, y_test_iris = train_test_split(X_2d,
y_iris, test_size=0.2, random_state=42)

# Entrenar el perceptrón multicapa
perceptron_multicapa = PerceptronMulticapa(hidden_layer_sizes=(10,),
activation='relu', solver='adam', max_iter=2000, random_state=42)
perceptron_multicapa.train(X_train_iris, y_train_iris)

# Evaluar el modelo
accuracy_iris = perceptron_multicapa.evaluate(X_test_iris, y_test_iris)
print(f'Accuracy (irisbin.csv): {accuracy_iris}')

# Graficar la proyección en dos dimensiones de la distribución de clases
plt.figure(figsize=(8, 6))
colors = {'setosa': 'red', 'versicolor': 'green', 'virginica': 'blue'}

for species in colors:
    subset = X_2d[y_iris == species]
    plt.scatter(subset[:, 0], subset[:, 1], color=colors[species], label=species)

plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('Proyección en Dos Dimensiones de la Distribución de Clases para el
Dataset Iris')
plt.legend()
plt.grid(True)
plt.show()
```

```python
# Graficar la distribución de clases en dos dimensiones
plt.figure(figsize=(8, 6))

for species in colors:
    subset = X_2d[y_iris == species]
    plt.scatter(subset[:, 0], subset[:, 1], color=colors[species], label=species)

plt.xlabel('Sepal Length (cm)')
plt.ylabel('Sepal Width (cm)')
plt.title('Distribución de Clases para el Dataset Iris')
plt.legend()
plt.grid(True)

# Agregar también los datos de "petal"
for species in colors:
    subset = X_iris[y_iris == species]
    plt.scatter(subset[:, 2], subset[:, 3], color=colors[species], marker='x')

plt.xlabel('Length (cm)')
plt.ylabel('Width (cm)')
plt.title('Distribución de Clases para el Dataset Iris')
plt.legend()
plt.grid(True)
plt.show()
```

Capturas:

```
PS C:\Users\IDEAPAD 3\Desktop\11-Semestre\Sem. IA 2\Practica 1 Ejercicio 4> & "C:
/Users/IDEAPAD 3/AppData/Local/Programs/Python/Python311/python.exe" "c:/Users/ID
EAPAD 3/Desktop/11-Semestre/Sem. IA 2/Practica 1 Ejercicio 4/main.py"
Leave-k-out: Mean Error = 0.006666666666666665 Standard Deviation = 0.01999999999
9999993
Leave-one-out: Mean Error = 0.006711409395973154 Standard Deviation = 0.081647819
19863381
Accuracy (irisbin.csv): 1.0
PS C:\Users\IDEAPAD 3\Desktop\11-Semestre\Sem. IA 2\Practica 1 Ejercicio 4>
```





Conclusiones

En este ejercicio podemos abordar la clasificación automática de especies de plantas Iris utilizando un perceptrón multicapa, lo cual demuestra la aplicación

de técnicas de inteligencia artificial en la botánica y la biología empleando métodos de validación como leave-k-out y leave-one-out para estimar el error de clasificación, promedio y desviación estándar, lo que resalta la importancia de evaluar y validar los modelos de manera rigurosa en el campo de la inteligencia artificial.

El uso de técnicas de reducción de dimensionalidad, como PCA para la proyección en dos dimensiones de los datos de plantas Iris, resalta la importancia de la visualización de datos en la comprensión y análisis de conjuntos de datos complejos, facilitando la interpretación de patrones y la toma de decisiones informadas en el campo de la inteligencia artificial y la ciencia de datos.