Centro Universitario de Ciencias Exactas e Ingenierías

Ingeniería en Computación
Práctica 1 ejercicio 4

PRESENTA:

Ramirez Gutierrez Hugo Vladimir
Código: 220287144
Materia:

Seminario de Solución de problemas de Inteligencia Artificial 2

Docente: Diego Campos Pena

Índice:

Introducción

El MultiLayerPerceptron (MLP) es una arquitectura fundamental en el campo del aprendizaje automático, especialmente en el área de redes neuronales artificiales. Inspirado en el funcionamiento del cerebro humano, un MLP consiste en múltiples capas de neuronas interconectadas, que pueden aprender y generalizar patrones complejos en conjuntos de datos.

El MLP se compone de al menos tres capas: una capa de entrada, una o más capas ocultas y una capa de salida. Cada capa está formada por neuronas que procesan la información y transfieren señales a través de conexiones ponderadas. Durante el entrenamiento, el MLP ajusta estos pesos para minimizar una función de pérdida, lo que permite que la red aprenda a hacer predicciones precisas.

En este contexto, el código proporcionado implementa un MLP en Python utilizando la biblioteca NumPy y otras herramientas de aprendizaje automático como scikit-learn. El MLP se utiliza para clasificar especies de iris utilizando un conjunto de datos preprocesados.

El código realiza varias tareas, incluyendo la preparación de datos, el entrenamiento del MLP, la evaluación del rendimiento del modelo y la visualización de resultados. Se emplean técnicas como validación cruzada (Leave-One-Out y Leave-K-Out) para evaluar el rendimiento del modelo de manera robusta y evitar el sobreajuste.

A lo largo de este informe, explicaremos en detalle cómo funciona el MLP y cómo se implementa en el código proporcionado, así como los resultados obtenidos en la clasificación de las especies de iris.

Desarrollo:

Definición de la clase MultiLayerPerceptron:
- La clase MultiLayerPerceptron implementa una red neuronal de múltiples capas (MLP) para clasificación.
- Inicializa los pesos y sesgos de la red con valores aleatorios.
- Proporciona métodos para entrenar la red, hacer predicciones y evaluar su rendimiento.

Métodos principales:
- sigmoid(x): Función de activación sigmoide utilizada en las capas ocultas de la red.
- sigmoid_derivative(x): Derivada de la función de activación sigmoide.
- softmax(x): Función de activación softmax utilizada en la capa de salida para la clasificación multiclase.
- forward(X): Realiza la propagación hacia adelante (forward propagation) de la red.

- backward(X, y, learning_rate): Realiza la retropropagación (backward propagation) y ajusta los pesos y sesgos de la red.
- train(X, y, epochs, learning_rate): Entrena la red durante un número específico de épocas.
- predict(X): Realiza predicciones sobre un conjunto de datos dado.
- evaluate_loo(X, y): Evalúa el rendimiento de la red utilizando validación Leave-One-Out (LOO).
- evaluate_lko(X, y, k): Evalúa el rendimiento de la red utilizando validación Leave-K-Out (LKO).

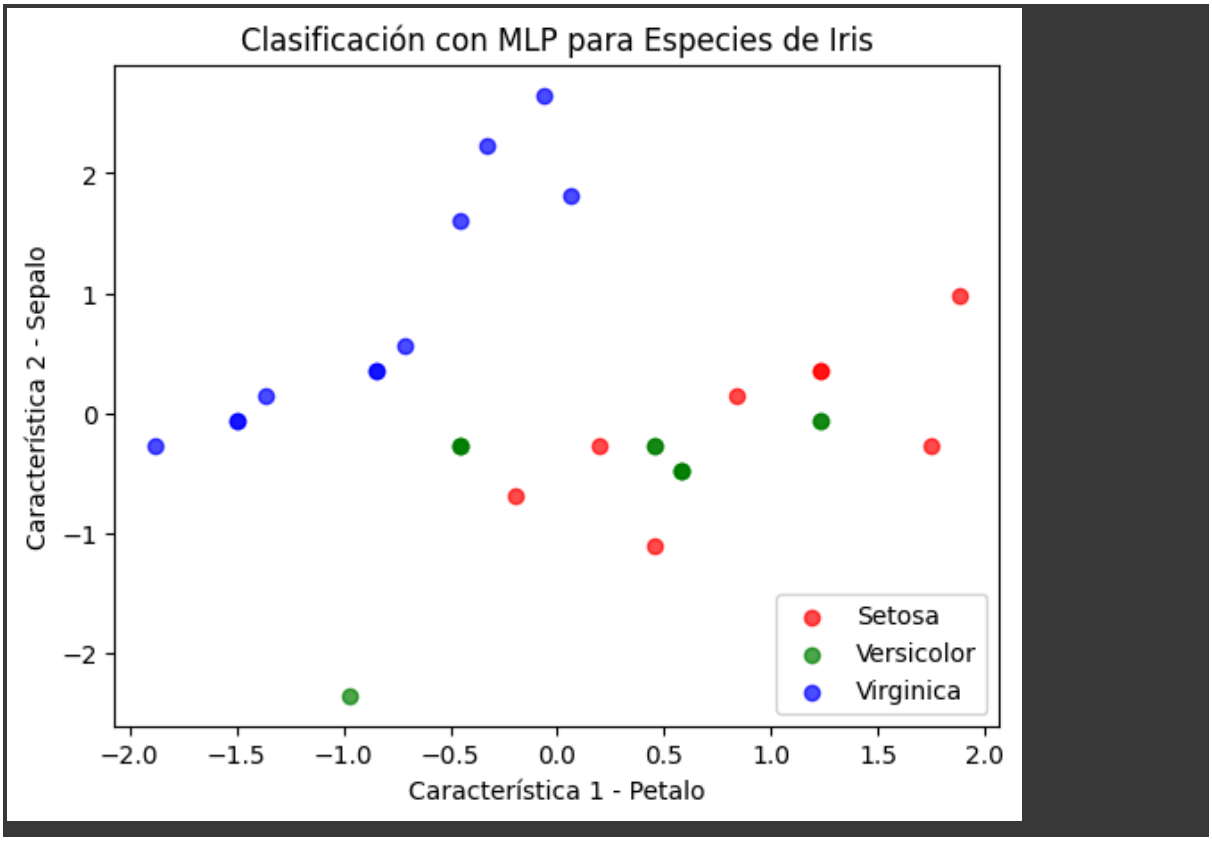
Proceso de entrenamiento y evaluación:
- Los datos de entrada se normalizan utilizando StandardScaler de sklearn.
- Se divide el conjunto de datos en conjuntos de entrenamiento y prueba utilizando train_test_split.
- La red se entrena en el conjunto de entrenamiento utilizando retropropagación.
- Se evalúa el rendimiento de la red utilizando validación cruzada (Leave-One-Out y Leave-K-Out) y se calculan las métricas de rendimiento, como el error esperado, el promedio y la desviación estándar de la precisión.

Visualización de resultados:
- Se muestran los resultados de precisión y desviación estándar para Leave-One-Out y Leave-K-Out.
- Se imprimen las predicciones de la red y las especies reales para el conjunto de prueba.
- Se muestra un gráfico de dispersión de las características de prueba, coloreado según las especies reales del iris.

Resultados:

leave-k-out
Error Esperado: 0.30000000000000004
Promedio: 0.7
Desviación Estándar: 0.05962847939999442
leave-one-out
Error Esperado: 0.26
Promedio: 0.74
Desviación Estándar: 0.43863424398922624
Predicciones y Especies Reales:
1: Predicción=Virginica, Especie real=Versicolor
2: Predicción=Setosa, Especie real=Setosa
3: Predicción=Setosa, Especie real=Setosa
4: Predicción=Setosa, Especie real=Setosa
5: Predicción=Setosa, Especie real=Setosa

6: Predicción=Virginica, Especie real=Versicolor
7: Predicción=Virginica, Especie real=Virginica
8: Predicción=Virginica, Especie real=Versicolor
9: Predicción=Setosa, Especie real=Setosa
10: Predicción=Setosa, Especie real=Setosa
11: Predicción=Setosa, Especie real=Setosa
12: Predicción=Virginica, Especie real=Versicolor
13: Predicción=Virginica, Especie real=Versicolor
14: Predicción=Virginica, Especie real=Virginica
15: Predicción=Virginica, Especie real=Virginica
16: Predicción=Virginica, Especie real=Versicolor
17: Predicción=Virginica, Especie real=Versicolor
18: Predicción=Virginica, Especie real=Virginica
19: Predicción=Setosa, Especie real=Setosa
20: Predicción=Setosa, Especie real=Setosa
21: Predicción=Virginica, Especie real=Virginica
22: Predicción=Virginica, Especie real=Virginica
23: Predicción=Virginica, Especie real=Versicolor
24: Predicción=Setosa, Especie real=Setosa
25: Predicción=Virginica, Especie real=Versicolor
26: Predicción=Virginica, Especie real=Versicolor
27: Predicción=Virginica, Especie real=Versicolor
28: Predicción=Setosa, Especie real=Setosa
29: Predicción=Virginica, Especie real=Virginica
30: Predicción=Virginica, Especie real=Virginica



Conclusión:

El MultiLayerPerceptron (MLP) implementado en Python demuestra su capacidad para clasificar especies de iris con precisión. Utilizando técnicas como validación cruzada, el MLP ajusta sus pesos para adaptarse a los datos y ofrece resultados confiables. Su flexibilidad y eficacia lo convierten en una herramienta valiosa para el aprendizaje automático.