Huỳnh Minh Luân
MSSV: B2106842

Bài tập số 1: Phân loại iris bằng mạng Perceptron đơn tầng (Single Layer Perceptron)

1. Nhập các thư viện

```
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense
from sklearn.datasets import load_iris
import pandas as pd
from sklearn.model_selection import train_test_split
```

- **from tensorflow.keras import Sequential:** Nhập lớp **Sequential** từ thư viện Keras của TensorFlow. Lớp Sequential dùng để tạo mô hình mạng nơ-ron theo dạng tuần tự, nơi các lớp được xếp chồng lên nhau.
- **from tensorflow.keras.layers import Dense:** Nhập lớp **Dense** từ Keras. Lớp **Dense** dùng để tạo các lớp nơ-ron đầy đủ kết nối (fully connected layer), mà mỗi nơ-ron trong lớp sẽ kết nối với tất cả các nơ-ron của lớp trước đó.
- **from sklearn.datasets import load_iris:** Nhập hàm **load_iris** từ thư viện scikit-learn. Hàm này dùng để tải tập dữ liệu hoa Iris, một tập dữ liệu phổ biến trong học máy để phân loại.
- **import pandas as pd:** Nhập thư viện **pandas** và đặt tên là **pd**. Thư viện này thường được dùng để xử lý và phân tích dữ liệu.
- **from sklearn.model_selection import train_test_split:** Nhập hàm **train_test_split** từ scikit-learn. Hàm này dùng để chia tập dữ liệu thành tập huấn luyện và tập kiểm tra, giúp đánh giá mô hình học máy.

2. Đọc file dữ liệu

```
#đọc file dữ liệu
df = pd.read_csv( filepath_or_buffer: 'iris.data', header=None)
```

- **pd.read_csv:** Đây là một hàm của thư viện pandas dùng để đọc dữ liệu từ một file CSV (Comma-Separated Values) và trả về một DataFrame. **DataFrame** là cấu trúc dữ liệu chính của pandas, tương tự như một bảng dữ liệu trong SQL hay một bảng tính Excel.

- **'iris.data':** Đây là đường dẫn đến file CSV mà bạn muốn đọc. Trong trường hợp này, file có tên là iris.data và nằm trong thư mục con **iris** (tạo một cấu trúc thư mục kiểu **iris.data**). Đường dẫn này cần phải đúng và file phải tồn tại tại vị trí này để việc đọc dữ liệu thành công.
- **header=None**: Tham số này chỉ định rằng file CSV không chứa hàng tiêu đề (header) ở dòng đầu tiên. Nếu không chỉ định **header=None**, pandas sẽ mặc định coi dòng đầu tiên của file là tên cột (header). Khi **header=None,** pandas sẽ tự động gán các tên cột là số nguyên liên tiếp (0, 1, 2, ...).

3. Mã hóa nhãn bằng map

```
#mã hóa nhãn bằng map
label_mapping = {'Iris-setosa': 0, 'Iris-versicolor': 1}
Data['label_encoded'] = Data[label_column].map(label_mapping)
```

label_mapping = {'Iris-setosa': 0, 'Iris-versicolor': 1}

- Đây là một dictionary định nghĩa cách ánh xạ các giá trị nhãn (lớp) thành các số nguyên.
- Trong dictionary này, nhãn **'Iris-setosa'** được ánh xạ thành số 0 và **'Iris-versicolor'** được ánh xạ thành số 1.

Data['label_encoded'] = Data[label_column].map(label_mapping)

- **Data** là tên của DataFrame chứa dữ liệu.
- **label_column** là tên cột trong DataFrame **Data** chứa các nhãn (lớp) mà bạn muốn mã hóa.
- **Data[label_column]** truy cập cột chứa các nhãn từ DataFrame.
- **.map(label_mapping)** là phương thức của pandas Series (cột dữ liệu) được sử dụng để ánh xạ giá trị của Series bằng cách sử dụng dictionary **label_mapping.**
    - **.map** sẽ thay thế mỗi giá trị trong cột **label_column** bằng giá trị tương ứng từ dictionary **label_mapping.**
- **Data['label_encoded']** thêm một cột mới vào DataFrame **Data** với tên **'label_encoded'**. Cột này chứa các giá trị đã được mã hóa (số nguyên) từ cột nhãn gốc.

4. xóa cột nhãn cũ

```
#xóa cột nhãn cũ
Data = Data.drop(label_column, axis=1)
```

5. Tách x, y

```
#tách x, y
X = Data.drop( labels: 'label_encoded', axis=1)  #
y = Data['label_encoded']
print(Data)
```

y = Data['label_encoded']

- **Data['label_encoded']**: Truy cập cột **'label_encoded'** từ DataFrame **Data**, chứa các nhãn mã hóa (encoded labels).
- Kết quả là một pandas Series chứa giá trị của cột **'label_encoded'**, và kết quả được gán cho biến y.
- Biến **y** sẽ chứa các nhãn mà mô hình sẽ cố gắng dự đoán dựa trên các đặc trưng trong **X.**

6. Tạo dữ liệu huấn luyện

```
#tạo tệp dữ liệu huấn luyện
X_train, X_test, y_train, y_test = train_test_split( *arrays: X, y, test_size=0.2, random_state=42)

model = Sequential()
model.add(Dense(1, input_shape=(4,), activation='sigmoid'))
```

train_test_split: Đây là một hàm từ thư viện scikit-learn dùng để chia dữ liệu thành các tập huấn luyện và kiểm tra.

- **X**: DataFrame hoặc numpy array chứa các đặc trưng (features) của dữ liệu.
- **y**: Series hoặc numpy array chứa nhãn (labels) tương ứng với các đặc trưng trong **X**.
- **test_size=0.2:** Tham số này chỉ định tỷ lệ dữ liệu sẽ được dùng làm tập kiểm tra. 0.2 có nghĩa là 20% dữ liệu sẽ được dùng cho tập kiểm tra, còn 80% sẽ được dùng cho tập huấn luyện.
- **random_state=42**: Tham số này đảm bảo rằng quá trình chia dữ liệu là có thể lặp lại. Nếu bạn sử dụng cùng một giá trị **random_state**, bạn sẽ nhận được cùng một chia dữ liệu mỗi khi chạy mã.

Sequential(): Tạo một mô hình mạng nơ-ron tuần tự (sequential model), nơi các lớp được thêm vào theo thứ tự tuần tự. Đây là loại mô hình đơn giản nhất trong Keras.
model.add(Dense(1, input_shape=(4,), activation='sigmoid')):

- **Dense(1):** Thêm một lớp nơ-ron đầy đủ kết nối (fully connected layer) với 1 nơ-ron đầu ra. Lớp này sẽ tạo ra một đầu ra duy nhất cho mỗi mẫu dữ liệu.
- **input_shape=(4,):** Chỉ định hình dạng của đầu vào cho lớp đầu tiên.
- **activation='sigmoid':** Chỉ định hàm kích hoạt sigmoid cho lớp này. Hàm sigmoid biến đổi đầu ra của nơ-ron vào khoảng **[0, 1]**, làm cho lớp này thích hợp cho bài toán phân loại nhị phân (binary classification).

7. Biên dịch mô hình

```
# Biên dịch mô hình
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

- **Trình Tối ƯU**: Adam được sử dụng để điều chỉnh các trọng số.
- **Hàm Mất Mát**: Mean squared error (MSE) được sử dụng cho các bài toán hồi quy.
- **Đánh Giá**: Accuracy được sử dụng để đánh giá mô hình, mặc dù đối với bài toán hồi quy, các chỉ số khác có thể phù hợp hơn.

8. Huấn luyện mô hình

```
# Huấn luyện mô hình
model.fit(X_train, y_train, epochs=150, batch_size=32, verbose=0)
```

- **Huấn Luyện**: Mô hình với dữ liệu huấn luyện, sử dụng 150 epochs để mô hình học, với kích thước lô là 32, và không hiển thị thông báo tiến trình huấn luyện trên màn hình.

9. Đánh giá mô hình

```
# Đánh giá mô hình
score = model.evaluate(X_test, y_test, verbose=0)
```

model.evaluate: Đây là phương thức của mô hình Keras dùng để đánh giá hiệu suất của mô hình trên dữ liệu kiểm tra. Nó tính toán các chỉ số hiệu suất (chẳng hạn như mất mát và độ chính xác) dựa trên dữ liệu kiểm tra mà mô hình chưa thấy trong quá trình huấn luyện.
X_test: Đây là tập dữ liệu chứa các đặc trưng (features) của dữ liệu kiểm tra. Nó được sử dụng để dự đoán nhãn và tính toán các chỉ số hiệu suất của mô hình.

y_test: Đây là tập dữ liệu chứa các nhãn (labels) thực sự cho các đặc trưng trong **X_test**. Nó được so sánh với các dự đoán của mô hình để tính toán các chỉ số hiệu suất.
verbose=0: Tham số này kiểm soát mức độ chi tiết của thông tin được in ra trong quá trình đánh giá. verbose=0 có nghĩa là không in thông tin gì trong quá trình đánh giá.

10. Kết quả