Iris Dataset Classification using Neural Networks

Rushikesh Musale

June 13, 2025

What I learn form this task

- Firstly I learn to load dat data from csv file and process on it.
- Then I learn to build small feedforward nn.
- Also I learn how to convert strings into integers means flower names into 0,1,2.

Task 1: Data Preparation

We begin by importing the necessary libraries:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.optimizers import Adam
```

We load the Iris dataset and separate the features and labels:

```
data = pd.read_csv('Iris.csv')

X = data.iloc[:, 1:5].values  # Extracting features
y = data.iloc[:, 5].values    # Extracting labels
```

Next, we encode the string class labels into integers and convert them to one-hot encoded format suitable for classification.

```
lc = LabelEncoder()
y_int = lc.fit_transform(y)
y = to_categorical(y_int, num_classes=3)
```

The data is split into training and testing sets (80% training, 20% testing) using stratified sampling to maintain class balance.

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y_int)
```

The features are standardized using z-score normalization to improve model training.

```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test  = scaler.transform(X_test)
```

Task 2: Build the Neural Network

A simple feedforward neural network is created using Keras' Sequential API.

```
model = Sequential([
    Dense(8, activation='relu', input_shape=(4,)),
    Dense(3, activation='softmax')
])
```

The network has:

- One hidden layer with 8 neurons and ReLU activation.
- An output layer with 3 neurons (for 3 classes) and softmax activation.

Task 3: Compile, Train and Evaluate the Model

The model is compiled using the Adam optimizer and categorical crossentropy as the loss function.

```
model.compile(optimizer=Adam(),
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

The model is trained for 100 epochs with a batch size of 5.

```
train = model.fit(X_train, y_train,
                  epochs=100,
                  batch_size=5,
                  verbose=2)
```

Task 4: Evalution

Finally, we evaluate the model on the test set:

```
loss, acc = model.evaluate(X_test, y_test, verbose=2)
print(f"\nTest-set␣accuracy:␣{acc␣*␣100:.2f}%")
```

Conclusion

The neural network was able to learn from the Iris dataset effectively, achieving high classification accuracy on the test set. This demonstrates the applicability of even simple neural networks on well-structured datasets.