Homework 11/14 (4-4 Backpropagation)

This is an homework for a neural network course offered by the master's class of the Department of IEM at the NCUT in the first semester of the 2024 academic year (113-1).

Submitted by: 4B315021 詹家緯

The codes can be viewed on GitHub: https://github.com/chankai1016/113-1_Neural_Network/

Prepare

Complete the analysis of Iris Dataset using the Anaconda environment.

1. Install Anaconda

Anaconda official website

2. Create a virtual environment

Use Anaconda's virtual environment to isolate project dependencies.

Execute command in Anaconda Prompt:

```
conda create -n [YOUR_ENV_NAME] python=3.9
conda activate [YOUR_ENV_NAME]
```

The tensorflow library requires that the python version must be 3.9 or below

3. Install the required packages

```
conda install -c conda-forge tensorflow
conda install scikit-learn
```

4. Write and execute code

Working in Jupyter Notebook, install Jupyter Notebook:

```
conda install jupyter
jupyter notebook
```

Code

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder
import numpy as np

# Load the Iris dataset
iris = load_iris()
X = iris.data
y = iris.target

# One-Hot encode tags
encoder = OneHotEncoder(sparse_output=False)  # Updated argument
y_encoded = encoder.fit_transform(y.reshape(-1, 1))

# Split the data set into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y_encoded, test_size=0.2,
random_state=42)

# Build a neural network-like model
model = Sequential([
    Dense(10, input_dim=4, activation='relu'),
    Dense(10, activation='relu'),
    Dense(3, activation='softmax')
])

# Compile model
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# Training model
history = model.fit(X_train, y_train, epochs=50, batch_size=10,
validation_split=0.1)

# Evaluation model
loss, accuracy = model.evaluate(X_test, y_test)
print(f"測試集損失: {loss}")
print(f"測試集準確率: {accuracy}")

# Forecasting example
sample = np.array([[5.1, 3.5, 1.4, 0.2]])  # Iris-setosa example
prediction = model.predict(sample)
predicted_class = np.argmax(prediction)
print(f"預測類別: {iris.target_names[predicted_class]}")
```

Output

```
Epoch 1/50
11/11 [==============================] - 1s 42ms/step - loss: 1.1913 - accuracy:
0.3333 - val_loss: 0.9946 - val_accuracy: 0.4167
Epoch 2/50
11/11 [==============================] - 0s 8ms/step - loss: 1.1263 - accuracy:
0.3426 - val_loss: 0.9742 - val_accuracy: 0.4167
Epoch 3/50
11/11 [==============================] - 0s 8ms/step - loss: 1.0851 - accuracy:
0.3519 - val_loss: 0.9558 - val_accuracy: 0.4167
Epoch 4/50
11/11 [==============================] - 0s 8ms/step - loss: 1.0522 - accuracy:
0.3426 - val_loss: 0.9291 - val_accuracy: 0.4167
Epoch 5/50
11/11 [==============================] - 0s 8ms/step - loss: 1.0205 - accuracy:
0.3333 - val_loss: 0.8973 - val_accuracy: 0.4167
Epoch 6/50
11/11 [==============================] - 0s 8ms/step - loss: 0.9915 - accuracy:
0.3333 - val_loss: 0.8785 - val_accuracy: 0.4167
Epoch 7/50
11/11 [==============================] - 0s 8ms/step - loss: 0.9656 - accuracy:
0.3333 - val_loss: 0.8596 - val_accuracy: 0.4167
Epoch 8/50
11/11 [==============================] - 0s 8ms/step - loss: 0.9419 - accuracy:
0.3333 - val_loss: 0.8433 - val_accuracy: 0.4167
Epoch 9/50
11/11 [==============================] - 0s 9ms/step - loss: 0.9189 - accuracy:
0.3426 - val_loss: 0.8320 - val_accuracy: 0.4167
Epoch 10/50
11/11 [==============================] - 0s 8ms/step - loss: 0.8999 - accuracy:
0.3889 - val_loss: 0.8213 - val_accuracy: 0.4167
Epoch 11/50
11/11 [==============================] - 0s 8ms/step - loss: 0.8804 - accuracy:
0.5926 - val_loss: 0.7992 - val_accuracy: 0.8333
Epoch 12/50
11/11 [==============================] - 0s 8ms/step - loss: 0.8629 - accuracy:
0.7222 - val_loss: 0.7845 - val_accuracy: 0.8333
Epoch 13/50
11/11 [==============================] - 0s 8ms/step - loss: 0.8430 - accuracy:
0.8241 - val_loss: 0.7725 - val_accuracy: 1.0000
Epoch 14/50
11/11 [==============================] - 0s 8ms/step - loss: 0.8080 - accuracy:
0.8981 - val_loss: 0.7469 - val_accuracy: 1.0000
Epoch 15/50
11/11 [==============================] - 0s 8ms/step - loss: 0.7622 - accuracy:
0.9537 - val_loss: 0.7170 - val_accuracy: 1.0000
Epoch 16/50
11/11 [==============================] - 0s 8ms/step - loss: 0.7198 - accuracy:
0.9259 - val_loss: 0.6874 - val_accuracy: 1.0000
Epoch 17/50
11/11 [==============================] - 0s 8ms/step - loss: 0.6801 - accuracy:
0.9352 - val_loss: 0.6647 - val_accuracy: 1.0000
Epoch 18/50
11/11 [==============================] - 0s 8ms/step - loss: 0.6249 - accuracy:
```

```
0.9537 - val_loss: 0.5980 - val_accuracy: 1.0000
Epoch 19/50
11/11 [==============================] - 0s 11ms/step - loss: 0.5677 - accuracy:
0.9444 - val_loss: 0.5831 - val_accuracy: 1.0000
Epoch 20/50
11/11 [==============================] - 0s 8ms/step - loss: 0.5408 - accuracy:
0.9444 - val_loss: 0.5399 - val_accuracy: 1.0000
Epoch 21/50
11/11 [==============================] - 0s 8ms/step - loss: 0.5128 - accuracy:
0.9815 - val_loss: 0.5120 - val_accuracy: 1.0000
Epoch 22/50
11/11 [==============================] - 0s 8ms/step - loss: 0.4881 - accuracy:
0.9722 - val_loss: 0.4961 - val_accuracy: 1.0000
Epoch 23/50
11/11 [==============================] - 0s 8ms/step - loss: 0.4662 - accuracy:
0.9722 - val_loss: 0.4796 - val_accuracy: 1.0000
Epoch 24/50
11/11 [==============================] - 0s 8ms/step - loss: 0.4486 - accuracy:
0.9722 - val_loss: 0.4594 - val_accuracy: 1.0000
Epoch 25/50
11/11 [==============================] - 0s 8ms/step - loss: 0.4293 - accuracy:
0.9630 - val_loss: 0.4336 - val_accuracy: 1.0000
Epoch 26/50
11/11 [==============================] - 0s 9ms/step - loss: 0.4118 - accuracy:
0.9630 - val_loss: 0.4252 - val_accuracy: 1.0000
Epoch 27/50
11/11 [==============================] - 0s 8ms/step - loss: 0.3955 - accuracy:
0.9722 - val_loss: 0.4099 - val_accuracy: 1.0000
Epoch 28/50
11/11 [==============================] - 0s 10ms/step - loss: 0.3828 - accuracy:
0.9630 - val_loss: 0.3880 - val_accuracy: 1.0000
Epoch 29/50
11/11 [==============================] - 0s 8ms/step - loss: 0.3660 - accuracy:
0.9815 - val_loss: 0.3703 - val_accuracy: 1.0000
Epoch 30/50
11/11 [==============================] - 0s 8ms/step - loss: 0.3539 - accuracy:
0.9630 - val_loss: 0.3575 - val_accuracy: 1.0000
Epoch 31/50
11/11 [==============================] - 0s 8ms/step - loss: 0.3425 - accuracy:
0.9722 - val_loss: 0.3479 - val_accuracy: 1.0000
Epoch 32/50
11/11 [==============================] - 0s 8ms/step - loss: 0.3311 - accuracy:
0.9630 - val_loss: 0.3349 - val_accuracy: 1.0000
Epoch 33/50
11/11 [==============================] - 0s 9ms/step - loss: 0.3235 - accuracy:
0.9630 - val_loss: 0.3258 - val_accuracy: 1.0000
Epoch 34/50
11/11 [==============================] - 0s 8ms/step - loss: 0.3106 - accuracy:
0.9722 - val_loss: 0.3111 - val_accuracy: 1.0000
Epoch 35/50
11/11 [==============================] - 0s 8ms/step - loss: 0.3030 - accuracy:
0.9630 - val_loss: 0.3025 - val_accuracy: 1.0000
Epoch 36/50
11/11 [==============================] - 0s 8ms/step - loss: 0.2933 - accuracy:
```

```
0.9630 - val_loss: 0.2969 - val_accuracy: 1.0000
Epoch 37/50
11/11 [==============================] - 0s 9ms/step - loss: 0.2837 - accuracy:
0.9722 - val_loss: 0.2799 - val_accuracy: 1.0000
Epoch 38/50
11/11 [==============================] - 0s 9ms/step - loss: 0.2781 - accuracy:
0.9537 - val_loss: 0.2704 - val_accuracy: 1.0000
Epoch 39/50
11/11 [==============================] - 0s 10ms/step - loss: 0.2709 - accuracy:
0.9630 - val_loss: 0.2669 - val_accuracy: 1.0000
Epoch 40/50
11/11 [==============================] - 0s 8ms/step - loss: 0.2626 - accuracy:
0.9630 - val_loss: 0.2540 - val_accuracy: 1.0000
Epoch 41/50
11/11 [==============================] - 0s 8ms/step - loss: 0.2548 - accuracy:
0.9722 - val_loss: 0.2478 - val_accuracy: 1.0000
Epoch 42/50
11/11 [==============================] - 0s 9ms/step - loss: 0.2484 - accuracy:
0.9722 - val_loss: 0.2398 - val_accuracy: 1.0000
Epoch 43/50
11/11 [==============================] - 0s 8ms/step - loss: 0.2418 - accuracy:
0.9722 - val_loss: 0.2308 - val_accuracy: 1.0000
Epoch 44/50
11/11 [==============================] - 0s 8ms/step - loss: 0.2354 - accuracy:
0.9722 - val_loss: 0.2241 - val_accuracy: 1.0000
Epoch 45/50
11/11 [==============================] - 0s 8ms/step - loss: 0.2315 - accuracy:
0.9815 - val_loss: 0.2150 - val_accuracy: 1.0000
Epoch 46/50
11/11 [==============================] - 0s 8ms/step - loss: 0.2251 - accuracy:
0.9722 - val_loss: 0.2134 - val_accuracy: 1.0000
Epoch 47/50
11/11 [==============================] - 0s 8ms/step - loss: 0.2197 - accuracy:
0.9630 - val_loss: 0.2036 - val_accuracy: 1.0000
Epoch 48/50
11/11 [==============================] - 0s 9ms/step - loss: 0.2154 - accuracy:
0.9722 - val_loss: 0.1963 - val_accuracy: 1.0000
Epoch 49/50
11/11 [==============================] - 0s 8ms/step - loss: 0.2098 - accuracy:
0.9630 - val_loss: 0.1902 - val_accuracy: 1.0000
Epoch 50/50
11/11 [==============================] - 0s 8ms/step - loss: 0.2092 - accuracy:
0.9630 - val_loss: 0.1901 - val_accuracy: 1.0000
1/1 [==============================] - 0s 58ms/step - loss: 0.1979 - accuracy:
1.0000
測試集損失: 0.197854682803154
測試集準確率: 1.0
1/1 [==============================] - 0s 165ms/step
預測類別: setosa
```