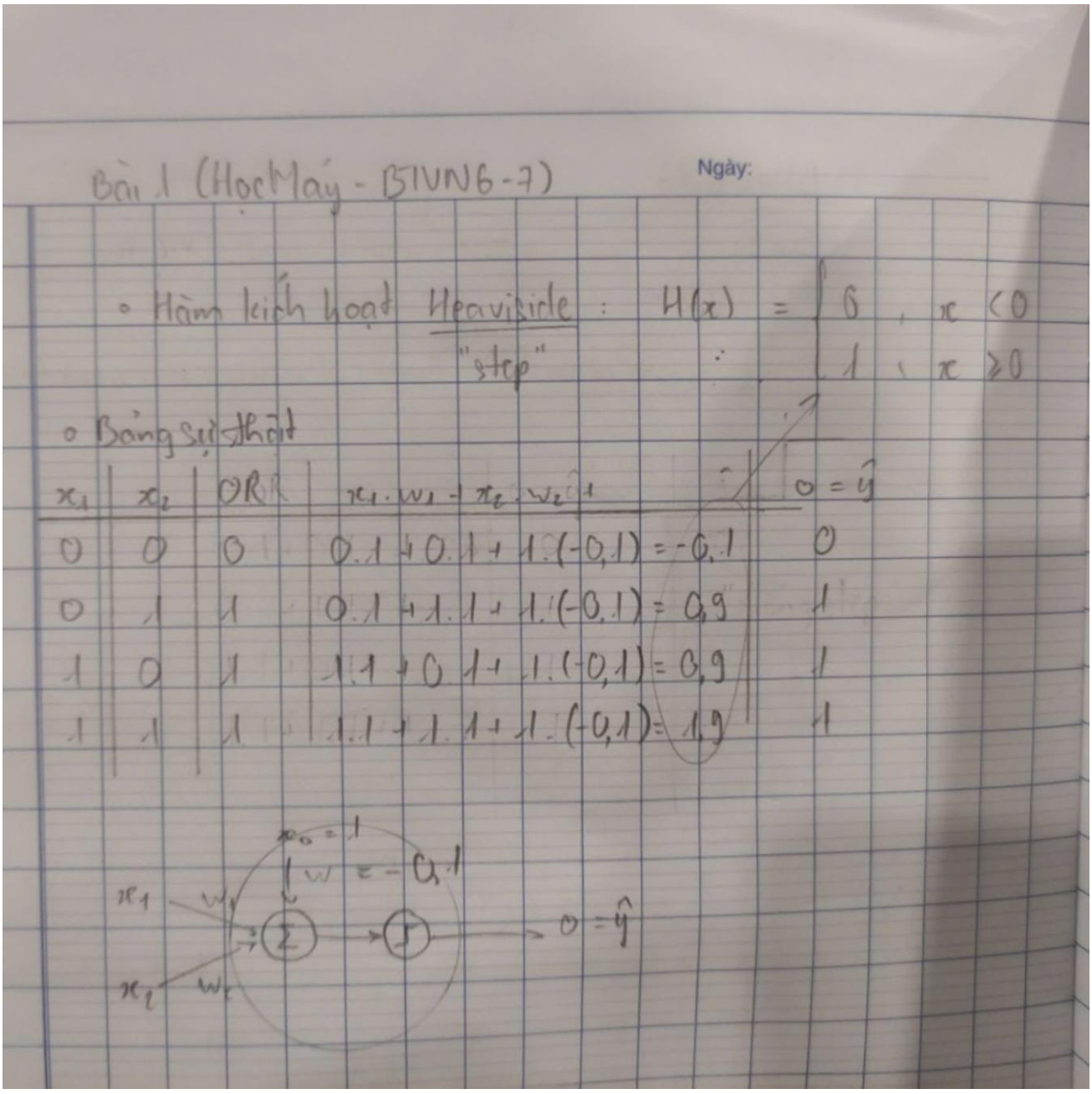
Nguyễn Khắc Sơn – MSSV : 21085691
Bài tập về nhà buổi 6 & 7

Bài 1: (Giải bài toán bằng bút và máy tính cầm tay)

Dựa trên ví dụ trên lớp về sử dụng perceptron để thực hiện một cổng AND với 02 đầu vào. Em hãy thiết kế một cổng OR với 02 đầu vào, sử dụng một perceptron với **hàm kích hoạt Heaviside**. Trình bày cách em lựa chọn các trọng số cho perceptron và vẽ thiết kế của em (có thể vẽ thêm bảng sự thật (truth table) cho hàm OR để hỗ trợ việc tính toán).



Bài 2: (Giải bài toán bằng bút và máy tính cầm tay)

Thiết kế một mạng neuron để thực hiện việc xấp xỉ một hàm phi tuyến cho bởi $f(x) = sin(6x)$ với $0 \leq x \leq 1$.

Gợi ý: Thiết kế và vẽ cấu trúc của mạng neuron với chú thích đầy đủ các thông số (ví dụ như: số neuron ở mỗi lớp, hàm kích hoạt đã sử dụng, giá trị của ma trận trong số khởi tạo đã dùng cho mỗi lớp). Trình bày và cho ví dụ về cách huấn luyện mạng neuron đã thiết kế.

Bài 3: (Thực hành với Python)

Trong ví dụ về việc sử dụng mạng neuron để phân lớp các thời trang (như áo. Mũ, túi, giày, dép, ...) sử dụng tập dữ liệu huấn luyện Fashion MNIST, tác giả Aurelien Geron đã xây dựng, huấn luyện và đánh giá mạng neuron nhiều lớp sử dụng Keras và TensorFlow. Ví dụ tham khảo có thể xem và download tại:

https://github.com/ageron/handson-ml3/blob/main/10_neural_nets_with_keras.ipynb

Dựa trên ví dụ của tác giả, em hãy

a) Tham khảo để hiểu cách xây dựng, huấn luyện và đánh giá một mạng neuron nhiều lớp sử dụng Keras và TensorFlow.

b) Sau đó, em hãy tự đề xuất một mạng neuron mới và thực hiện lại việc phân lớp các thời trang sử dụng tập dữ liệu huấn luyện Fashion MNIST. Đánh giá kết quả phân lớp khi sử dụng mạng neuron do em đề xuất.

a/ Tham khảo để hiểu cách xây dựng, huấn luyện

- **Dữ liệu Fashion MNIST:** Tập dữ liệu bao gồm 70,000 hình ảnh thời trang có nhãn từ 0 đến 9, với mỗi nhãn tương ứng với một loại trang phục (như áo, quần, giày, mũ, ...).
- **Mô hình Sequential:** Đây là loại mạng neuron dạng chuỗi, đơn giản và phổ biến.
- **Các lớp Dense (Fully Connected):** Các lớp neuron với số lượng neuron được chỉ định.
- **Hàm kích hoạt (Activation Function):** ReLU, softmax, sigmoid, v.v.
- **Loss Function:** Hàm mất mát để tối ưu hóa mô hình, thường là sparse_categorical_crossentropy cho bài toán phân loại.
- **Optimizer:** Bộ tối ưu hóa giúp cập nhật trọng số mạng neuron, phổ biến là Adam, SGD, RMSprop.
- Các tham số khác như số epoch, batch size, learning rate, v.v.

b/ Đề xuất mạng neutron mới

```
import tensorflow as tf
```

```python
from tensorflow.keras import layers, models
from tensorflow.keras.datasets import fashion_mnist

# Tải dữ liệu Fashion MNIST
(train_images, train_labels), (test_images, test_labels) =
fashion_mnist.load_data()

# Chuẩn hóa dữ liệu
train_images = train_images / 255.0
test_images = test_images / 255.0

# Xây dựng mô hình mạng neuron mới
model = models.Sequential([
    layers.Flatten(input_shape=(28, 28)),
    layers.Dense(512, activation='relu'),  # Lớp ẩn đầu tiên với 512 neuron
    layers.Dropout(0.5),  # Dropout để giảm overfitting
    layers.Dense(256, activation='relu'),  # Lớp ẩn thứ hai với 256 neuron
    layers.Dropout(0.5),
    layers.Dense(128, activation='relu'),  # Lớp ẩn thứ ba với 128 neuron
    layers.Dense(10, activation='softmax')  # Lớp đầu ra với 10 lớp phân loại
])

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

# Huấn luyện mô hình
model.fit(train_images, train_labels, epochs=10, batch_size=32,
validation_data=(test_images, test_labels))

# Đánh giá mô hình
test_loss, test_acc = model.evaluate(test_images, test_labels)
print(f"Test accuracy: {test_acc:.4f}")
```

Nếu kết quả chưa tốt:

- Tăng/giảm số lượng neuron hoặc lớp ẩn.
- Thay đổi hàm kích hoạt.
- Tăng số epoch hoặc điều chỉnh các tham số như learning rate trong bộ tối ưu hóa.

Bài 4: (Giải bài toán bằng lập trình)

Thiết kế một mạng neuron để thực hiện việc xấp xỉ một hàm phi tuyến cho bởi $f(x)=sin(6x)$ với $0 \leq x \leq 1$ sử dụng Keras và TensorFlow.

Gợi ý: Tự tạo ra bộ dữ liệu huấn luyện cho mạng neuron với số mẫu là 100. Sau khi huấn luyện được mạng neuron, vẽ kết quả đầu ra của mạng neuron để so sánh với hàm số cần xấp xỉ.

B1: Tạo dữ liệu huấn luyện

Sử dụng 100 mẫu giá trị x từ khoảng [0,1], sau đó tính f(x) = sin(6x) cho từng mẫu để tạo nhãn

B2: Xây dựng mạng neuron

Sử dụng Keras để xây dựng một mạng neuron nhiều lớp, với các lớp ẩn và hàm kích hoạt phi tuyến (như ReLU hoặc tanh), giúp mô hình học được tính phi tuyến của hàm f(x).

B3: Huấn luyện mô hình:

Sử dụng tập dữ liệu huấn luyện để huấn luyện mô hình.

B4: Vẽ biểu đồ:

So sánh kết quả đầu ra của mạng neuron với hàm thực sin(6x) bằng cách vẽ biểu đồ.

```
#Code đã cải thiện độ chính xác cao hơn, bám hơn


import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras import layers, models

# Tạo dữ liệu huấn luyện
x_train = np.linspace(0, 1, 500)  # Tăng số lượng mẫu lên 500
y_train = np.sin(6 * np.pi * x_train)

# Xây dựng mô hình với các cải tiến
model = models.Sequential([
    layers.Dense(128, activation='relu', input_shape=(1,)),
    layers.Dense(128, activation='relu'),
    layers.Dropout(0.2),  # Thêm Dropout để tránh overfitting
    layers.Dense(64, activation='relu'),
    layers.Dense(1)  # Lớp đầu ra
])

# Sử dụng Adam với learning rate nhỏ hơn
optimizer = tf.keras.optimizers.Adam(learning_rate=0.001)
model.compile(optimizer=optimizer, loss='mean_squared_error')

# Huấn luyện mô hình
model.fit(x_train, y_train, epochs=1000, verbose=0)  # Huấn luyện với 1000 epoch

# Dự đoán giá trị đầu ra
x_test = np.linspace(0, 1, 100)
y_pred = model.predict(x_test)
```

```
# Vẽ biểu đồ so sánh
plt.plot(x_test, np.sin(6 * np.pi * x_test), label='f(x) = sin(6πx)',
color='blue')
plt.plot(x_test, y_pred, label='Neural Network Approximation', color='red')
plt.legend()
plt.title('Comparison between f(x) and Neural Network Output')
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```

Để kết quả đầu ra của mạng neuron bám sát hơn với hàm f(x) = sin(6x):

1. Tăng số lượng neuron, cũng như số lượng lớp ẩn
2. Dùng hàm kích hoạt khác (ko đổi)
3. Tăng số lượng epoch (500 >> 1000)
4. Giảm tốc độ học (0.01 >> 0.001)
5. Tăng kích thước tập dữ liệu huấn luyện

Nếu mô hình chưa khớp với hàm cần xấp xỉ, cần điều chỉnh các tham số như số lượng lớp, số neuron, hàm kích hoạt, và learning rate...hoặc thêm thời gian huấn luyện (tăng số epoch) và tăng số lượng dữ liệu cũng sẽ giúp cải thiện hiệu suất.