

Họ và tên: Nguyễn Gia Thịnh  
MSSV: B2017081

### Thực hành 1:

- Tổng hợp các mã lệnh được trình bày trong phần Trên để xây dựng một mạng nơ ron perceptron đơn Tầng mô phỏng phép toán AND
- Khởi tạo ngẫu nhiên W và b thay vì gán sẵn
- Xem module tf.random
- Sau khi huấn luyện xong,

Mô hình cho đầu ra là 1 số

Thực từ 0 đến 1. Cần phải viết

Thêm phần xử lý để xác định

Nhân dự báo (so sánh với 0.5)

```
import tensorflow as tf

# Khởi tạo ngẫu nhiên W và b
W = tf.Variable(tf.random.normal((2, 1)))
b = tf.Variable(tf.random.normal(()))

# Định nghĩa perceptron model
@tf.function
def predict(X, W, b):
    return tf.nn.sigmoid(tf.matmul(X, W) + b)

# Định nghĩa hàm loss
@tf.function
def L(y, y_hat):
    return tf.reduce_mean((y - y_hat)**2)

# Khởi tạo dữ liệu đầu vào và đầu ra
X = tf.constant([[0.0, 0], [0, 1], [1, 0], [1, 1]])
y = tf.constant([[0.0], [0], [0], [1]])

alpha = 0.1
for it in range(500):
    with tf.GradientTape() as t:
        current_loss = L(y, predict(X, W, b))
        print("it", it, current_loss)
    dW, db = t.gradient(current_loss, [W, b])
    W.assign_sub(alpha * dW)
    b.assign_sub(alpha * db)
    y_hat = predict(X, W, b)

print(y_hat)

# Huấn luyện mô hình
# alpha = 0.1
```

### Kết quả:

```
+ Code + Text
it 466 tf.Tensor(0.0940089, shape=(), dtype=float32)
it 470 tf.Tensor(0.0938893, shape=(), dtype=float32)
it 471 tf.Tensor(0.093769155, shape=(), dtype=float32)
it 472 tf.Tensor(0.09364936, shape=(), dtype=float32)
it 473 tf.Tensor(0.09352992, shape=(), dtype=float32)
it 474 tf.Tensor(0.093410835, shape=(), dtype=float32)
it 475 tf.Tensor(0.09329212, shape=(), dtype=float32)
it 476 tf.Tensor(0.09317372, shape=(), dtype=float32)
it 477 tf.Tensor(0.09305567, shape=(), dtype=float32)
it 478 tf.Tensor(0.09293799, shape=(), dtype=float32)
it 479 tf.Tensor(0.092820644, shape=(), dtype=float32)
it 480 tf.Tensor(0.09270361, shape=(), dtype=float32)
it 481 tf.Tensor(0.09258696, shape=(), dtype=float32)
it 482 tf.Tensor(0.092470616, shape=(), dtype=float32)
it 483 tf.Tensor(0.09235464, shape=(), dtype=float32)
it 484 tf.Tensor(0.09223897, shape=(), dtype=float32)
it 485 tf.Tensor(0.09212363, shape=(), dtype=float32)
it 486 tf.Tensor(0.09200865, shape=(), dtype=float32)
it 487 tf.Tensor(0.09189397, shape=(), dtype=float32)
it 488 tf.Tensor(0.09177966, shape=(), dtype=float32)
it 489 tf.Tensor(0.09166563, shape=(), dtype=float32)
it 490 tf.Tensor(0.09155196, shape=(), dtype=float32)
it 491 tf.Tensor(0.09143859, shape=(), dtype=float32)
it 492 tf.Tensor(0.09132555, shape=(), dtype=float32)
it 493 tf.Tensor(0.09121285, shape=(), dtype=float32)
it 494 tf.Tensor(0.091100454, shape=(), dtype=float32)
it 495 tf.Tensor(0.090988375, shape=(), dtype=float32)
it 496 tf.Tensor(0.09087661, shape=(), dtype=float32)
it 497 tf.Tensor(0.09076516, shape=(), dtype=float32)
it 498 tf.Tensor(0.09065403, shape=(), dtype=float32)
it 499 tf.Tensor(0.09054321, shape=(), dtype=float32)
tf.Tensor(
[[0.11053396]
 [0.25511247]
 [0.35093823]
 [0.59841186]], shape=(4, 1), dtype=float32)
```

## Thực hành 2:

- Làm lại bài thực hành 1 với hàm lỗi binary cross entropy được định nghĩa như sau:

```
+ Code + Text
import tensorflow as tf

# Khởi tạo ngẫu nhiên W và b
W = tf.Variable(tf.random.normal((2, 1)))
b = tf.Variable(tf.random.normal(()))

# Định nghĩa perceptron model
@tf.function
def predict(X, W, b):
    return tf.nn.sigmoid(tf.matmul(X, W) + b)

# Định nghĩa hàm loss
@tf.function
def L(y, y_hat):
    # return tf.reduce_mean((y - y_hat)**2)
    epsilon = 1e-15 # Để tránh log(0)
    y_hat = tf.clip_by_value(y_hat, epsilon, 1 - epsilon)
    return -tf.reduce_mean(y * tf.math.log(y_hat) + (1 - y) * tf.math.log(1 - y_hat))

# Khởi tạo dữ liệu đầu vào và đầu ra
X = tf.constant([[0.0, 0], [0, 1], [1, 0], [1, 1]])
y = tf.constant([[0.0], [0], [0], [1]])

alpha = 0.1
for it in range(500):
    with tf.GradientTape() as t:
        current_loss = L(y, predict(X, W, b))
        print("it", it, current_loss)
    dW, db = t.gradient(current_loss, [W, b])
    W.assign_sub(alpha * dW)
    b.assign_sub(alpha * db)
    y_hat = predict(X, W, b)

print(y_hat)
```

## Kết quả:

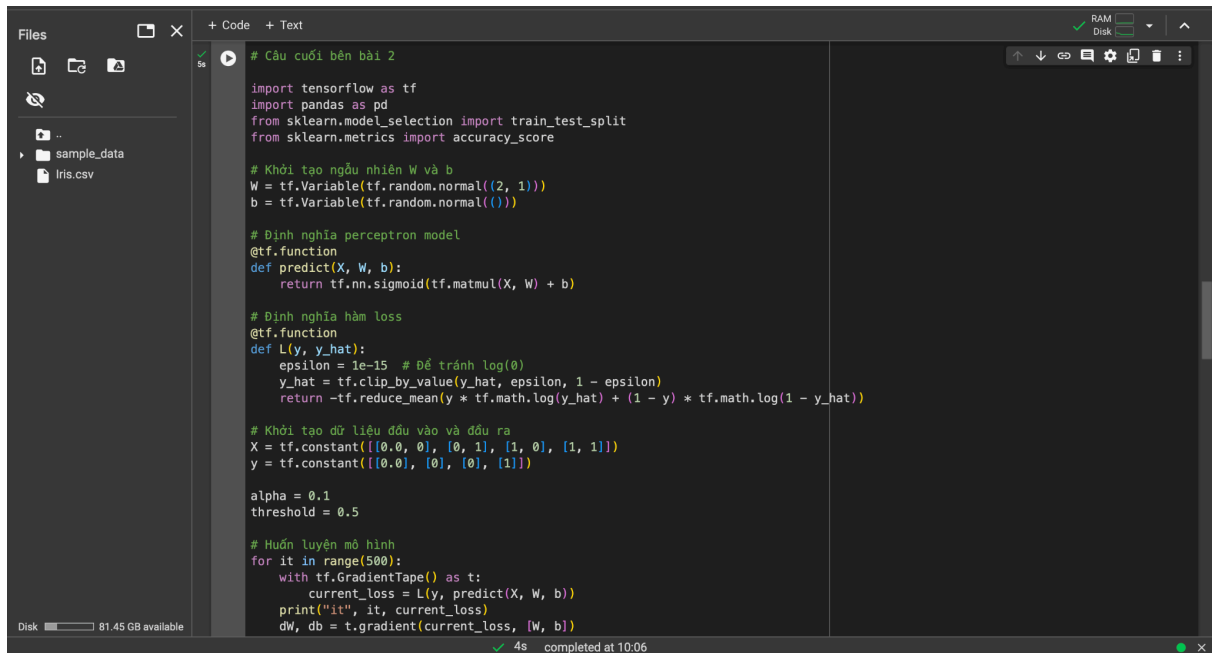
```
+ Code + Text
it 465 tf.Tensor(0.23119503, shape=(), dtype=float32)
it 466 tf.Tensor(0.23087433, shape=(), dtype=float32)
it 467 tf.Tensor(0.23058982, shape=(), dtype=float32)
it 468 tf.Tensor(0.23030506, shape=(), dtype=float32)
it 469 tf.Tensor(0.23002306, shape=(), dtype=float32)
it 470 tf.Tensor(0.22974089, shape=(), dtype=float32)
it 471 tf.Tensor(0.2294594, shape=(), dtype=float32)
it 472 tf.Tensor(0.22917871, shape=(), dtype=float32)
it 473 tf.Tensor(0.22889876, shape=(), dtype=float32)
it 474 tf.Tensor(0.22861956, shape=(), dtype=float32)
it 475 tf.Tensor(0.22834113, shape=(), dtype=float32)
it 476 tf.Tensor(0.22806343, shape=(), dtype=float32)
it 477 tf.Tensor(0.22778651, shape=(), dtype=float32)
it 478 tf.Tensor(0.2275103, shape=(), dtype=float32)
it 479 tf.Tensor(0.22723483, shape=(), dtype=float32)
it 480 tf.Tensor(0.22696006, shape=(), dtype=float32)
it 481 tf.Tensor(0.22668603, shape=(), dtype=float32)
it 482 tf.Tensor(0.22641277, shape=(), dtype=float32)
it 483 tf.Tensor(0.22614019, shape=(), dtype=float32)
it 484 tf.Tensor(0.22586831, shape=(), dtype=float32)
it 485 tf.Tensor(0.22559714, shape=(), dtype=float32)
it 486 tf.Tensor(0.22532672, shape=(), dtype=float32)
it 487 tf.Tensor(0.22505704, shape=(), dtype=float32)
it 488 tf.Tensor(0.22478803, shape=(), dtype=float32)
it 489 tf.Tensor(0.22451967, shape=(), dtype=float32)
it 490 tf.Tensor(0.22425207, shape=(), dtype=float32)
it 491 tf.Tensor(0.22398517, shape=(), dtype=float32)
it 492 tf.Tensor(0.22371897, shape=(), dtype=float32)
it 493 tf.Tensor(0.22345343, shape=(), dtype=float32)
it 494 tf.Tensor(0.22318858, shape=(), dtype=float32)
it 495 tf.Tensor(0.22292441, shape=(), dtype=float32)
it 496 tf.Tensor(0.22266094, shape=(), dtype=float32)
it 497 tf.Tensor(0.2223981, shape=(), dtype=float32)
it 498 tf.Tensor(0.22213599, shape=(), dtype=float32)
it 499 tf.Tensor(0.22187454, shape=(), dtype=float32)
tf.Tensor(
[[0.93189765]
 [0.193494 ]
 [0.23138237]
 [0.68671775]], shape=(4, 1), dtype=float32)
```

## Thực hành 3

- Làm lại bài phân loại hoa iris (2 lớp) với Tensorflow thay vì dùng Keras
- Sử dụng hàm lỗi Binary crossentropy
- Cần phân ngưỡng kết quả đầu ra để có được nhãn

chính xác

- Tính độ chính xác phân lớp bằng cách so sánh nhãn dự báo và nhãn mong muốn
- Có thể dùng hàm `Tensor.numpy()` để lấy giá trị của Tensor về dạng numpy để xử lý.



```
# Câu cuối bên bài 2

import tensorflow as tf
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Khởi tạo ngẫu nhiên W và b
W = tf.Variable(tf.random.normal((2, 1)))
b = tf.Variable(tf.random.normal((1)))

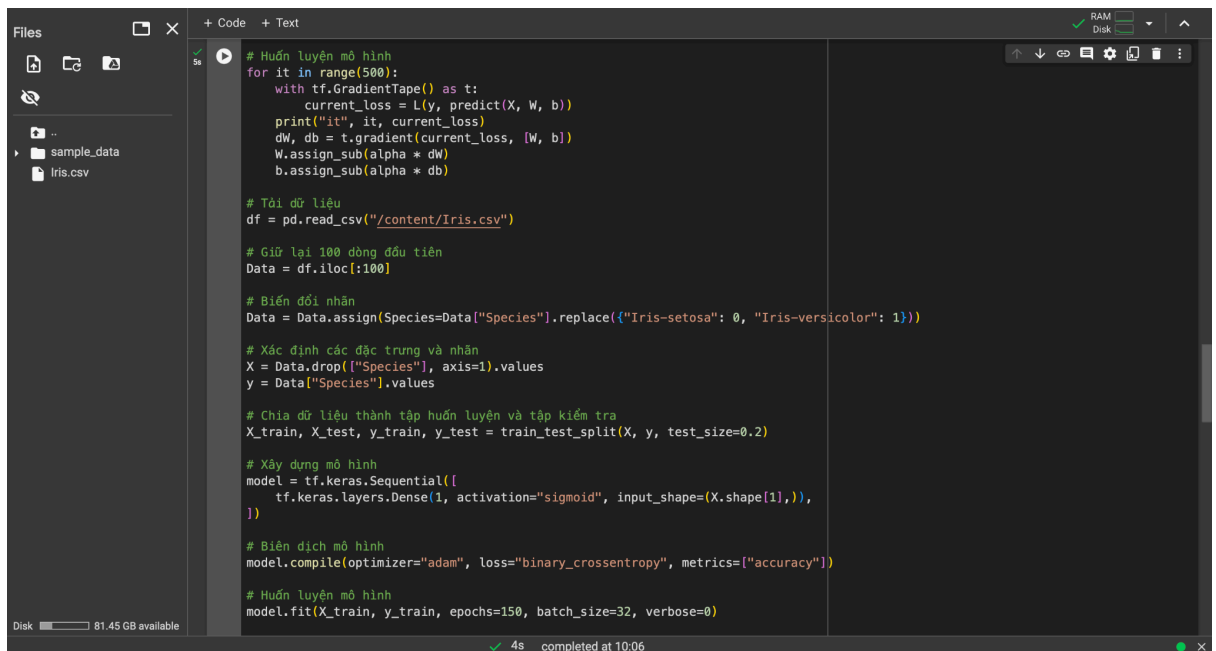
# Định nghĩa perceptron model
@tf.function
def predict(X, W, b):
    return tf.nn.sigmoid(tf.matmul(X, W) + b)

# Định nghĩa hàm loss
@tf.function
def L(y, y_hat):
    epsilon = 1e-15 # Để tránh log(0)
    y_hat = tf.clip_by_value(y_hat, epsilon, 1 - epsilon)
    return -tf.reduce_mean(y * tf.math.log(y_hat) + (1 - y) * tf.math.log(1 - y_hat))

# Khởi tạo dữ liệu đầu vào và đầu ra
X = tf.constant([[0.0, 0], [0, 1], [1, 0], [1, 1]])
y = tf.constant([[0.0], [0], [0], [1]])

alpha = 0.1
threshold = 0.5

# Huấn luyện mô hình
for it in range(500):
    with tf.GradientTape() as t:
        current_loss = L(y, predict(X, W, b))
    print("it", it, current_loss)
    dW, db = t.gradient(current_loss, [W, b])
    W.assign_sub(alpha * dW)
    b.assign_sub(alpha * db)
```



```
# Huấn luyện mô hình
for it in range(500):
    with tf.GradientTape() as t:
        current_loss = L(y, predict(X, W, b))
    print("it", it, current_loss)
    dW, db = t.gradient(current_loss, [W, b])
    W.assign_sub(alpha * dW)
    b.assign_sub(alpha * db)

# Tải dữ liệu
df = pd.read_csv("/content/Iris.csv")

# Giữ lại 100 dòng đầu tiên
Data = df.iloc[:100]

# Biến đổi nhãn
Data = Data.assign(Species=Data["Species"].replace({"Iris-setosa": 0, "Iris-versicolor": 1}))

# Xác định các đặc trưng và nhãn
X = Data.drop(["Species"], axis=1).values
y = Data["Species"].values

# Chia dữ liệu thành tập huấn luyện và tập kiểm tra
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Xây dựng mô hình
model = tf.keras.Sequential([
    tf.keras.layers.Dense(1, activation="sigmoid", input_shape=(X.shape[1],)),
])

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

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

Kết quả:

The screenshot displays a Jupyter Notebook environment. The left sidebar shows the file explorer with a folder named 'sample\_data' containing 'Iris.csv'. The top bar indicates the current file is 'Code + Text'. The main area shows a Python script with the following code:

```
# Đánh giá mô hình
y_hat_tf = model.predict(X_test)
predicted_labels_tf = (y_hat_tf > threshold).astype(int)

# Chuyển đổi nhãn thực tế sang dạng numpy array
true_labels = y_test

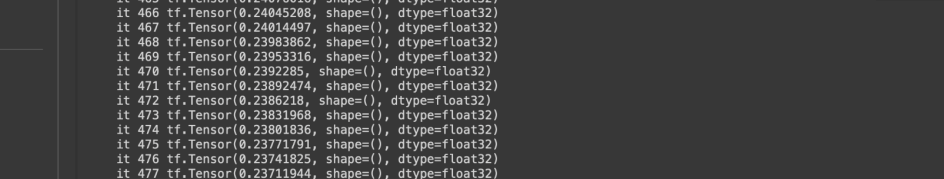
# Tính độ chính xác sử dụng scikit-learn
accuracy_tf = accuracy_score(true_labels, predicted_labels_tf)

print("Độ chính xác TensorFlow:", accuracy_tf)
```

The output of the script is a list of 48 tf.Tensor objects, each representing a prediction for a specific instance. The output is truncated at the bottom, showing the last few lines of the list.

```
it 443 tf.Tensor(0.24776164, shape=(), dtype=float32)
it 444 tf.Tensor(0.24743374, shape=(), dtype=float32)
it 445 tf.Tensor(0.24710676, shape=(), dtype=float32)
it 446 tf.Tensor(0.24678075, shape=(), dtype=float32)
it 447 tf.Tensor(0.24645565, shape=(), dtype=float32)
it 448 tf.Tensor(0.24613151, shape=(), dtype=float32)
it 449 tf.Tensor(0.24580829, shape=(), dtype=float32)
it 450 tf.Tensor(0.24548602, shape=(), dtype=float32)
it 451 tf.Tensor(0.24516469, shape=(), dtype=float32)
it 452 tf.Tensor(0.24484423, shape=(), dtype=float32)
it 453 tf.Tensor(0.24452469, shape=(), dtype=float32)
it 454 tf.Tensor(0.24420607, shape=(), dtype=float32)
it 455 tf.Tensor(0.24388835, shape=(), dtype=float32)
it 456 tf.Tensor(0.24357158, shape=(), dtype=float32)
it 457 tf.Tensor(0.24325567, shape=(), dtype=float32)
it 458 tf.Tensor(0.24294063, shape=(), dtype=float32)
it 459 tf.Tensor(0.2426265, shape=(), dtype=float32)
it 460 tf.Tensor(0.24231327, shape=(), dtype=float32)
it 461 tf.Tensor(0.2420009, shape=(), dtype=float32)
it 462 tf.Tensor(0.24168938, shape=(), dtype=float32)
it 463 tf.Tensor(0.24137877, shape=(), dtype=float32)
it 464 tf.Tensor(0.24106902, shape=(), dtype=float32)
it 465 tf.Tensor(0.24076016, shape=(), dtype=float32)
it 466 tf.Tensor(0.24045208, shape=(), dtype=float32)
it 467 tf.Tensor(0.24014497, shape=(), dtype=float32)
it 468 tf.Tensor(0.23983862, shape=(), dtype=float32)
```

The bottom status bar shows the system is running on a disk with 81.45 GB available and the notebook is completed at 10:06.



The screenshot shows a Jupyter Notebook interface. On the left is a file explorer with a folder named 'sample\_data' containing a file 'Iris.csv'. The main area displays a list of 49 TensorFlow tensors, each with its shape and data type. The tensors are named 'it' followed by a number from 464 to 499. Each tensor has a shape of () and a dtype of float32. At the bottom, there is a status bar indicating '4s completed at 10:06'.

```
it 464 tf.Tensor(0.24106902, shape=(), dtype=float32)
it 465 tf.Tensor(0.24076016, shape=(), dtype=float32)
it 466 tf.Tensor(0.24045208, shape=(), dtype=float32)
it 467 tf.Tensor(0.24014497, shape=(), dtype=float32)
it 468 tf.Tensor(0.23983862, shape=(), dtype=float32)
it 469 tf.Tensor(0.23953316, shape=(), dtype=float32)
it 470 tf.Tensor(0.2392285, shape=(), dtype=float32)
it 471 tf.Tensor(0.23892474, shape=(), dtype=float32)
it 472 tf.Tensor(0.2386218, shape=(), dtype=float32)
it 473 tf.Tensor(0.23831968, shape=(), dtype=float32)
it 474 tf.Tensor(0.23801836, shape=(), dtype=float32)
it 475 tf.Tensor(0.23771791, shape=(), dtype=float32)
it 476 tf.Tensor(0.23741825, shape=(), dtype=float32)
it 477 tf.Tensor(0.23711944, shape=(), dtype=float32)
it 478 tf.Tensor(0.23682144, shape=(), dtype=float32)
it 479 tf.Tensor(0.23652424, shape=(), dtype=float32)
it 480 tf.Tensor(0.23622783, shape=(), dtype=float32)
it 481 tf.Tensor(0.23593223, shape=(), dtype=float32)
it 482 tf.Tensor(0.23563743, shape=(), dtype=float32)
it 483 tf.Tensor(0.2353434, shape=(), dtype=float32)
it 484 tf.Tensor(0.23505026, shape=(), dtype=float32)
it 485 tf.Tensor(0.23475778, shape=(), dtype=float32)
it 486 tf.Tensor(0.23446615, shape=(), dtype=float32)
it 487 tf.Tensor(0.2341753, shape=(), dtype=float32)
it 488 tf.Tensor(0.23388524, shape=(), dtype=float32)
it 489 tf.Tensor(0.23359591, shape=(), dtype=float32)
it 490 tf.Tensor(0.23330739, shape=(), dtype=float32)
it 491 tf.Tensor(0.2330196, shape=(), dtype=float32)
it 492 tf.Tensor(0.23273262, shape=(), dtype=float32)
it 493 tf.Tensor(0.23244634, shape=(), dtype=float32)
it 494 tf.Tensor(0.23216087, shape=(), dtype=float32)
it 495 tf.Tensor(0.23187615, shape=(), dtype=float32)
it 496 tf.Tensor(0.23159218, shape=(), dtype=float32)
it 497 tf.Tensor(0.23130897, shape=(), dtype=float32)
it 498 tf.Tensor(0.23102644, shape=(), dtype=float32)
it 499 tf.Tensor(0.23074469, shape=(), dtype=float32)
1/1 [=====] - 0s 69ms/step
Đồ chình xác TensorFlow: 0.6
```