# DEEP LEARNING

## Perceptron đa tầng (MLP)

**Phạm Nguyên Khang**
*pnkhang@cit.ctu.edu.vn*

CAN THO, 22/12/2022

## Nội dung

- Tensor là gì?
- Đồ thị tính toán
- Cài đặt Perceptron bằng Tensorflow



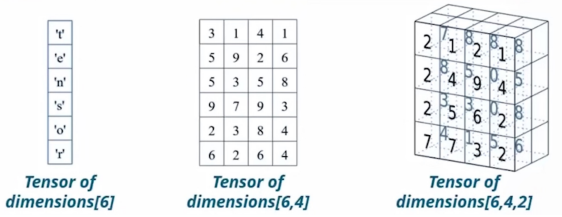
## Tensorflow

- Tensors là phương pháp biểu diễn dữ liệu chuẩn trong học sâu
- Tensors là các mảng nhiều chiều



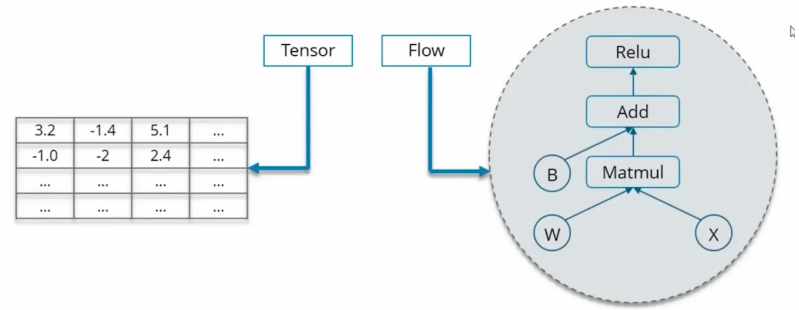
## Tensorflow

- Tensors là các mảng nhiều nhiều
  - Số (0 chiều)
  - Vector (1 chiều)
  - Ma trận (2 chiều)
  - Mảng nhiều chiều (n-d array)



# Tensorflow

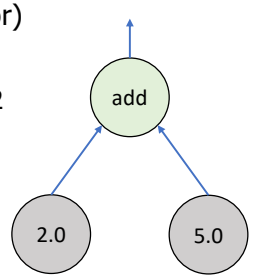
- Tensorflow (luồng tensors)
  - Trong Tensorflow các tính toán được biểu diễn dưới dạng đồ thị luồng dữ liệu (dataflow graph) hay đồ thị tính toán (computation graph)






# Tensorflow

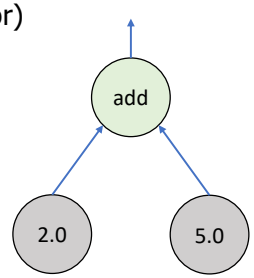
- Đồ thị tính toán (computational graph)
  - Các biểu thức tính toán được tổ chức dưới dạng đồ thị
    - Nút/đỉnh nhận 0/nhiều đầu vào (inputs), tính toán và cho ra 1 đầu ra (output)
  - Nút trong đồ thị tính toán (tf.Tensor)
    - Hình dạng (shape)
    - Kiểu dữ liệu (dtype): float32, int32






# Tensorflow

- Đồ thị tính toán (computational graph)
  - Các biểu thức tính toán được tổ chức dưới dạng đồ thị
    - Nút/đỉnh nhận 0/nhiều đầu vào (inputs), tính toán và cho ra 1 đầu ra (output)
  - Nút trong đồ thị tính toán (tf.Tensor)
    - Hằng (tf.constant)
    - Biến (tf.Variable)
    - Phép toán (tf.Operation)
    - Hàm (tf.function)





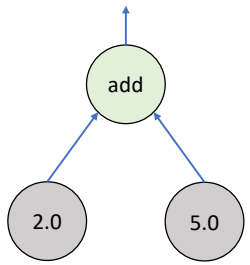

# Tensorflow

- Đồ thị tính toán (computational graph)
  - Các biểu thức tính toán được tổ chức dưới dạng đồ thị
    - Nút/đỉnh nhận 0/nhiều đầu vào (inputs), tính toán và cho ra 1 đầu ra (output)

```
import tensorflow as tf

a = tf.constant(2.0, tf.float32)
b = tf.constant(5.0, tf.float32)
c = tf.add(a, b)
print(a, b, c)
```

```
tf.Tensor(2.0, shape=(), dtype=float32)
tf.Tensor(5.0, shape=(), dtype=float32)
tf.Tensor(7.0, shape=(), dtype=float32)
```



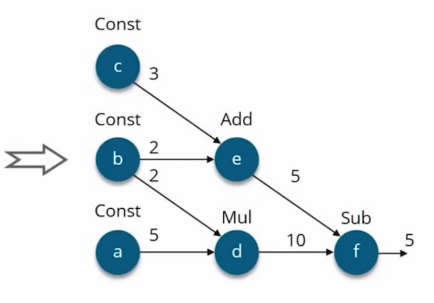
# Tensorflow

- Ví dụ

```
import tensorflow as tf

a = tf.constant(5)
b = tf.constant(2)
c = tf.constant(3)

d = tf.multiply(a, b)
e = tf.add(c, b)
f = tf.subtract(d, e)
```





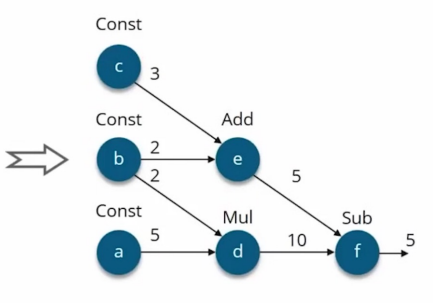

# Tensorflow

- Ví dụ

```
import tensorflow as tf

a = tf.constant(5)
b = tf.constant(2)
c = tf.constant(3)

d = a * b
e = c + b
f = d - e
```






# Tensorflow

- Sử dụng hàm trong Tensorflow
  - Hàm tương đương với một đồ thị tính toán
    - Tham số: input
    - Giá trị trả về: output
  - Hàm có thể được dùng như một nút trong đồ thị tính toán



# Tensorflow

- Sử dụng hàm trong Tensorflow

```
import tensorflow as tf

def my_add(x, y):
    return x + y

a = tf.constant(5)
b = tf.constant(2)

my_func = tf.function(my_add)
print(my_func)
c = my_func(a, b)
print(c)
```

```
<tensorflow.python.eager.def_function.Function object at 0x115b7b5e0>
tf.Tensor(7, shape=(), dtype=int32)
```

# Tensorflow

- Sử dụng hàm trong Tensorflow

```
import tensorflow as tf

@tf.function
def my_func(x, y):
    return x + y

a = tf.constant(5)
b = tf.constant(2)

print(my_func)
c = my_func(a, b)
print(c)
```

<tensorflow.python.eager.def_function.Function object at 0x115b7b5e0>
tf.Tensor(7, shape=(), dtype=int32)



# Tensorflow

- Sử dụng hàm trong Tensorflow

```
import tensorflow as tf

@tf.function
def h(x, w1, w0):
    return x * w1 + w0

x = tf.constant(5.0)
y = tf.constant(10.0)

w0 = tf.Variable(1.0)
w1 = tf.Variable(2.5)
diff = h(x, w1, w0) - y
```

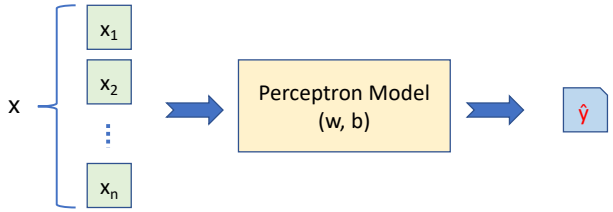
Vẽ đồ thị tính toán tương ứng với đoạn mã bên trái

tf.Tensor(3.5, shape=(), dtype=float32)



# Cài đặt Perceptron bằng TF

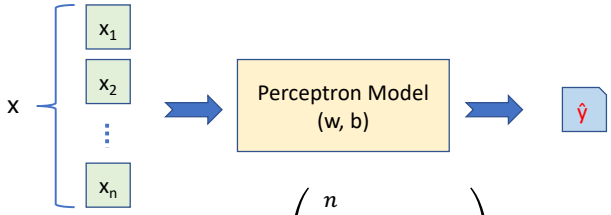
- Perceptron nhận đầu vào x và sinh ra ŷ






# Cài đặt Perceptron bằng TF

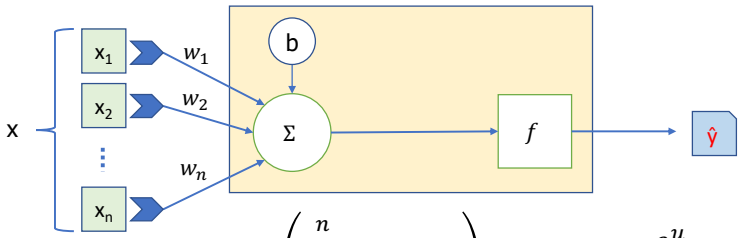
- Cài đặt Perceptron như
  - Một hàm nhận đầu vào x, tính toán và cho ra đầu ra ŷ.




$$\hat{y} = f\left(\sum_{j=1}^{n} x_j w_j + b\right) \qquad f(u) = \frac{e^u}{e^u + 1}$$

# Cài đặt Perceptron bằng TF

- Cài đặt Perceptron như
  - Một hàm nhận đầu vào x, tính toán và cho ra đầu ra ŷ.

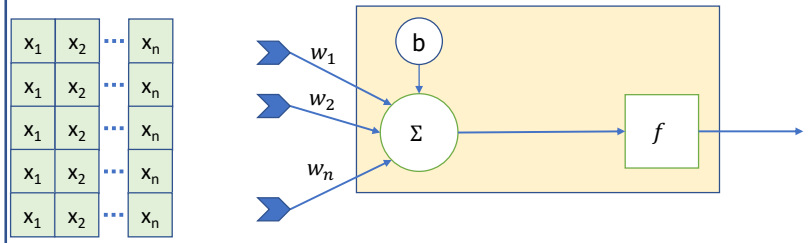



$$\hat{y} = f\left(\sum_{j=1}^{n} x_j w_j + b\right) \qquad f(u) = \frac{e^u}{e^u + 1}$$



# Cài đặt Perceptron bằng TF

- Cài đặt Perceptron như
  - Một hàm nhận đầu vào x, tính toán và cho ra đầu ra ŷ.



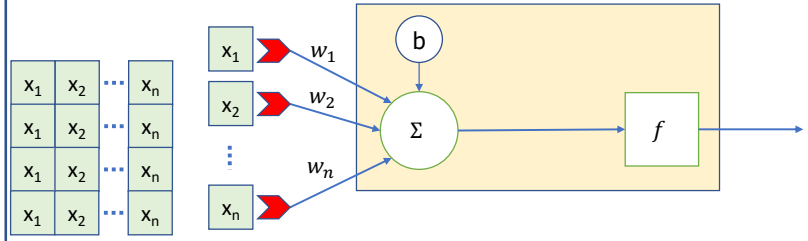

Thực tế, mỗi mô hình deep learning nhận đầu vào là **một lô (batch)** dữ liệu đầu vào chứ không phải 1 phần tử dữ liệu.

Vì thế cần định nghĩa hàm tính toán của mô hình trên toàn bộ lô dữ liệu này.



# Cài đặt Perceptron bằng TF

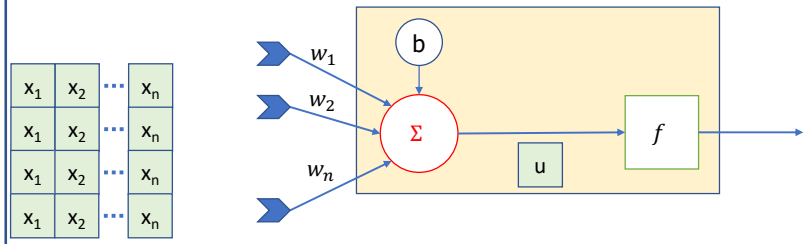
- Cài đặt Perceptron như
  - Một hàm nhận đầu vào x, tính toán và cho ra đầu ra ŷ.





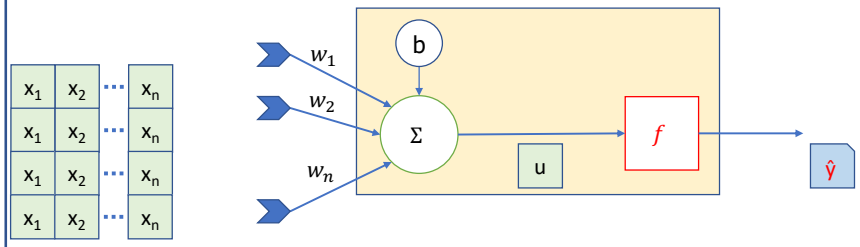

# Cài đặt Perceptron bằng TF

- Cài đặt Perceptron như
  - Một hàm nhận đầu vào x, tính toán và cho ra đầu ra ŷ.



# Cài đặt Perceptron bằng TF

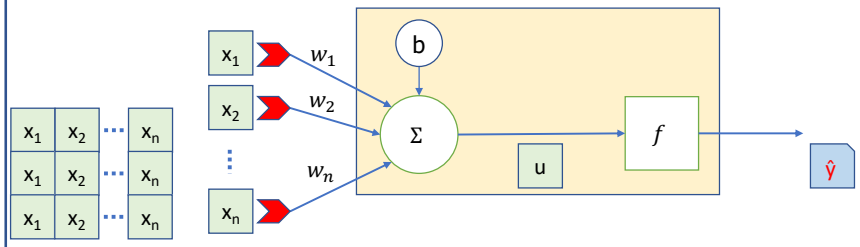
- Cài đặt Perceptron như
  - Một hàm nhận đầu vào x, tính toán và cho ra đầu ra ŷ.






# Cài đặt Perceptron bằng TF

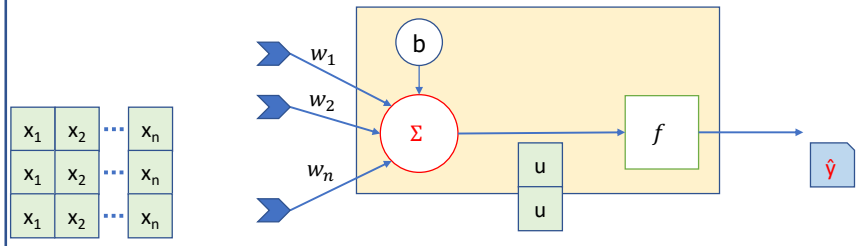
- Cài đặt Perceptron như
  - Một hàm nhận đầu vào x, tính toán và cho ra đầu ra ŷ.






# Cài đặt Perceptron bằng TF

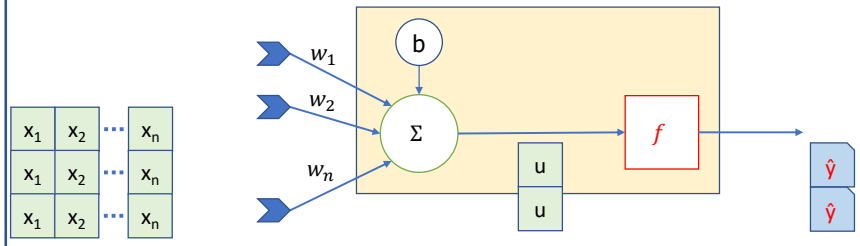
- Cài đặt Perceptron như
  - Một hàm nhận đầu vào x, tính toán và cho ra đầu ra ŷ.





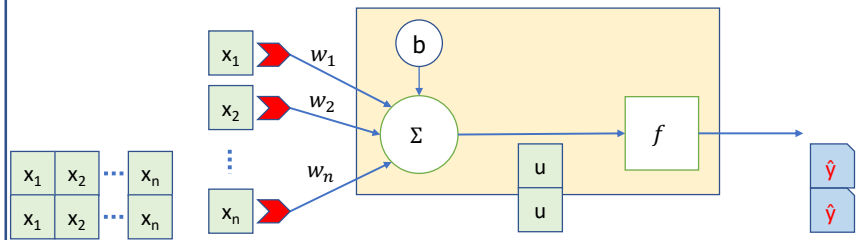

# Cài đặt Perceptron bằng TF

- Cài đặt Perceptron như
  - Một hàm nhận đầu vào x, tính toán và cho ra đầu ra ŷ.



# Cài đặt Perceptron bằng TF

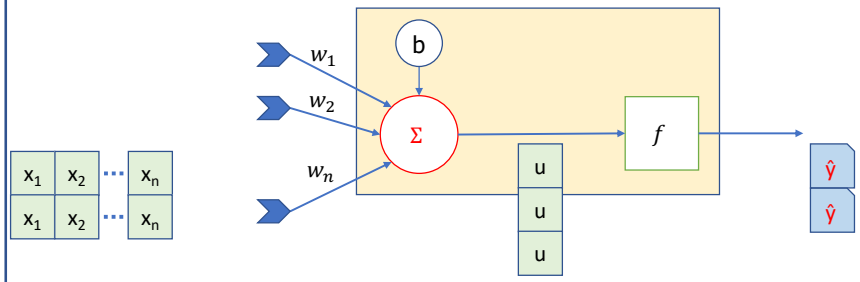
- Cài đặt Perceptron như
  - Một hàm nhận đầu vào x, tính toán và cho ra đầu ra ŷ.






# Cài đặt Perceptron bằng TF

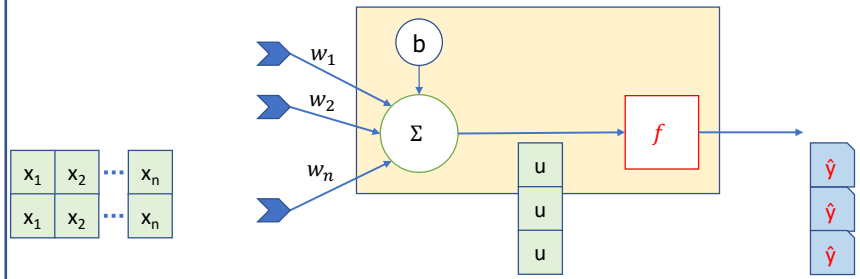
- Cài đặt Perceptron như
  - Một hàm nhận đầu vào x, tính toán và cho ra đầu ra ŷ.






# Cài đặt Perceptron bằng TF

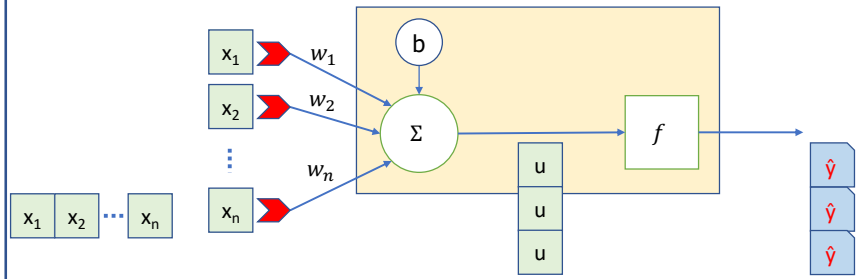
- Cài đặt Perceptron như
  - Một hàm nhận đầu vào x, tính toán và cho ra đầu ra ŷ.





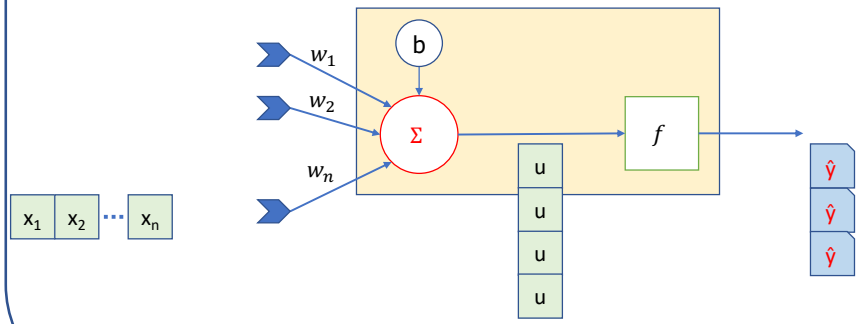

# Cài đặt Perceptron bằng TF

- Cài đặt Perceptron như
  - Một hàm nhận đầu vào x, tính toán và cho ra đầu ra ŷ.



# Cài đặt Perceptron bằng TF

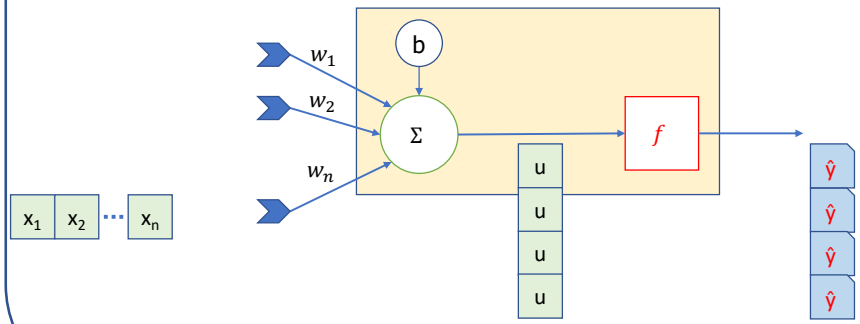
- Cài đặt Perceptron như
  - Một hàm nhận đầu vào x, tính toán và cho ra đầu ra ŷ.






# Cài đặt Perceptron bằng TF

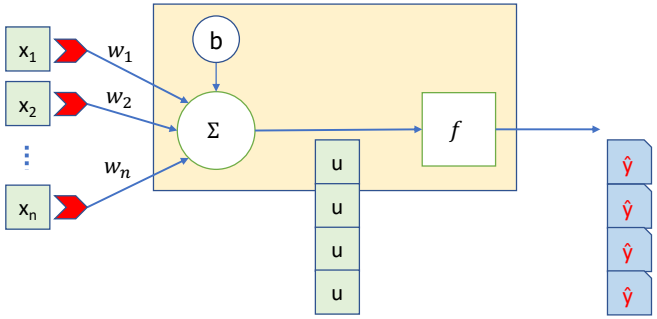
- Cài đặt Perceptron như
  - Một hàm nhận đầu vào x, tính toán và cho ra đầu ra ŷ.






# Cài đặt Perceptron bằng TF

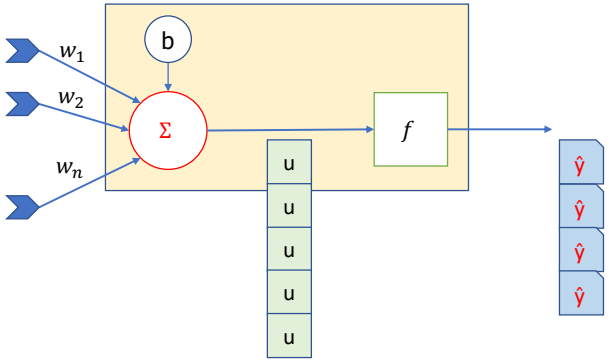
- Cài đặt Perceptron như
  - Một hàm nhận đầu vào x, tính toán và cho ra đầu ra ŷ.





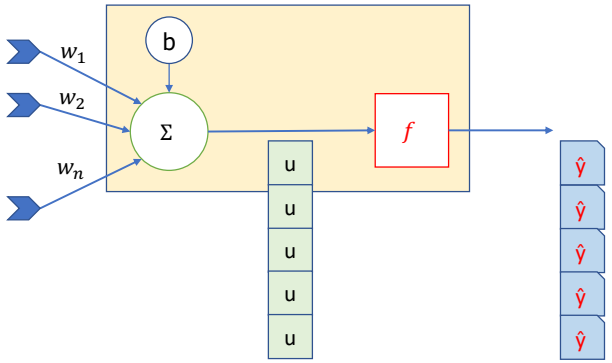

# Cài đặt Perceptron bằng TF

- Cài đặt Perceptron như
  - Một hàm nhận đầu vào x, tính toán và cho ra đầu ra ŷ.



# Cài đặt Perceptron bằng TF

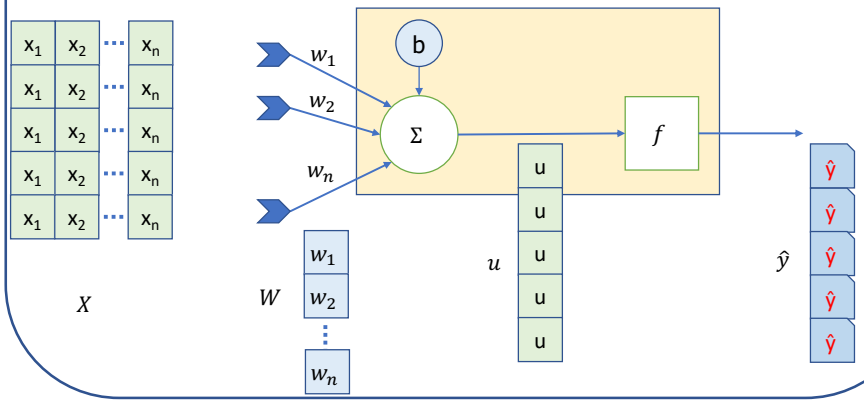
- Cài đặt Perceptron như
  - Một hàm nhận đầu vào x, tính toán và cho ra đầu ra ŷ.






# Cài đặt Perceptron bằng TF

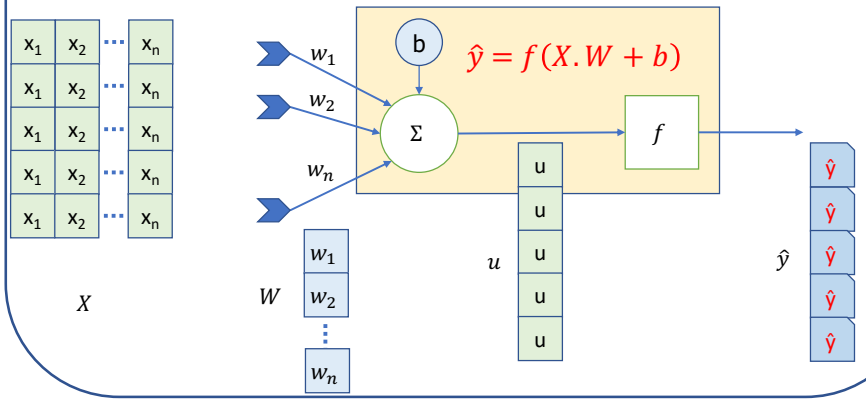
- Cài đặt Perceptron như
  - Một hàm nhận đầu vào x, tính toán và cho ra đầu ra ŷ.






# Cài đặt Perceptron bằng TF

- Cài đặt Perceptron như
  - Một hàm nhận đầu vào x, tính toán và cho ra đầu ra ŷ.



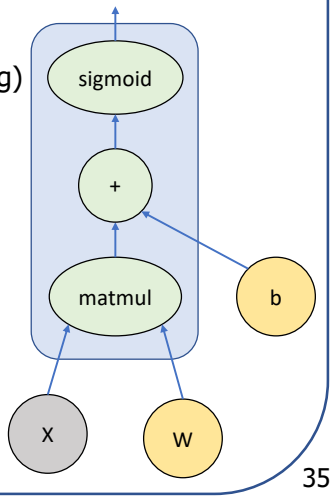



# Cài đặt Perceptron bằng TF

- Đồ thị tính toán của Perceptron
  - X: ma trận dữ liệu huấn luyện (hằng)
  - W: ma trận trọng số (biến)
  - b: độ lệch (biến)

$$\hat{y} = f(X.W + b)$$

$$f(u) = \frac{e^u}{e^u + 1}$$



# Cài đặt Perceptron bằng TF

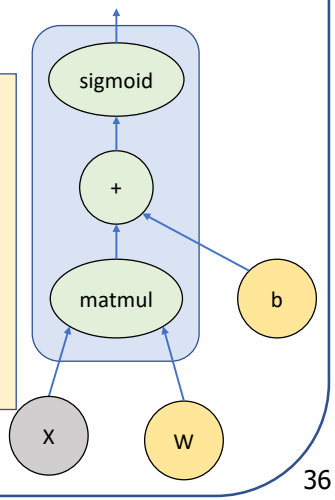
- Đồ thị tính toán của Perceptron

```
@tf.function
def predict(X, W, b):
    return tf.nn.sigmoid(tf.matmul(X, W) + b)

X = tf.constant([[0.0, 0], [0, 1],
                 [1, 0], [1, 1]])

W = tf.Variable([[1.0], [2]])
b = tf.Variable(0.0)

y_hat = predict(X, W, b)
print(y)
```






# Cài đặt Perceptron bằng TF

- Đồ thị tính toán của Perceptron

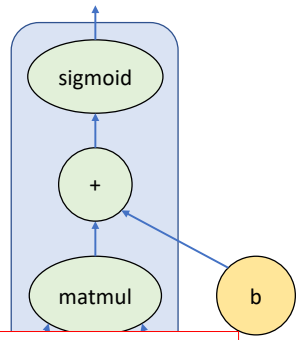
```
@tf.function
def predict(X, W, b):
    return tf.nn.sigmoid(tf.matmul(X, W) + b)

X = tf.constant([[0.0, 0], [0, 1],
                 [1, 0], [1, 1]])

W = tf.Variable([[1.0], [2]])
b = tf.Variable(0.0)

y_hat = predict(X, W, b)
print(y)
```

```
tf.Tensor(
[[0.5       ]
 [0.88079715]
 [0.7310586 ]
 [0.95257413]], shape=(4, 1), dtype=float32)
```

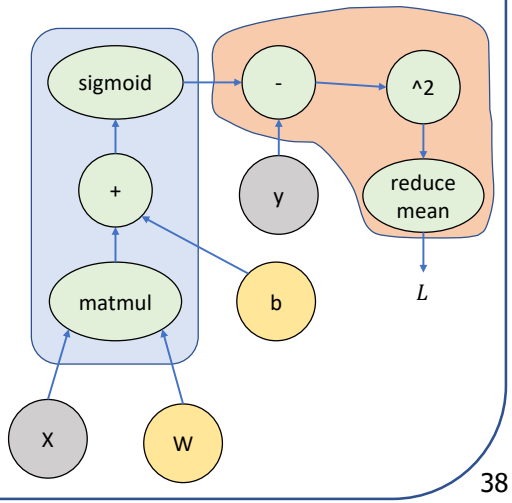





# Cài đặt Perceptron bằng TF

- Kết hợp với hàm lỗi

$$L = \frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)} - \hat{y}^{(i)}\right)^2$$



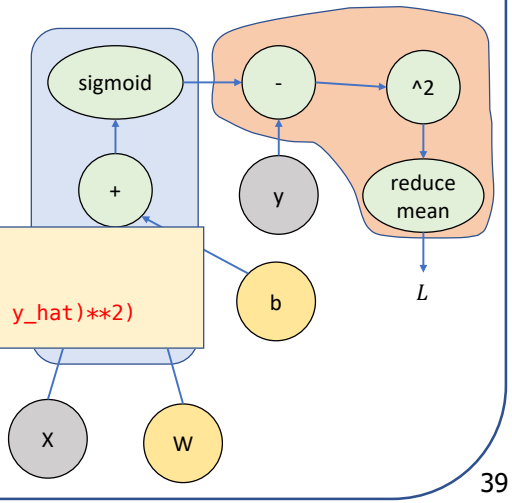



# Cài đặt Perceptron bằng TF

- Kết hợp với hàm lỗi

$$L = \frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)} - \hat{y}^{(i)}\right)^2$$

```
@tf.function
def L(y, y_hat):
    return tf.reduce_mean((y - y_hat)**2)
```



# Cài đặt Perceptron bằng TF

- Huấn luyện mô hình ⇔ Tối ưu hàm lỗi bằng gradient descent
  - Khởi tạo W, b
  - Lặp
    - Tính y_hat
    - Tính giá trị hàm lỗi
    - Tính đạo hàm riêng
    - Cập nhật W và b

$$W = W - \alpha \frac{\partial L}{\partial W}$$

$$b = b - \alpha \frac{\partial L}{\partial b}$$



# Cài đặt Perceptron bằng TF

- Tính đạo hàm riêng tự động với GradientTape

```
X = tf.constant([[0.0, 0], [0, 1], [1, 0], [1, 1]])
y = tf.constant([[0.0], [0], [0], [1]])
W = tf.Variable([[1.0], [2]])
b = tf.Variable(0.0)

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)
```



# Cài đặt Perceptron bằng TF

- Tính đạo hàm riêng tự động với GradientTape

```
X = tf.constant([[0.0, 0], [0, 1], [1, 0], [1, 1]])
y = tf.constant([[0.0], [0], [0], [1]])
W = tf.Variable([[1.0], [2]])
b = tf.Variable(0.0)

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(
    W.assign_sub(alpha *
    b.assign_sub(alpha *
```

```
tf.Tensor(
[[0.07605696]
 [0.30126226]
 [0.25633788]
 [0.64354485]], shape=(4, 1), dtype=float32)
```



# Cài đặt Perceptron bằng TF

- Tính đạo hàm riêng tự động với GradientTape

```
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)
```

```
tf.Tensor(
[[0.07605696]
 [0.30126226]
 [0.25633788]
 [0.64354485]], shape=(4, 1), dtype=float32)
```

# 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 nhãn dự báo (so sánh với 0.5)

```
X = tf.constant([[0.0, 0],
                 [0, 1],
                 [1, 0],
                 [1, 1]])

y = tf.constant([[0.0],
                 [0],
                 [0],
                 [1]])
```



# Thực hành 2

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

$$L = -\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\log\left(\hat{y}^{(i)}\right) + \left(1-y^{(i)}\right)\log\left(\hat{y}^{(i)}\right)\right)$$



# 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 hậu xử lý.



# THANK YOU