DEEP LEARNING
TENSORFLOW & KERAS

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

Tensorflow

- Tensor
  - Mảng nhiều chiều (rất giống ndarray của numpy)
  - Tên lớp: tf.Tensor
  - Kiểu dữ liệu (data type)
  - Hình dạng (shape)
- Các phép toán trên Tensor
  - Cộng, trừ, nhân, sum, mean, log, …

Tensorflow

- Ví dụ:
  - print(tf.add(1, 2))
  - print(tf.add([1, 2], [3, 4]))
  - print(tf.square(5))
  - print(tf.reduce_sum([1, 2, 3]))
    print(tf.square(2) + tf.square(3))

- Kết quả:
  - tf.Tensor(3, shape=(), dtype=int32)
  - tf.Tensor([4 6], shape=(2,), dtype=int32)
  - tf.Tensor(25, shape=(), dtype=int32)

TensorFlow

- Biến (tf.Variable)
  - là Tensor dùng để lưu các giá trị, đại diện cho các biến trong các biểu thức.
  - Giá trị của biến có thể thay đổi thay thời gian
  - Có thể tính đạo hàm theo các biến này
- Gán giá trị cho biến:
  - Hàm assgin() dùng để gán giá trị cho biến
  - v = tf.Variable(1.0)
  - v.assign(3.0) # v chứa 3.0
  - v.assign(tf.square(v)) # v sẽ chứa 9.0

TensorFlow

- Biến (tf.Variable)
  - là Tensor dùng để lưu các giá trị, đại diện cho các biến trong các biểu thức.
  - Giá trị của biến có thể thay đổi thay thời gian
  - Có thể tính đạo hàm theo các biến này
- Gán giá trị cho biến:
  - Hàm assgin() dùng để gán giá trị cho biến
  - v = tf.Variable(1.0)
  - v.assign(3.0) # v chứa 3.0
  - v.assign(tf.square(v)) # v sẽ chứa 9.0

TensorFlow

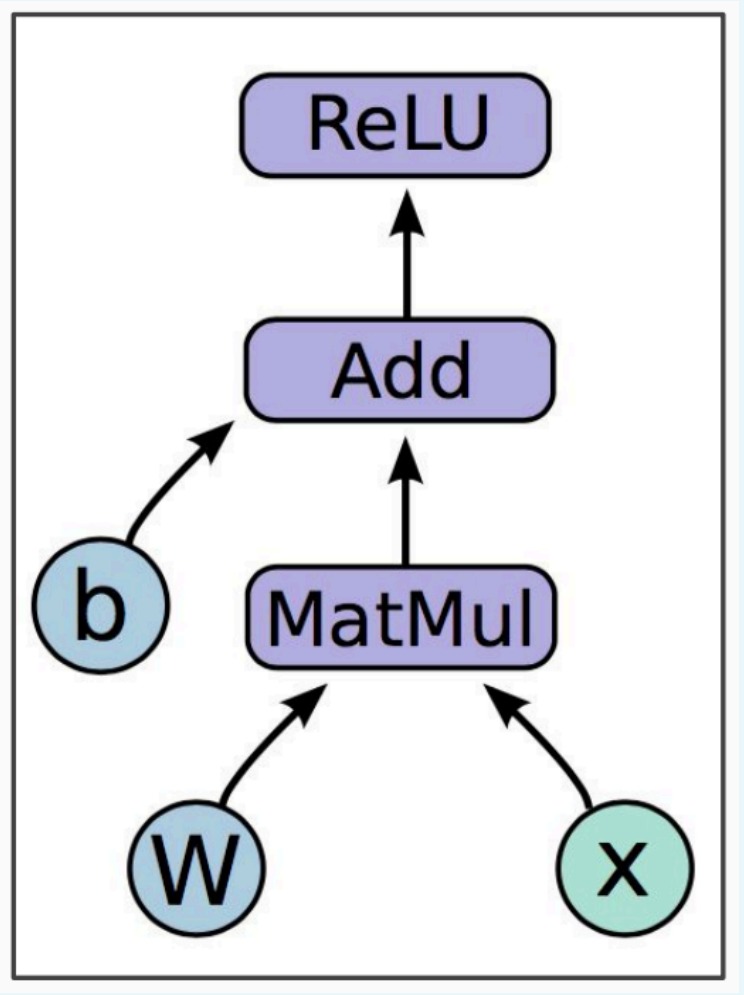
- Đồ thị tính toán:
  - Sử dụng các phép toán trên biến và hằng để tạo ra các biểu thức, hay còn gọi là đồ thị luồng dữ liệu (data flow graph) hay đồ thị tính toán (computation graph).
  - Đỉnh: các phép toán, Tensor
  - Cung: tensor -> phép toán, phép toán -> phép toán

TensorFlow

- Ví dụ: xét hàm h

$$h(x) = \text{ReLU}(Wx + b)$$

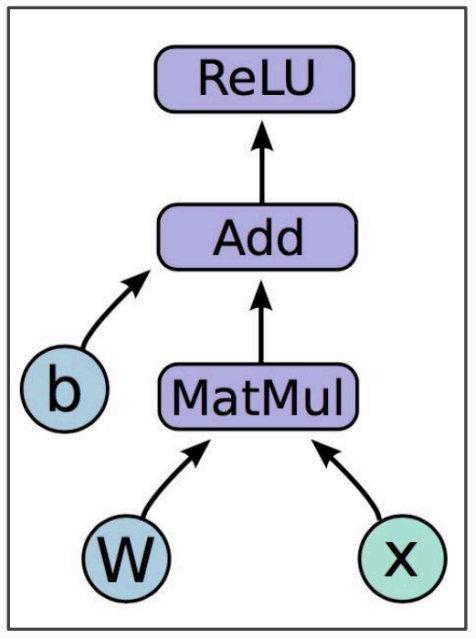
  - x là đối số
  - W, b là tham số



TensorFlow

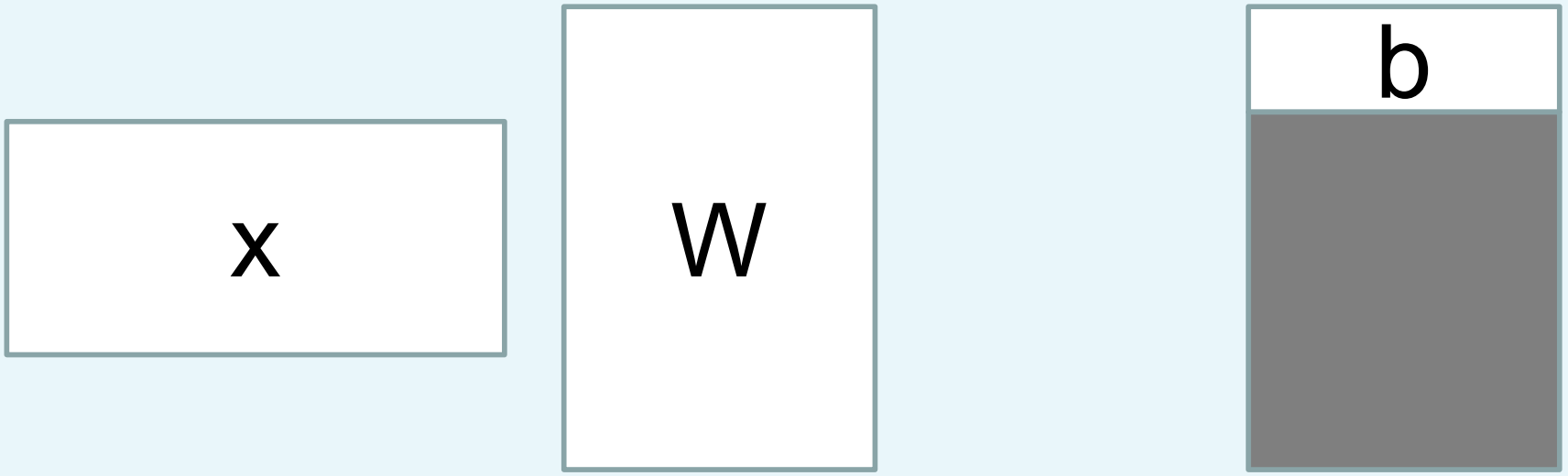
- Trong cài đặt (lập trình) ta sẽ xem:
  - W, b là các biến
  - x: đầu vào của hàm

```
W = tf.Variable(tf.random.uniform((10, 5), -1, 1))
b = tf.Variable(tf.zeros((5, )))


def h(x):
    return tf.nn.relu(tf.matmul(x, W) + b)


p = h(tf.random.uniform((4, 10), -1, 1))


print(p)
```

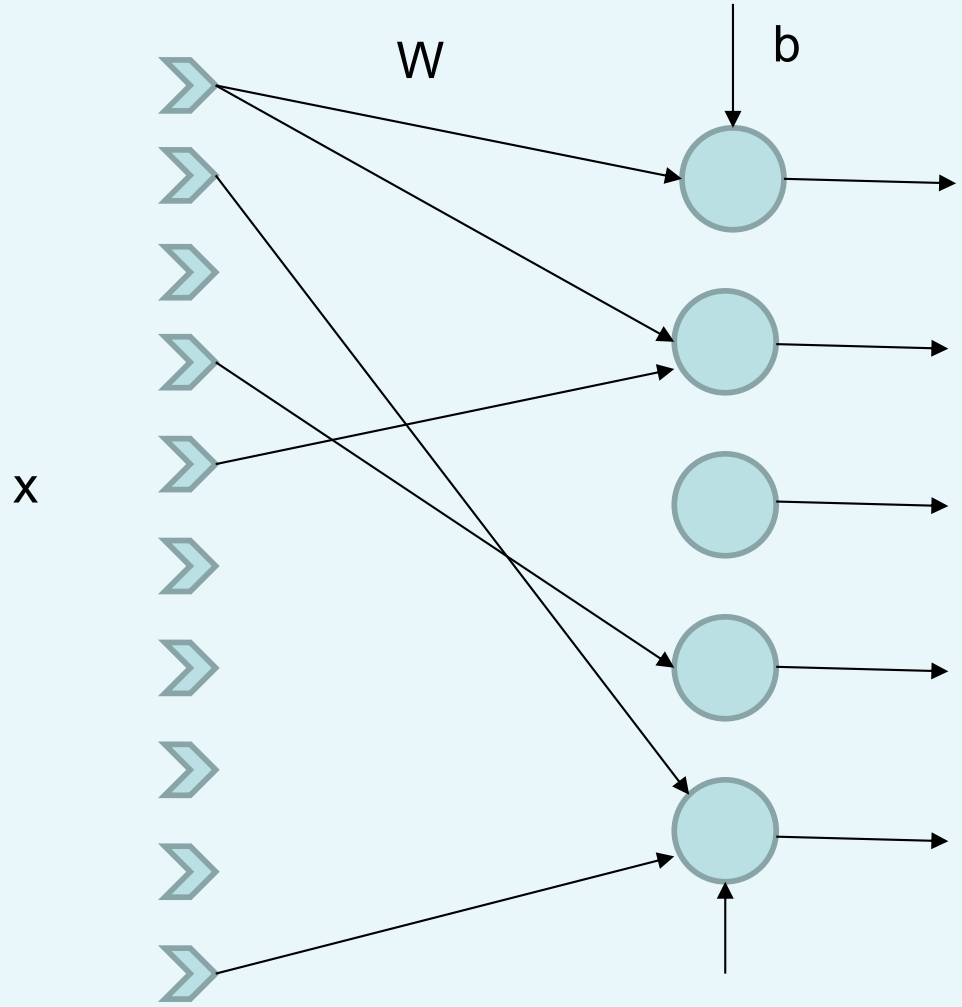


TensorFlow

- x: 4 phần tử, 10 chiều
- W: 10 x 5, tương ứng 5 nơ-ron
- b: 10 phần tử, khi cộng xW với b, b sẽ được broadcast (copy thành 4 cái b giống nhau)

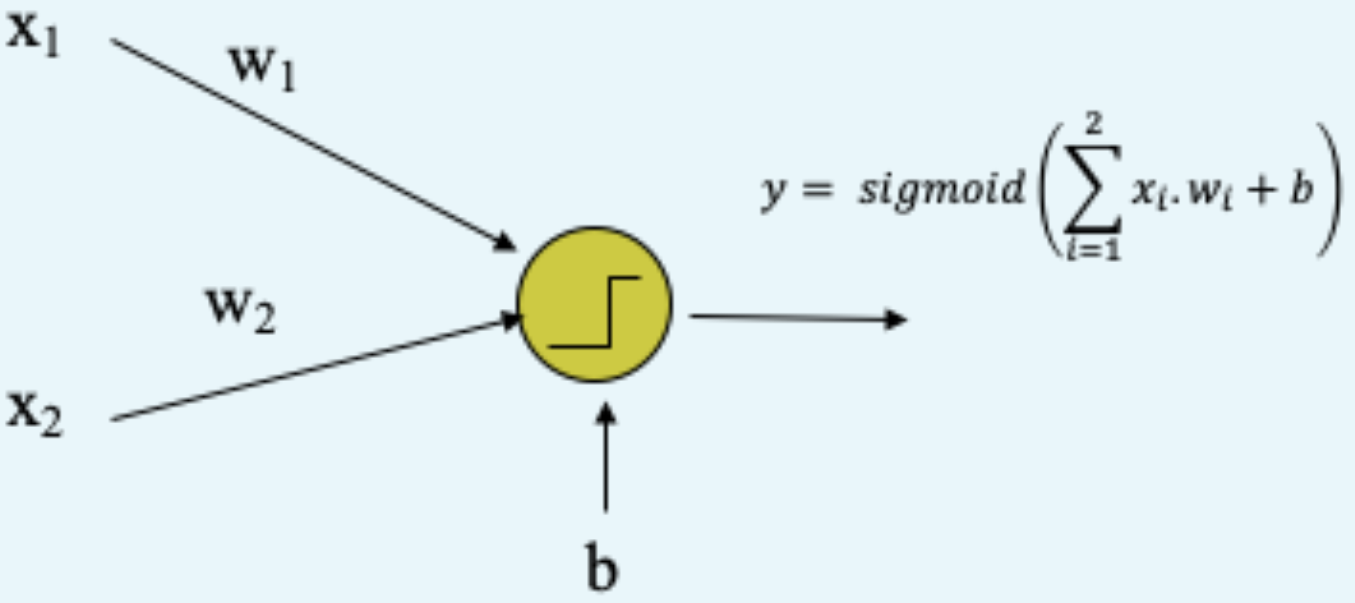


TensorFlow

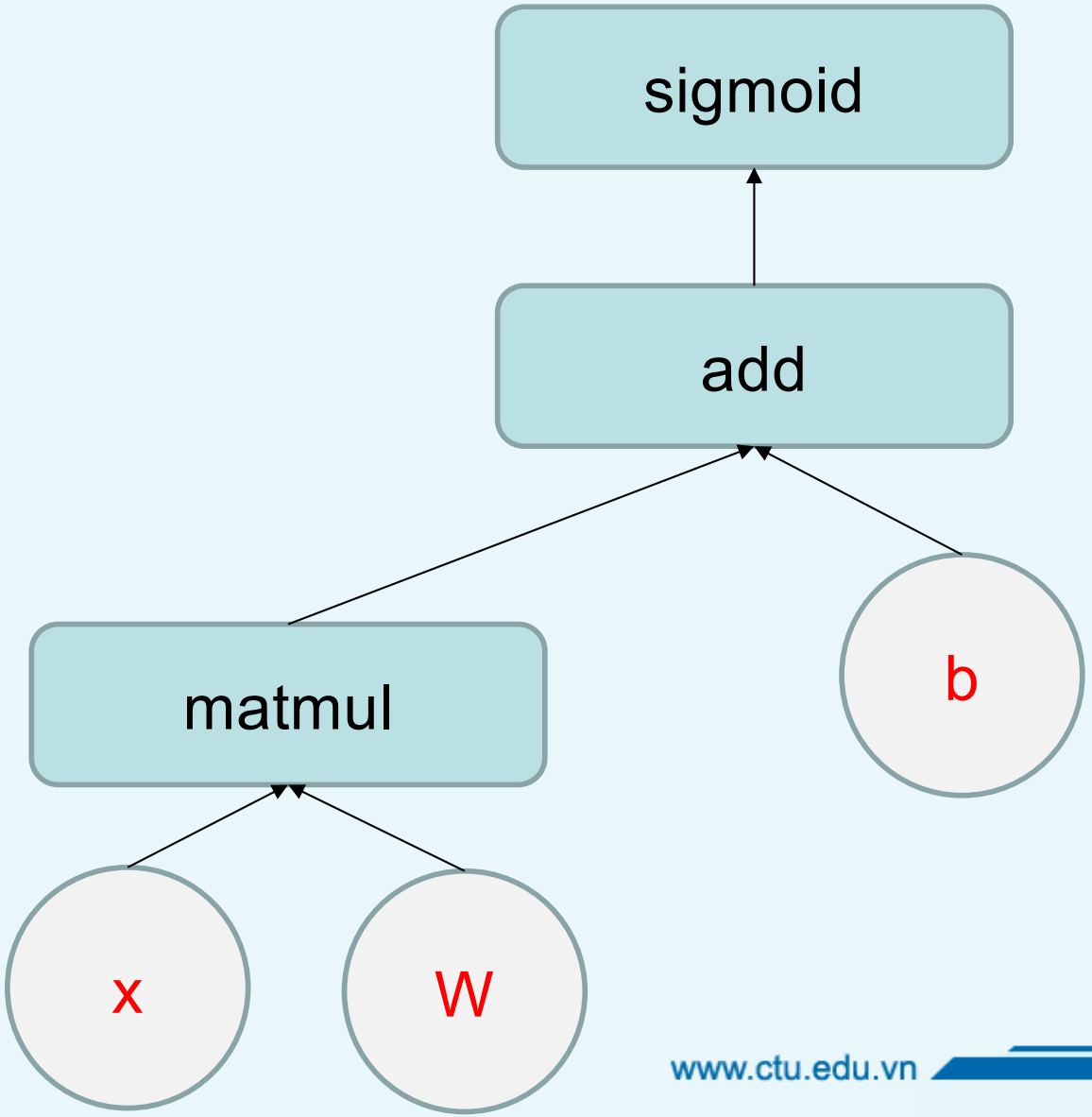


TensorFlow

- Huấn luyện mô hình
  - Xét perceptron như sau:



TensorFlow



TensorFlow

```
W = tf.Variable([[1.0], [2.0]])
b = tf.Variable(4.0)

def predict(x):
    return tf.sigmoid(tf.matmul(x, W) + b)
```

Hàm lỗi/mất mát

```
def loss(target_y, predicted_y):
    return tf.reduce_mean(tf.square(target_y - predicted_y))
```

TensorFlow

- Huấn luyện
  - Lặp:
    - Tính đạo hàm (thực sự là gradient) hàm loss theo W và b
    - Cập nhật:
      - W = W – tốc độ học x đạo hàm theo W
      - b = b – tốc độ học x đạo hàm theo b

TensorFlow

```
def train(inputs, outputs, learning_rate):
    with tf.GradientTape() as t:
        current_loss = loss(outputs, predict(inputs))

    dW, db = t.gradient(current_loss, [W, b])

    W.assign_sub(learning_rate * dW)
    b.assign_sub(learning_rate * db)
```

TensorFlow

- inputs = [[0.0, 0.0],
  [0.0, 1.0],
  [1.0, 0.0],
  [1.0, 1.0]]

  outputs = [[0],
  [0],
  [0],
  [1]]

TensorFlow

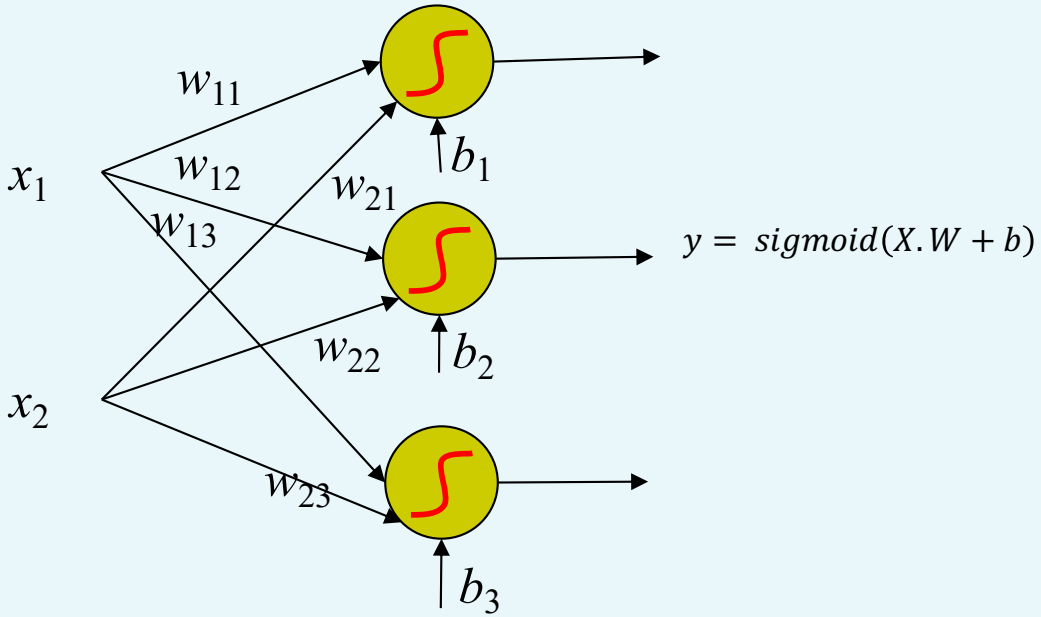
- **for** epoch **in** range(10000):
    current_loss = loss(outputs, predict(inputs))
    print('**Epoch %2d: loss=%2.5f**' % (epoch, current_loss))
    train(inputs, outputs, 0.8)

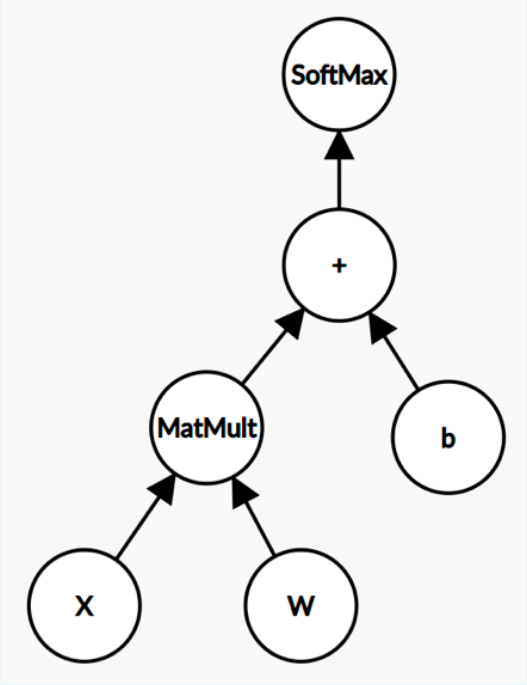
print(W)
print(b)

print(predict(inputs))

TensorFlow

- Mạng nơron cho bài toán nhiều lớp





TensorFlow

- W = tf.Variable(tf.random.uniform((2, 3), -1, -1)) *#w: 2 hàng, 3 cột (3 nơ ron)*

  b = tf.Variable(tf.zeros((3,)), tf.float32) *#b: 3 nơ ron*

```
def predict(x):
    return tf.nn.softmax(tf.matmul(x, W) + b)
```

TensorFlow

- inputs = [[0.0, 0],
  [1, 0],
  [1, 0],
  [1, 1]]

- One hot coding:
  outputs = [[1.0, 0, 0],
  [0, 1, 0],
  [0, 1, 0],
  [0, 0, 1]
  ]

TensorFlow

- **Hàm cross entropy so sánh 2 phân phối xác suất**

```
def loss(y, predicted_y):
    return tf.reduce_mean(-
tf.reduce_sum(y*tf.math.log(predicted_y), 1))
```

-
```
def train(inputs, outputs, learning_rate):
    with tf.GradientTape() as t:
        current_loss = loss(outputs, predict(inputs))
    dW, db = t.gradient(current_loss, [W, b])
    W.assign_sub(learning_rate * dW)
    b.assign_sub(learning_rate * db)
```

TensorFlow

```
for epoch in range(1000):
    current_loss = loss(outputs, predict(inputs))

    print('Epoch %2d: loss=%2.5f' % (epoch,
current_loss))

    train(inputs, outputs, 0.1)

print(W)
print(b)


print(predict(inputs))
```

TensorFlow

```
[[0.9241695  0.05475314 0.02107737]
 [0.01878049 0.9596365  0.02158291]
 [0.01878049 0.9596365  0.02158291]
 [0.00710641 0.03922937 0.95366424]]
```

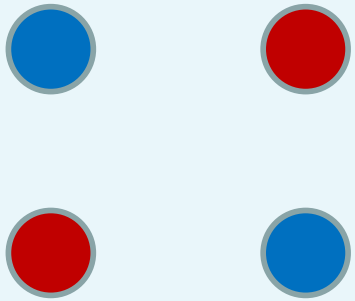
TensorFlow

- Bài tập áp dụng
  - Xây dựng mạng nơ rơn 1 tầng với nhiều ngõ ra để phân lớp tập dữ liệu IRIS

TensorFlow

- Mạng nơ-ron đa tầng
  - Bài toán XOR

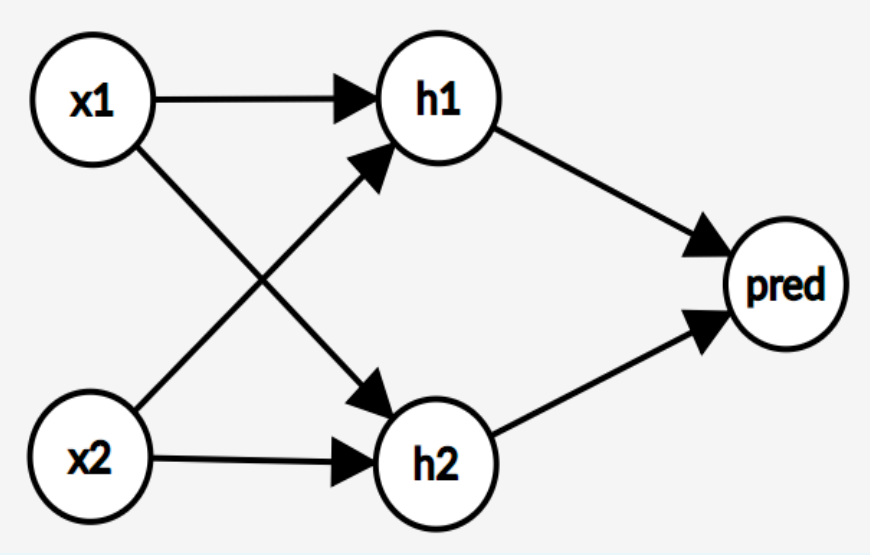
0	0	0
0	1	1
1	0	1
1	1	0



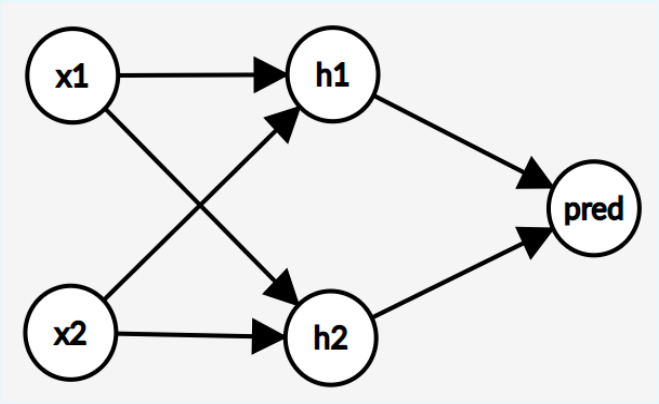
TensorFlow

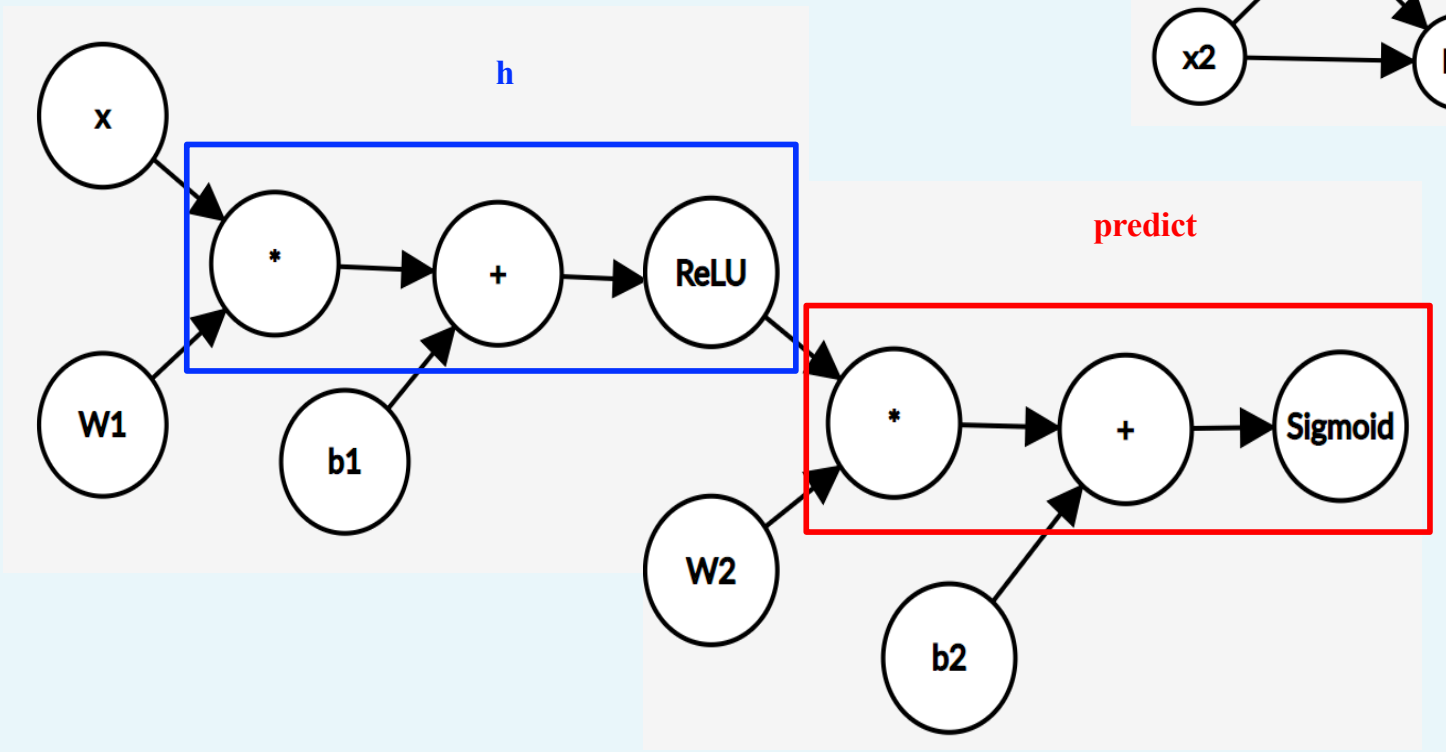
- Mạng nơ-ron đa tầng
  - Bài toán XOR

0	0	0
0	1	1
1	0	1
1	1	0



TensorFlow





CANTHO UNIVERSITY

TensorFlow

CẢM ƠN