DEEP LEARNING

Perceptron đa tầng (MLP) (tiếp theo)

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

CAN THO, 22/12/2022

Nội dung

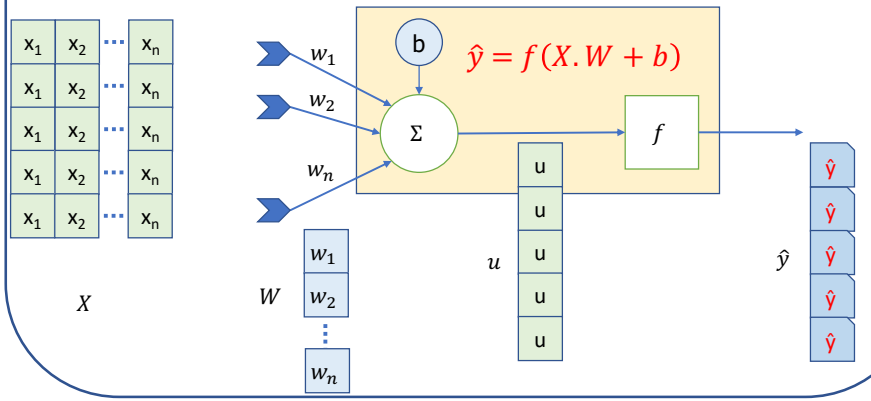
- Cài đặt Perceptron bằng Tensorflow (nhắc lại)
- Mở rộng
  - Mạng nơ ron đơn tầng đa lớp (nhiều perceptron)
- Mạng nơ ron đa tầng
  - hai lớp
  - đa lớp

Percetron

- 1 perceptron = mạng perceptron đơn tầng chỉ gồm 1 nơ ron (perceptron) duy nhất
  - Có khả năng phân lớp tuyến tính (đường thẳng, mặt phẳng, siêu phẳng) nhị phân (hai lớp)

Cài đặt Perceptron bằng TF (nhắc lại)

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

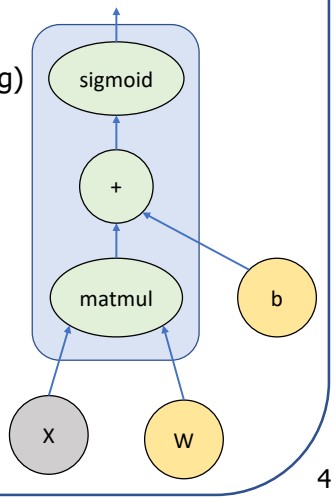


Cài đặt Perceptron bằng TF (nhắc lại)

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

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

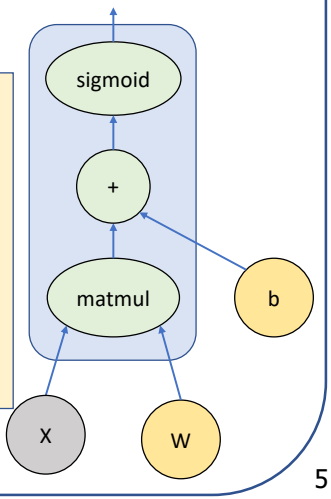
$$f(u) = \frac{e^u}{e^u + 1}$$



Cài đặt Perceptron bằng TF (nhắc lại)

- 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)
```

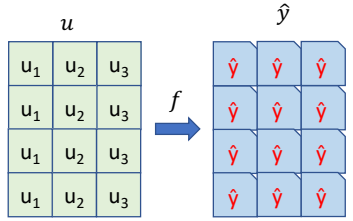


Mạng Percetron đơn tầng (SLP)

- Mạng Single Layer Perceptron (SLP)
  - Mạng đơn tầng gồm nhiều (>=3) nơ ron (perceptron)
  - Có khả năng phân lớp tuyến tính (đường thẳng, mặt phẳng, siêu phẳng) đa phân (từ 3 lớp trở lên)

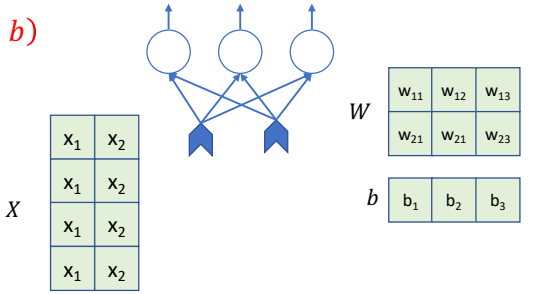
Mạng Perceptron đơn tầng, đa lớp

- Mạng SLP (đa lớp)
  - 1 tầng duy nhất gồm nhiều (>=3) nơ ron



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

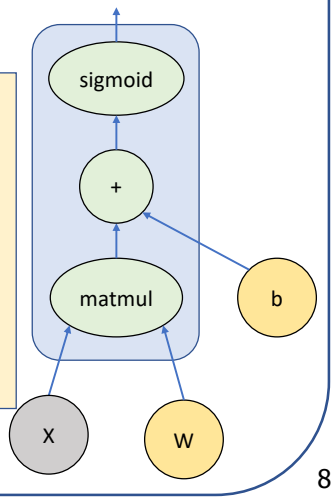
$$u = X.W + b$$



Cài đặt mạng Perceptron đơn tầng, đa lớp

- Đồ thị tính toán SLP (không đổi)

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

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

W = tf.Variable(tf.random.normal((2, 3))
b = tf.Variable([0.0, 0.0, 0.0])

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



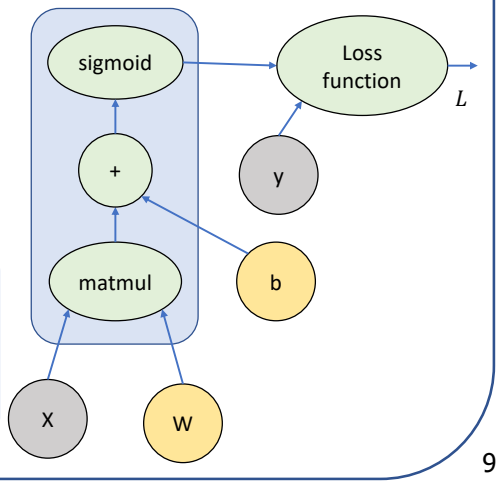
Cài đặt mạng Perceptron đơn tầng, đa lớp

- Kết hợp với hàm lỗi

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

$d\left(y^{(i)}, \hat{y}^{(i)}\right)$: đo sự khác biệt giữa đầu ra mong muốn ($y^{(i)}$) và đầu ra dự báo ($y^{(i)}$) trên mẫu dữ liệu huấn luyện thứ i

Hàm lỗi/mất mát = trung bình sự khác biệt trên toàn bộ tập huấn luyện

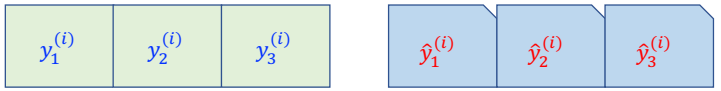


Cài đặt mạng Perceptron đơn tầng, đa lớp

- Đầu ra dự báo của phần tử i: $\hat{y}^{(i)}$
  - 1 vector, có số phần tử = số nơ ron
- Mã hoá đầu ra mong muốn
  - One hot encoding
    - $y^{(i)}$: vector, có số phần tử = số lớp, chỉ có duy nhất 1 phần tử = 1, các phần tử còn lại = 0
    - Ví dụ:
      - Setosa [1, 0, 0]
      - Versicolor [0, 1, 0]
      - Virginica [0, 0, 1]

Cài đặt mạng Perceptron đơn tầng, đa lớp

- Đo sự khác biệt giữa đầu ra mong muốn và đầu ra dự báo của phần tử i
  - $d\left(y^{(i)}, \hat{y}^{(i)}\right)$ = sự khác biệt của 2 vectors $y^{(i)}$ và $\hat{y}^{(i)}$



$$d\left(y^{(i)}, \hat{y}^{(i)}\right) = -\sum_{j=1}^{k} y_j^{(i)}.\log\left(\hat{y}_j{}^{(i)}\right)$$

Categorical Cross-Entropy

Cài đặt mạng Perceptron đơn tầng, đa lớp

- Kết hợp với hàm lỗi

$$L = \frac{1}{m}\sum_{i=1}^{m}\left(-\sum_{j=1}^{k} y_j^{(i)}.\log\left(\hat{y}_j{}^{(i)}\right)\right) = -\frac{1}{m}\sum_{i=1}^{m}\left(\sum_{j=1}^{k} y_j^{(i)}.\log\left(\hat{y}_j{}^{(i)}\right)\right)$$

```
@tf.function
def L(y, y_hat):
    return -tf.reduce_mean(tf.reduce_sum(y*log(y_hat), axis=1))
```


Cài đặt mạng Perceptron đơn tầng, đa lớp

- **Tổng kết cài đặt SLP đa lớp**
  - Mạng chỉ gồm 1 tầng, nhiều (>=3) nơ rơn
  - Xác định số nơ ron của mạng = số lớp của bài toán
  - Định nghĩa kích thước của các tham số (Variables)
    - W: (n, k) n: số chiều dữ liệu, k: số lớp
    - b: (k,)
  - Hàm activation: dùng softmax thay cho sigmoid
  - Định nghĩa hàm lỗi: sử dụng Categorial Cross-Entropy (đa lớp) thay cho Binary Cross-Entropy (hai lớp)
  - Mã hoá đầu ra mong muốn
    - One hot encoding

Thực hành 1

- Tổng hợp các mã lệnh được trình bày trong phần trên để xây dựng một mạng nơ ron perceptron đơn tầng, 3 lớp với dữ liệu huấn luyện như bên dưới

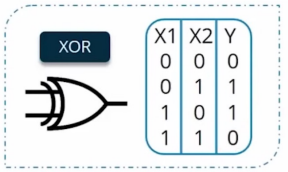
```
X = tf.constant([[0.0, 0],
                 [0, 1],
                 [1, 0],
                 [1, 1]])

y = tf.constant([[1.0, 0, 0],
                 [0, 1, 0],
                 [0, 1, 0],
                 [0, 0, 1]])
```

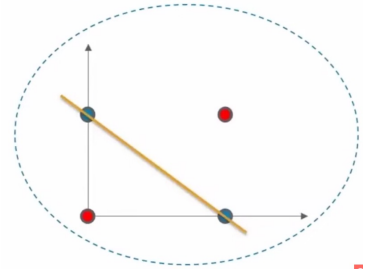

Thực hành 2

- Làm lại bài phân loại hoa iris (3 lớp) với Tensorflow thay vì dùng Keras
  - Cần mã hoá đầu ra mong muốn: có thể xử lý thủ công hoặc dùng công cụ
  - Sử dụng hàm lỗi Categorical Cross-Entropy
  - Cần xử lý đầu ra để tìm nhãn chính xác (sử dụng hàm argmax để tìm cột có giá trị lớn nhất)
  - 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.

Nhược điểm của mạng SLP

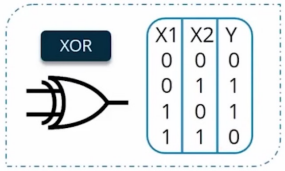
- Mô phỏng cổng XOR
  - 1 nơ ron

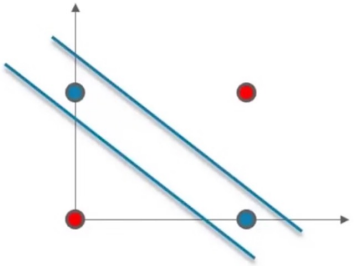




Nhược điểm của mạng SLP

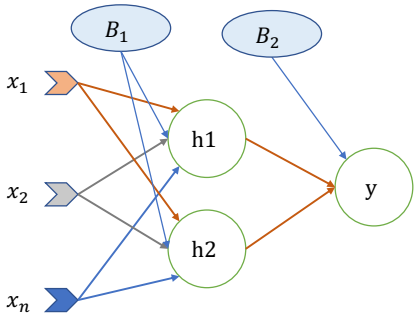
- Mô phỏng cổng XOR
  - 2 nơ ron và (1 nơ ron output)





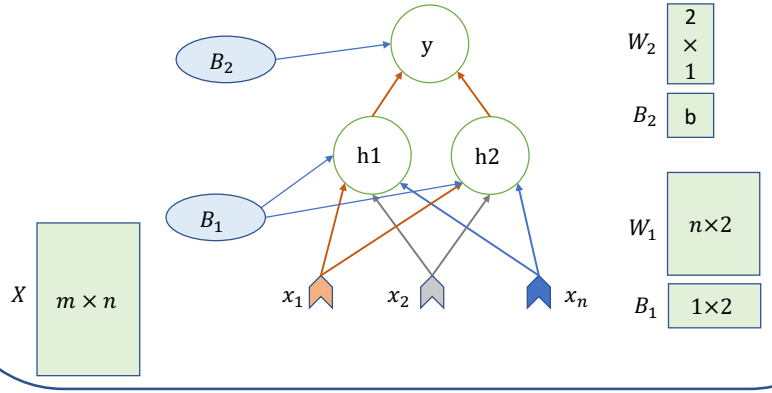
Mạng nơ ron đa tầng MLP

- Mô phỏng cổng XOR
  - 2 nơ ron và (1 nơ ron output)



Mạng nơ ron đa tầng MLP

- Mô phỏng cổng XOR
  - 2 nơ ron và (1 nơ ron output)

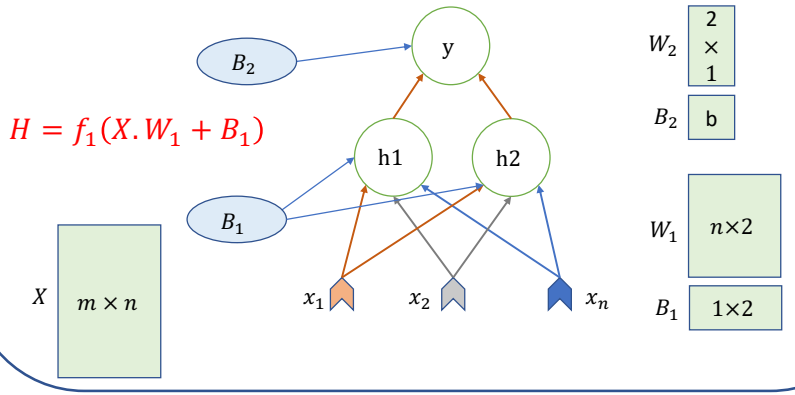


Mạng nơ ron đa tầng MLP

- Mô phỏng cổng XOR
  - 2 nơ ron và (1 nơ ron output)

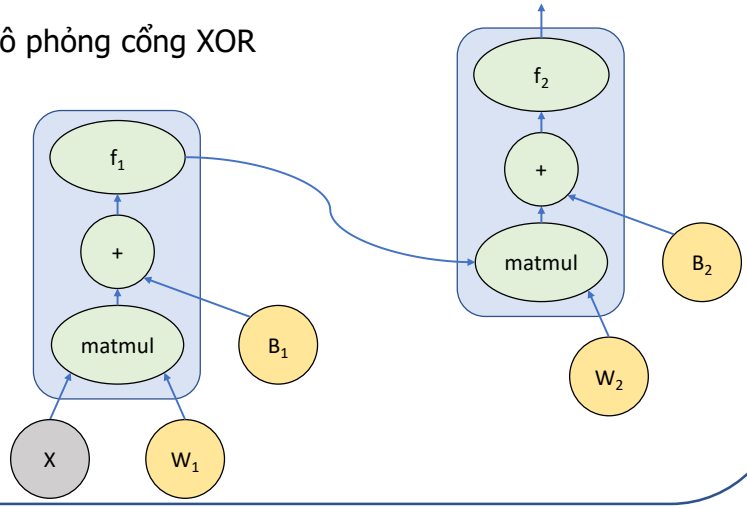
$\hat{y} = f_2(H.W_2 + B_2)$

$H = f_1(X.W_1 + B_1)$



Mạng nơ ron đa tầng MLP

- Mô phỏng cổng XOR



Mạng nơ ron đa tầng MLP

- Mô phỏng cổng XOR

```
@tf.function
def layer1(X, W, b):
    return tf.nn.relu(tf.matmul(X, W) + B)

@tf.function
def layer2(X, W, B):
    return tf.nn.sigmoid(tf.matmul(X, W) + B)

@tf.function
def predict(X, W1, B1, W2, B2):
    return layer2(layer1(X, W1, B1), W2, B2)
```


Mạng nơ ron đa tầng MLP

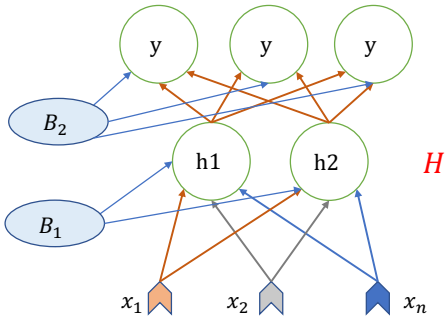
- Mô phỏng cổng XOR

```
n = 2
W1 = tf.Variable(tf.random.normal((n, 2))
B1 = tf.Variable([0.0, 0.0])

W2 = tf.Variable(tf.random.normal((2, 1))
B2 = tf.Variable([0.0])


y_hat = predict(X, W1, B1, W2, B2)
print(y_hat)
```

Mạng nơ ron đa tầng MLP

- MLP đa lớp (tương tự MLP 2 lớp)

$$\hat{y} = f_2(H.W_2 + B_2)$$



$$H = f_1(X.W_1 + B_1)$$

Thực hành 3

- Tổng hợp các mã lệnh được trình bày trong phần trên để xây dựng một mạng nơ ron perceptron đa tầng (2 tầng), 2 lớp cho bài toán XOR với dữ liệu huấn luyện như bên dưới

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

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


Thực hành 4

- Xây dựng một mạng nơ ron perceptron 2 tầng, 2 lớp. Đọc tập dữ liệu trong file data.csv, chia dữ liệu thành 2 phần 80% - 20%, dùng 80% huấn và 20% còn lại để đánh giá

Thực hành 5

- Xây dựng một mạng nơ ron MLP xử lý tập dữ liệu: https://archive.ics.uci.edu/ml/machine-learning-databases/character-trajectories/
  - Chia dữ liệu thành 2 phần 80% - 20%, dùng 80% huấn và 20% còn lại để đánh giá

THANK YOU