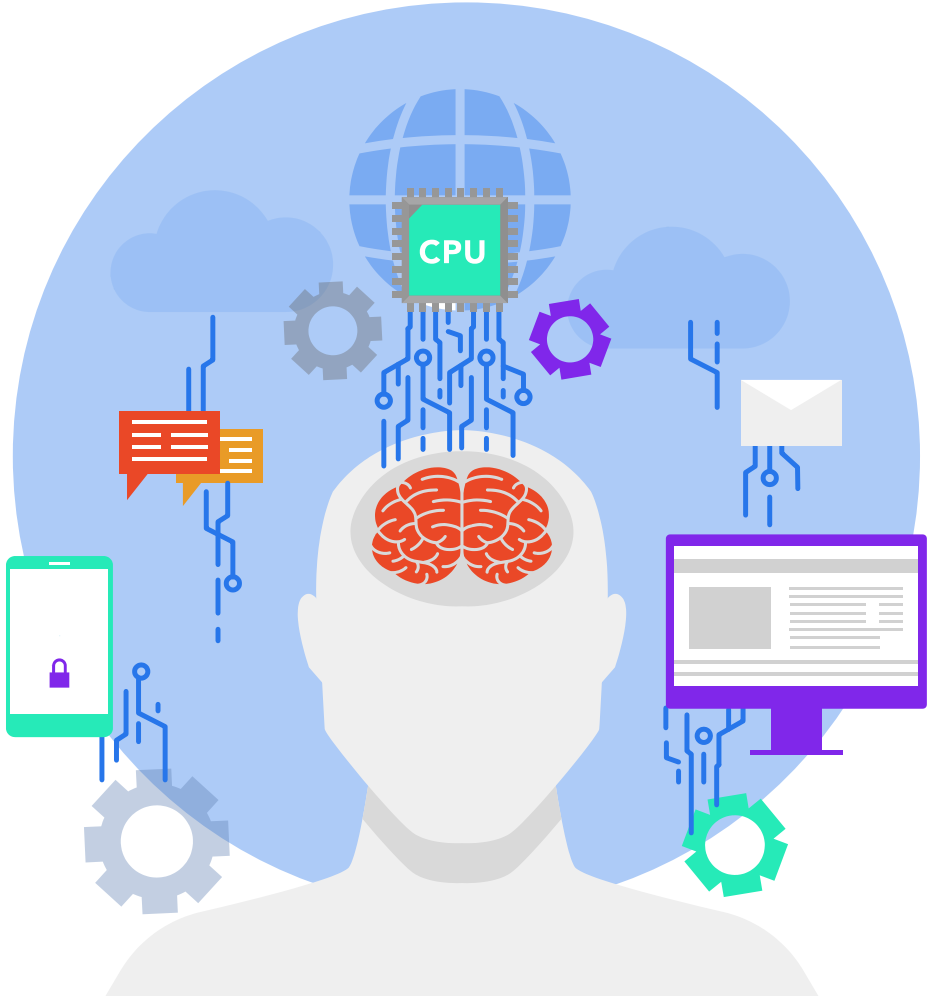
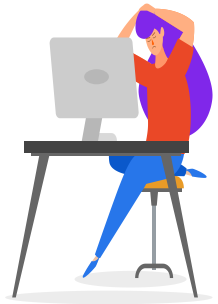


Chương 5: Phương pháp mạng nơ-ron

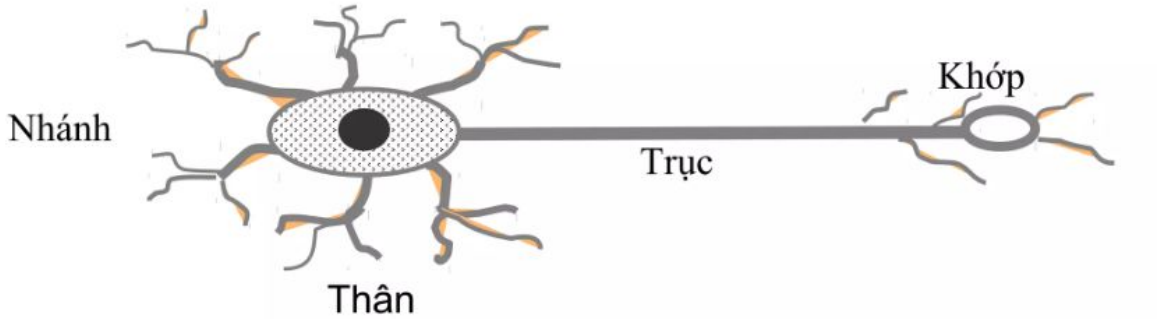
5.1 Giới thiệu



Giới thiệu

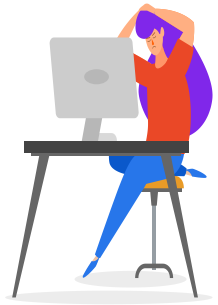
Mạng Neural là thuật ngữ nói đến một phương pháp giải quyết vấn đề - bài toán trên máy tính mô phỏng theo hoạt động của các tế bào thần kinh trong não bộ

Mạng Neural nhân tạo là sự mô phỏng cấu trúc của mạng Neural sinh học. Mạng Neural nhân tạo được tạo thành bởi sự nối kết giữa rất nhiều đơn vị thần kinh gọi là perceptron



Cấu trúc của một tế bào thần kinh sinh học

5.1 Giới thiệu



Cấu tạo một đơn vị thần kinh nhân tạo

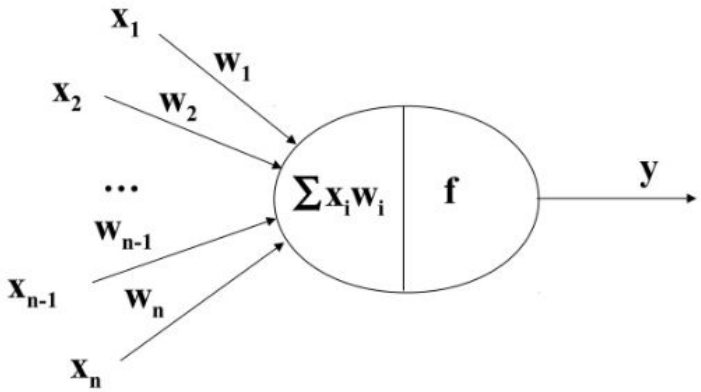
Giá trị đầu ra y của một perceptron được tính bằng công thức sau:

$$y = f((x_n W_n + x_{n-1} W_{n-1} + \cdots + W_2 n_2 + W_1 n_1 + W_0) - \Phi)$$

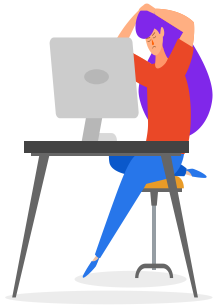
Φ được gọi là ngưỡng kích hoạt của Neural

Hàm f được gọi là hàm truyền. Một hàm truyền cần phải có tính chất sau:

- Bị chặn
- Đơn điệu tăng
- Hàm liên tục tăng



5.1 Giới thiệu



Các hàm truyền thường được sử dụng"

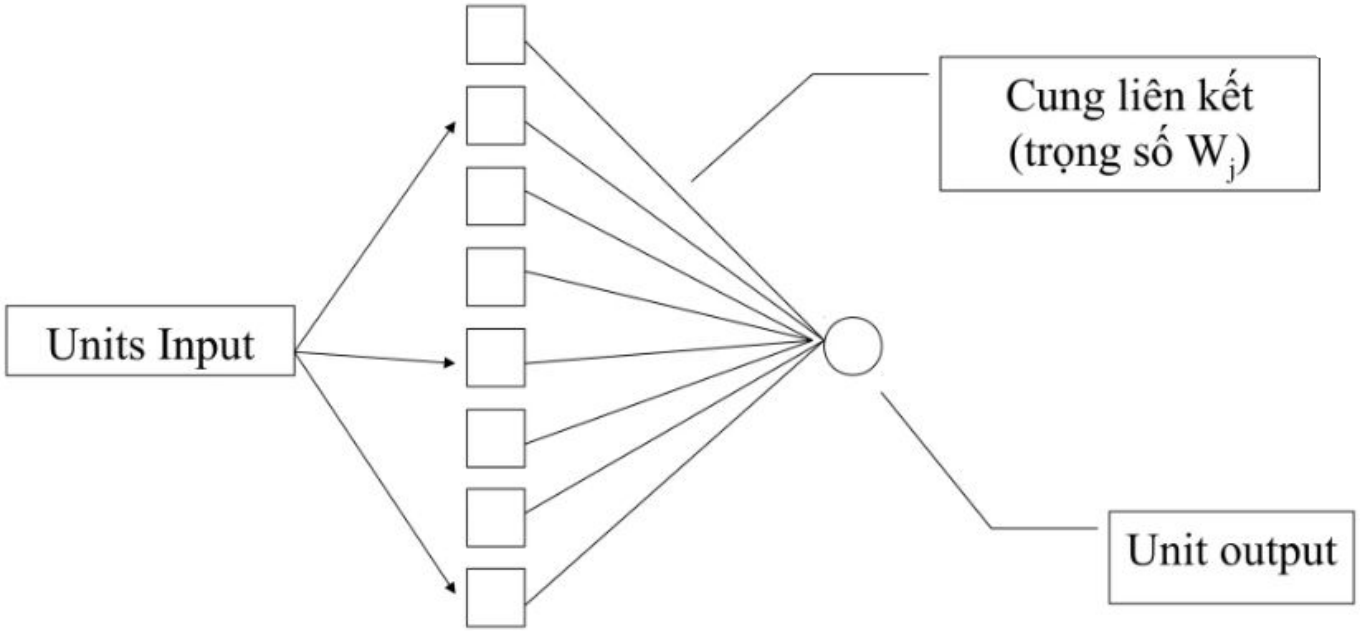
Hàm logistic (hay còn gọi là hàm Sigma) $$f(x)=\frac{1}{1+e^{-x}}$$

Hàm hyperbol $$h(x)=\frac{1-e^{-x}}{1+e^{-x}}$$

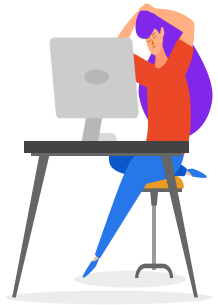
Hàm tăng-hyperbol $$tanh\,(x)=\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$$

5.1 Giới thiệu

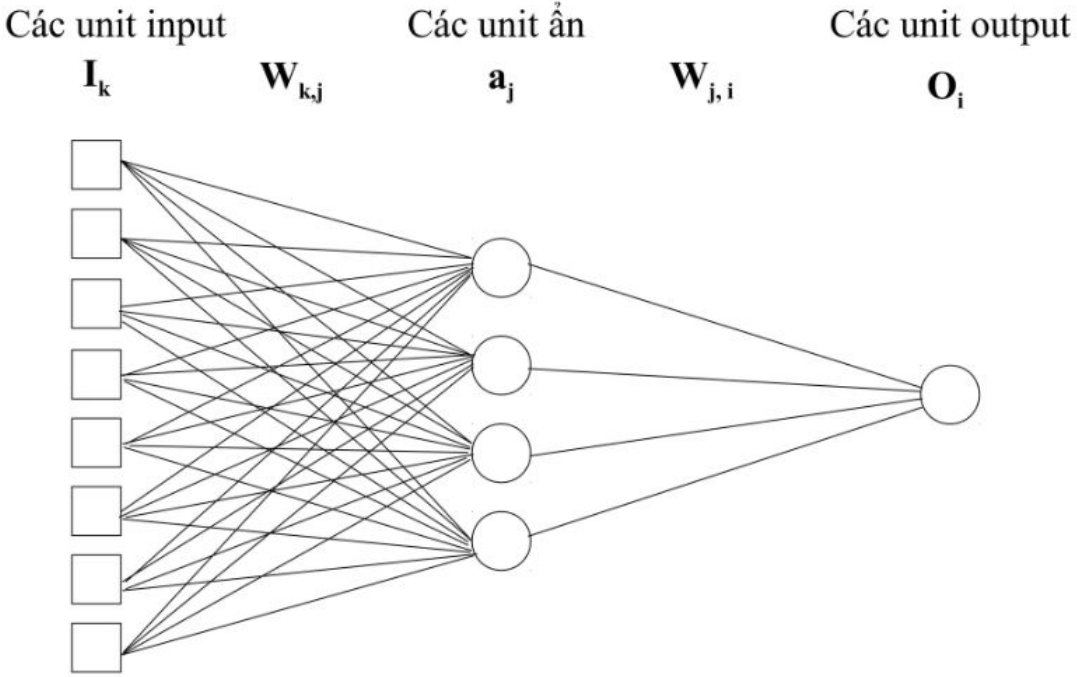
Mô hình minh hoạ mạng Neural 1 lớp



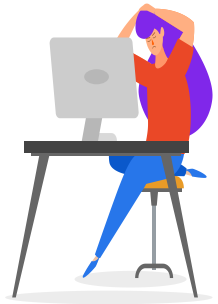
5.1 Giới thiệu



Mô hình minh hoạ mạng Neural tổng quát

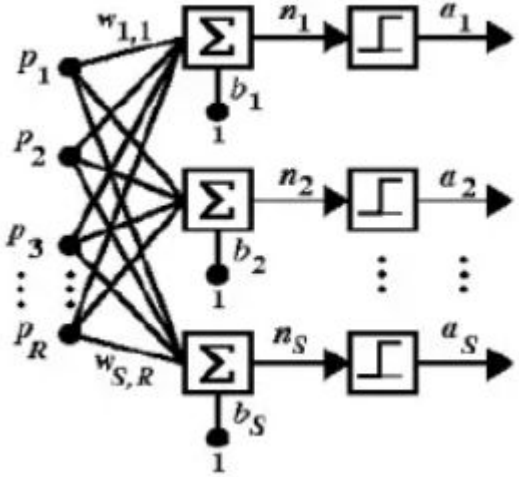


5.2 Mạng Perceptron



Mạng Perceptron

- Cuối những năm 1950, Frank Rosenblatt và các cộng sự phát triển một lớp mạng nơ -ron được gọi là Perceptron.
- Perceptron là mạng một lớp và được ứng dụng cho bài toán phân lớp nhị phân

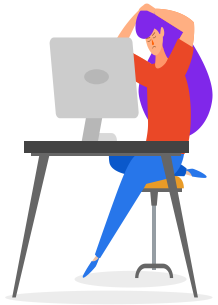


- $p = (p_1, p_2, \dots, p_R)$ là vector tín hiệu vào
- Ma trận trọng số W với kích thước S x R:

$$W = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,R} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,R} \\ \cdots & \cdots & \cdots & \cdots \\ w_{S,1} & w_{S,2} & \cdots & w_{S,R} \end{bmatrix}$$

- $b = (p_1, p_2, \dots, p_R)^T$
- $n_i = W_i^T p + b_i$

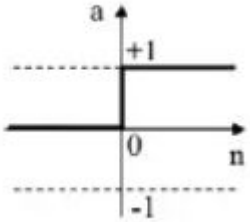
5.2 Mạng Perceptron

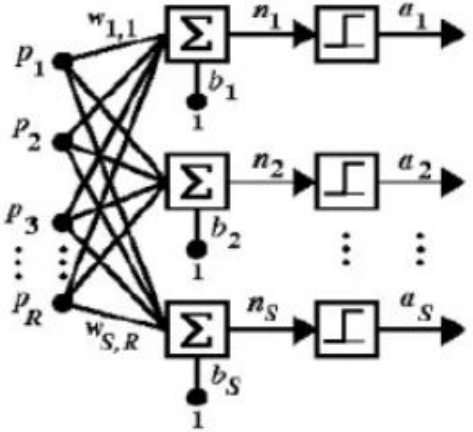


Mạng Perceptron

- Kiến trúc mạng:

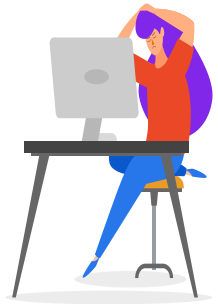
Hàm kích hoạt





Đầu ra thứ i được tính theo $a_i = hardlim(n_i)$, $hardlim(n) = 1$ nếu n ≥ 0 và bằng 0 trong các trường hợp còn lại

5.2 Mạng Perceptron



Thuật giải huấn luyện mạng Perceptron

- Input: $\{p_1, t_1\}, \{p_2, t_2\}, \ldots, \{p_Q, t_Q\}$
- Output: W, b

Bước 1: Khởi tạo ngẫu nhiên W, b.

Bước 2: Với mỗi p_i tính đầu ra thực tế:

$$a = \text{hardlim}(Wp_i + b)$$

Bước 3: So sánh a với t_i

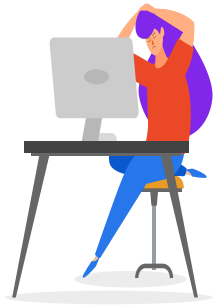
Nếu $a = t_i$ thì $W^{new} = W^{old}$

Nếu $a \neq t_i$ thì $W^{new} = W^{old} + (t_i - a)p_i^T$

$$b^{new} = b^{old} + (t_i - a)$$

Bước 4: Lặp bước 2, bước 3 cho đến khi $a = t_i$, $\forall i = 1, 2, \ldots, Q$.

5.2 Mạng Perceptron

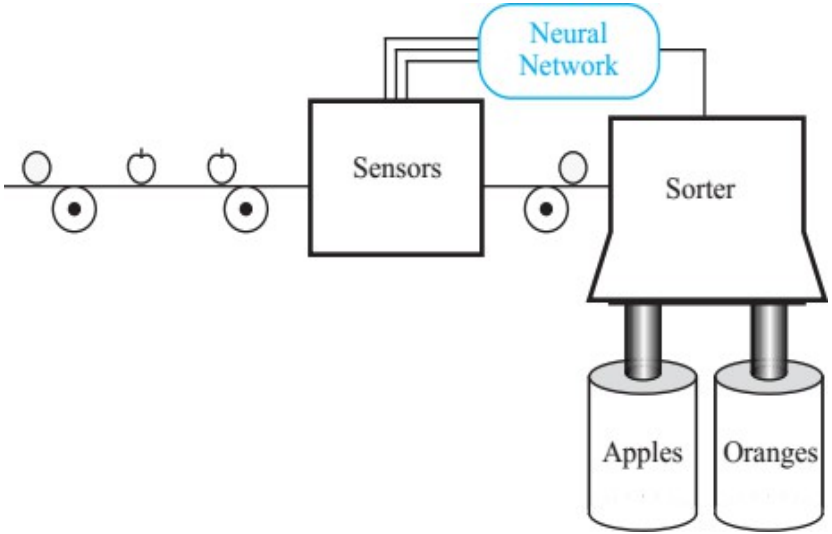


Thuật giải huấn luyện mạng Perceptron – Ví dụ minh hoạ

Giả sử có một kho chứa hoa quả: Cam và táo để lẫn

lộn. Thiết kế mô hình cho phép phân 2 loại quả này.

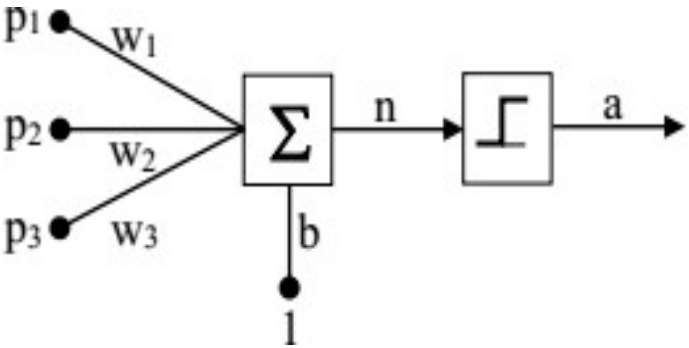
- Mỗi loại quả x(Hình dạng, vỏ, cân nặng)

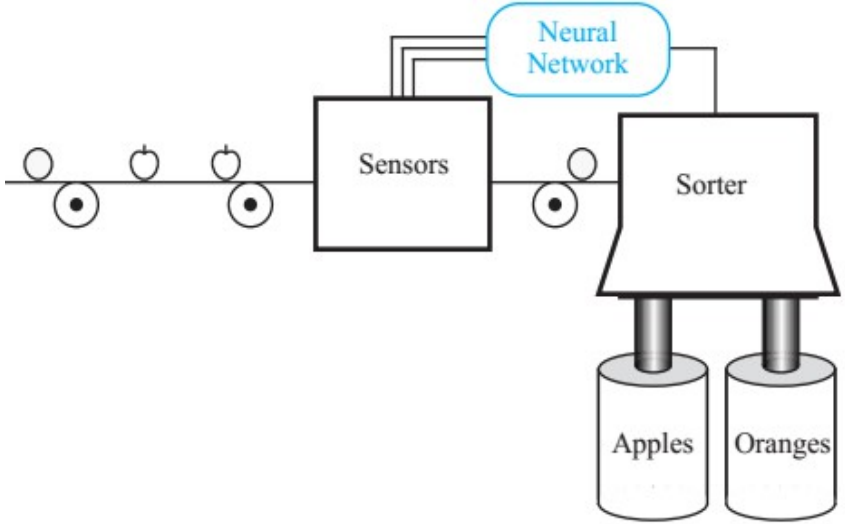


5.2 Mạng Perceptron

Thuật giải huấn luyện mạng Perceptron – Ví dụ minh hoạ

- **Ví dụ minh họa**: Xét cặp tín hiệu vào/ra: $\{p_1 = (1\ -1\ -1)^T, t_1 = 0\}$, $\{p_2 = (1\ 1\ -1)^T, t_2 = 1\}$





5.2 Mạng Perceptron

Thuật giải huấn luyện mạng Perceptron – Ví dụ minh hoạ

- **Ví dụ minh họa**: Xét cặp tín hiệu vào/ra: $\{p_1 = (1\ -1\ -1)^T, t_1 = 0\}$, $\{p_2 = (1\ 1\ -1)^T, t_2 = 1\}$

Khởi tạo bộ trọng số: $W = [0{,}5\ -1\ \ -0{,}5]$, $b = 0{,}5$.

Lần lặp thứ 1:

$$a = \text{hardlim}(Wp_1 + b) = \text{hardlim}\left([0{,}5\ -1\ -0{,}5] \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} + 0{,}5 \right)$$

$$= \text{hardlim}(2{,}5) = 1$$

Do $a \neq t_1$ nên cập nhật ma trận trọng số:

$$W^{new} = W^{old} + (t_1 - a)p_1^T = [0{,}5\ \ -1\ \ -0{,}5] + (-1)(1\ \ -1\ \ -1) = [-0{,}5\ \ 0\ \ 0{,}5]$$

$$b^{new} = b^{old} + (t_1 - a) = 0{,}5 + (-1) = -0{,}5$$

5.2 Mạng Perceptron

Thuật giải huấn luyện mạng Perceptron – Ví dụ minh hoạ

- **Ví dụ minh họa**: Xét cặp tín hiệu vào/ra: $\{p_1 = (1\ -1\ -1)^T, t_1 = 0\}$, $\{p_2 = (1\ 1\ -1)^T, t_2 = 1\}$

Lần lặp thứ 1:

$$a = \text{hardlim}(Wp_2 + b) = \text{hardlim}\left([-0{,}5\ 0\ 0{,}5] \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} - 0{,}5 \right)$$

$$= \text{hardlim}(-0{,}5) = 0$$

Vì $a \neq t_2$ nên cập nhật ma trận trọng số:

$$W^{new} = W^{old} + (t_2 - a)p_2^T = [-0{,}5\ \ 0\ \ 0{,}5] + (1)(1\ \ 1\ \ -1) = [0{,}5\ \ 1\ -0{,}5]$$

$$b^{new} = b^{old} + (t_2 - a) = -0{,}5 + 1 = 0{,}5$$

5.2 Mạng Perceptron

Thuật giải huấn luyện mạng Perceptron – Ví dụ minh hoạ

- **Ví dụ minh họa**: Xét cặp tín hiệu vào/ra: $\{p_1 = (1\ -1\ -1)^T, t_1 = 0\}$, $\{p_2 = (1\ 1\ -1)^T, t_2 = 1\}$

Lần lặp thứ 2:

$$a = \text{hardlim}(Wp_1 + b) = \text{hardlim}\left([0{,}5\ 1\ -0{,}5] \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} + 0{,}5 \right)$$

$$= \text{hardlim}(0{,}5) = 1 \neq t_1$$

$$W^{new} = W^{old} + (t_1 - a)p_1^T = [0{,}5\ \ 1\ \ -0{,}5] + (-1)(1\ \ -1\ \ -1) = [-0{,}5\ \ 2\ \ 0{,}5]$$

$$b^{new} = b^{old} + (t_1 - a) = 0{,}5 + (-1) = -0{,}5$$

$$a = \text{hardlim}(Wp_2 + b) = \text{hardlim}\left([-0{,}5\ 2\ 0{,}5] \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} - 0{,}5 \right) = \text{hardlim}(0{,}5)$$

$$= 1$$

Do $a = t_2$ nên giữ nguyên trọng số.

5.2 Mạng Perceptron

Thuật giải huấn luyện mạng Perceptron – Ví dụ minh hoạ

- **Ví dụ minh họa**: Xét cặp tín hiệu vào/ra: $\{p_1 = (1\ -1\ -1)^T, t_1 = 0\}$, $\{p_2 = (1\ 1\ -1)^T, t_2 = 1\}$

Lần lặp thứ 3:

$$a = \text{hardlim}(Wp_1 + b) = \text{hardlim}\left([-0{,}5\ \ 2\ 0{,}5] \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} - 0{,}5 \right) = 0 = t_1$$

$$a = \text{hardlim}(Wp_2 + b) = \text{hardlim}\left([-0{,}5\ 2\ 0{,}5] \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} - 0{,}5 \right) = 1 = t_2$$

5.2 Mạng Perceptron

Thuật giải huấn luyện mạng Perceptron – Ví dụ minh hoạ

- **Ví dụ minh họa**: Xét cặp tín hiệu vào/ra: $\{p_1 = (1\ -1\ -1)^T, t_1 = 0\}$, $\{p_2 = (1\ 1\ -1)^T, t_2 = 1\}$

Đến đây thuật toán dừng và ta nhận được bộ trọng số: W = [-0,5 2 0,5], b = -0,5.

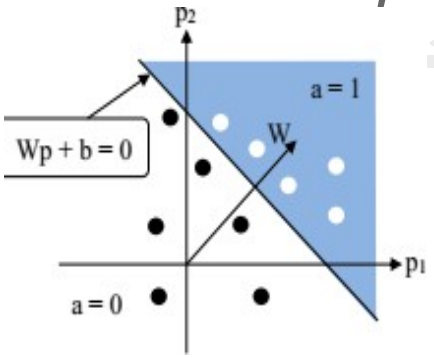
Cần gán nhãn cho mẫu mới $p = (1\ \ -1\ \ 1)^T$. Với mẫu này đầu ra a =

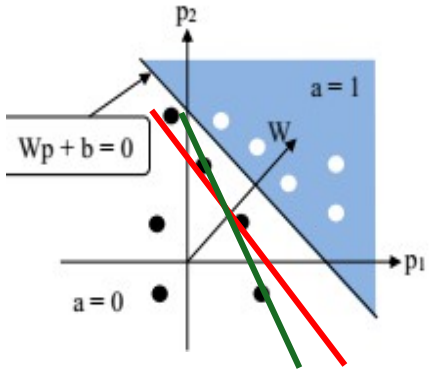
$$\text{hardlim}(Wp + b) = \text{hardlim}\left([-0{,}5\ 2\ 0{,}5] \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} - 0{,}5 \right) = 0.$$

5.2 Mạng Perceptron

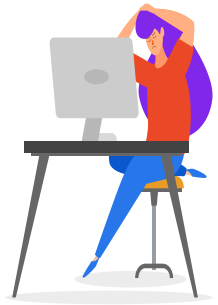
Thuật giải huấn luyện mạng Perceptron – Ví dụ minh hoạ

- **Nhận xét**:
  - Quá trình huấn luyện mạng Perceptron thực chất là tìm siêu phẳng phân tách tập điểm {p_1, p_2,…, p_Q} thành hai vùng, điểm đen và điểm trắng.
  - Có vô số đường phân tách hai vùng này.



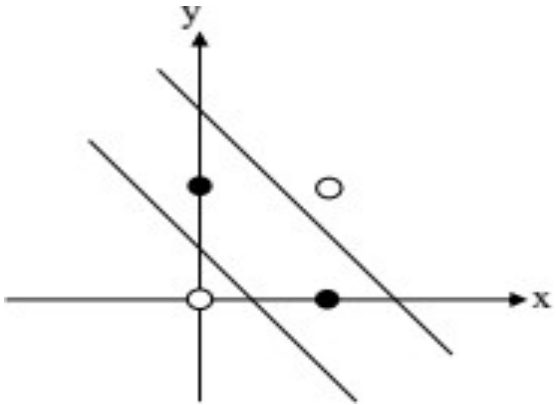


5.2 Mạng Perceptron



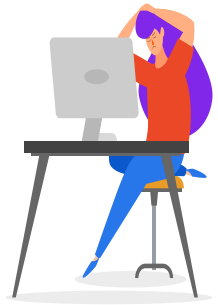
Thuật giải huấn luyện mạng Perceptron – Ví dụ minh hoạ

- **Nhận xét**:
  - Mạng này không giải quyết được bài toán XOR



- Thiết kế mạng nhiều lớp cùng thuật toán lan truyền ngược (backpropagation algorithm) để huấn luyện mạng này.

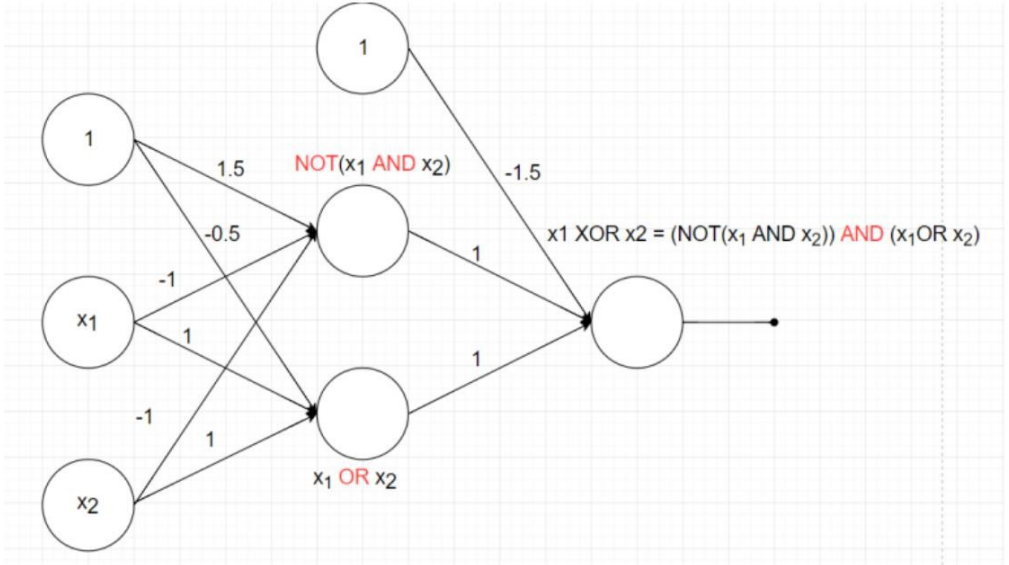
5.2 Mạng Backpropagation



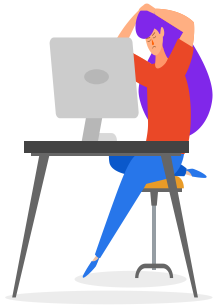
Mạng Nơ-ron nhiều lớp

- Mạng nơ-ron hai lớp cho bài toán XOR

A XOR B = (NOT(A AND B) AND (A OR B))

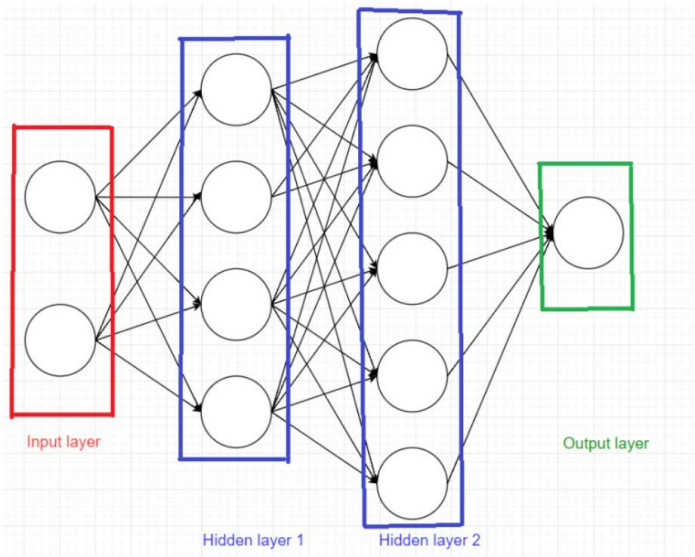


5.2 Mạng Backpropagation

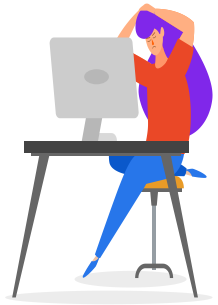


Mạng Nơ-ron nhiều lớp

- Mô hình mạng nhiều lớp:
  - Lớp input
  - Lớp ẩn
  - Lớp ra: số nơ-ron ở lớp ra phụ thuộc vào yêu cầu của bài toán. Ví dụ với bài toán nhị phân, ta chỉ cần 1 nơ-ron ở lớp ra; bài toán phân loài hoa Iris cần 3 nơ-ron vì hoa Iris có 3 loài.

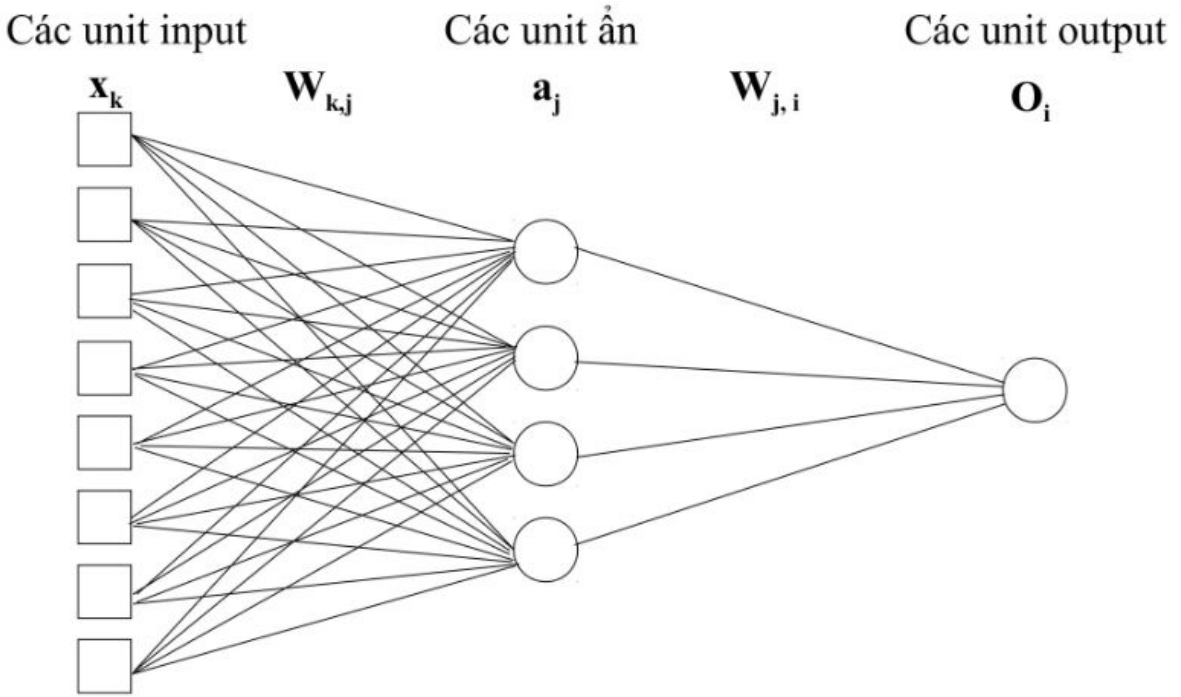


5.2 Mạng Backpropagation

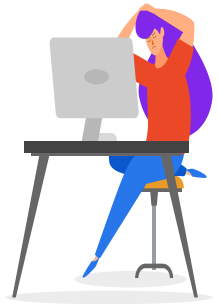


Mạng Nơ-ron nhiều lớp

- **Các Unit:**

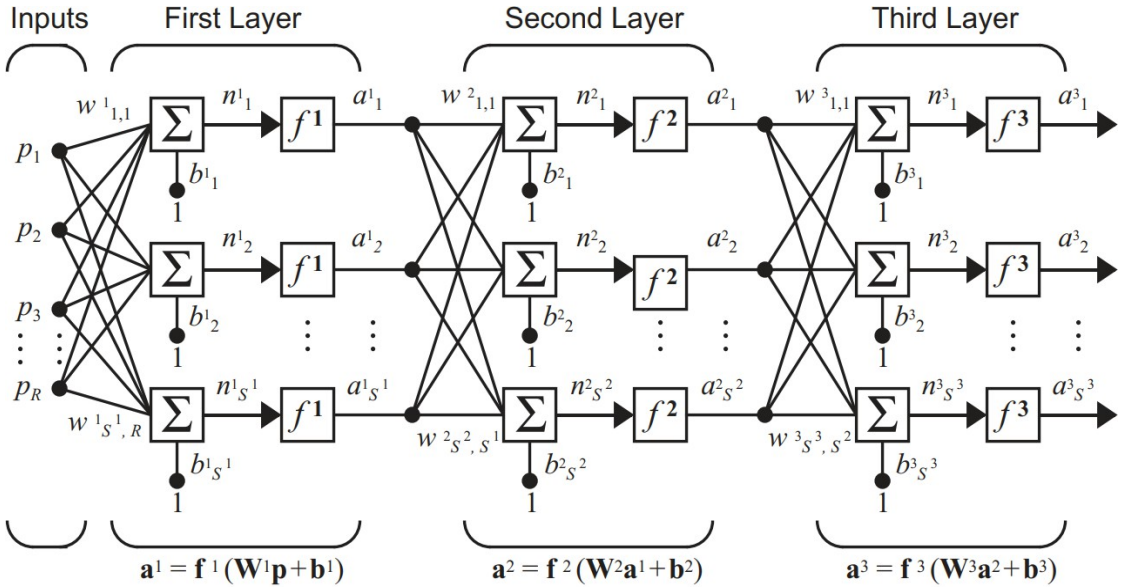


5.2 Mạng Backpropagation



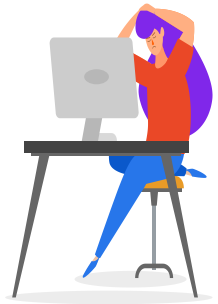
Mạng Nơ-ron nhiều lớp

- Mô hình mạng 3 lớp: đầu ra của lớp trước là đầu vào của lớp sau



$$\mathbf{a}^3 = \mathbf{f}^3(\mathbf{W}^3\mathbf{f}^2(\mathbf{W}^2\mathbf{f}^1(\mathbf{W}^1\mathbf{p}+\mathbf{b}^1)+\mathbf{b}^2)+\mathbf{b}^3)$$

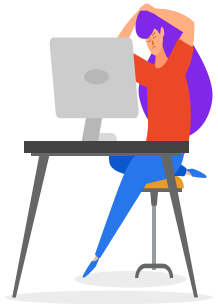
5.2 Mạng Backpropagation



Mạng Nơ-ron nhiều lớp

- Huấn luyện mạng:
    - Pha 1: Lan truyền tiến, từ dữ liệu đầu vào, lan truyền qua các lớp để tính đầu ra thực tế.
    - Pha 2: Lan truyền ngược, dữ liệu từ đầu ra (lớp thứ n) được truyền ngược lại lớp thứ n-1. Quá trình này tiếp tục cho đến khi dữ liệu được chuyển qua lớp input. Ở mỗi lớp trọng số được điều chỉnh theo thuật toán Gradient Descent. Mục tiêu của pha này là điều chỉnh trong số sao cho đầu ra thực tế gần với đầu ra mong muốn nhất.

5.2 Mạng Backpropagation



Mạng Nơ-ron nhiều lớp

- Huấn luyện mạng:
  - Pha 1: Lan truyền tiến

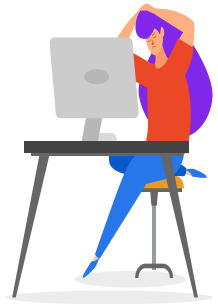
$$z^{(1)} = \begin{bmatrix} z_1^{(1)} \\ z_2^{(1)} \\ z_3^{(1)} \end{bmatrix} = \begin{bmatrix} a_1^{(0)} * w_{11}^{(1)} + a_2^{(0)} * w_{21}^{(1)} + a_3^{(0)} * w_{31}^{(1)} + b_1^{(1)} \\ a_1^{(0)} * w_{12}^{(1)} + a_2^{(0)} * w_{22}^{(1)} + a_3^{(0)} * w_{32}^{(1)} + b_2^{(1)} \\ a_1^{(0)} * w_{13}^{(1)} + a_2^{(0)} * w_{23}^{(1)} + a_3^{(0)} * w_{33}^{(1)} + b_3^{(1)} \end{bmatrix}$$

$$= (W^{(1)})^T * a^{(0)} + b^{(1)}$$

$$a^{(1)} = \sigma(z^{(1)})$$

$$z^{(2)} = (W^{(2)})^T * a^{(1)} + b^{(2)}, a^{(2)} = \sigma(z^{(2)})$$

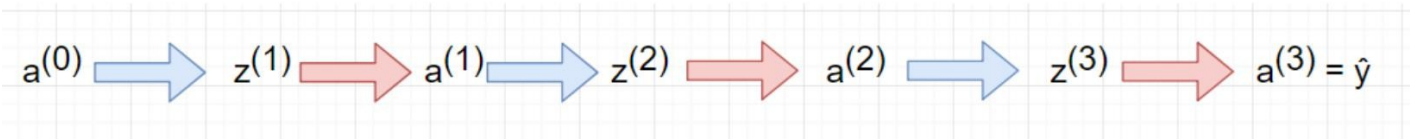
5.2 Mạng Backpropagation



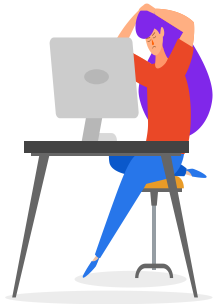
Mạng Nơ-ron nhiều lớp

- Huấn luyện mạng:
  - Pha 1: Lan truyền tiến

$$z^{(2)} = (W^{(2)})^T * a^{(1)} + b^{(2)}$$
$$a^{(2)} = \sigma(z^{(2)})$$
$$z^{(3)} = (W^{(3)})^T * a^{(2)} + b^{(3)}$$
$$\hat{y} = a^{(3)} = \sigma(z^{(3)})$$

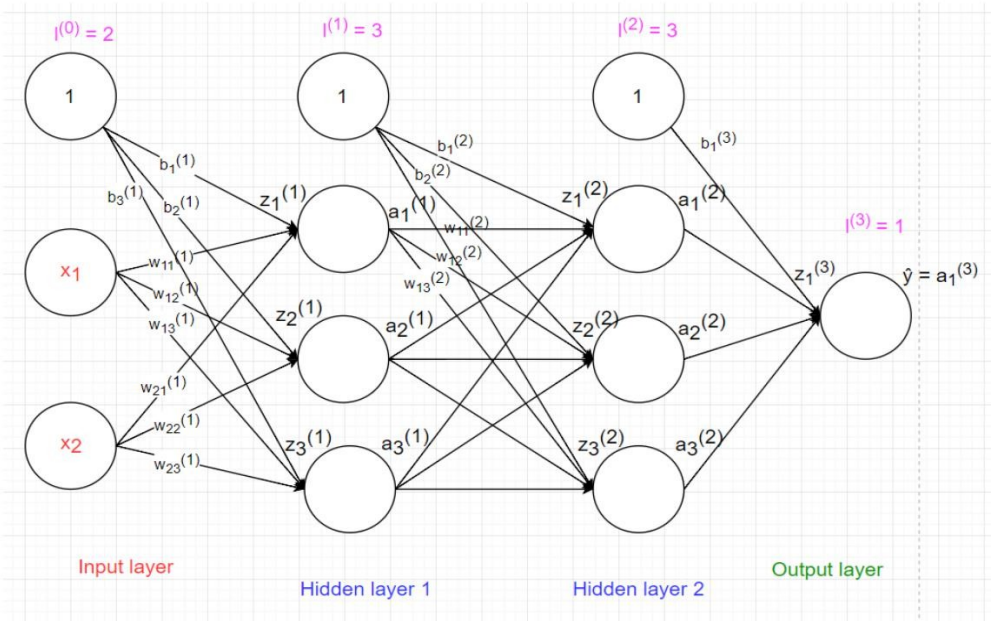


5.2 Mạng Backpropagation



Mạng Nơ-ron nhiều lớp

- Huấn luyện mạng:
  - Pha 2: Lan truyền ngược



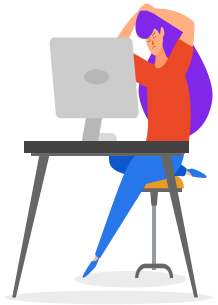
5.3 Implement

Perceptron đơn giản (Chỉ giải được bài toán tuyến tính)

```python
import numpy as np

# Hàm kích hoạt (bậc thang)
def step_function(x):
    return 1 if x >= 0 else 0

# Perceptron training
def perceptron_train(X, y, lr=0.1, epochs=10):
    w = np.zeros(X.shape[1])
    b = 0

    for _ in range(epochs):
        for xi, target in zip(X, y):
            output = step_function(np.dot(xi, w) + b)
            error = target - output
            w += lr * error * xi
            b += lr * error
    return w, b
```

```python
# Dữ liệu bài toán AND
X = np.array([[0,0], [0,1], [1,0], [1,1]])
y = np.array([0, 0, 0, 1])  # AND

w, b = perceptron_train(X, y)
print("Trọng số:", w, "Bias:", b)

# Kiểm tra
for x in X:
    print(f"Input: {x}, Output:
{step_function(np.dot(x, w) + b)}")
```

5.3 Implement



Mạng nhiều lớp (MLP) huấn luyện bằng Backpropagation

Để giải bài toán **XOR**, ta dùng:

- 2 input neurons
- 1 hidden layer với 2 neurons (hàm sigmoid)
- 1 output neuron (hàm sigmoid)
- Loss function: MSE

5.3 Implement

Mạng nhiều lớp (MLP) huấn luyện bằng Backpropagation

```python
import numpy as np
# Sigmoid và đạo hàm
def sigmoid(x):
    return 1 / (1 + np.exp(-x))

def sigmoid_deriv(x):
    return x * (1 - x)
# Dữ liệu XOR
X = np.array([[0,0], [0,1], [1,0], [1,1]])
y = np.array([[0], [1], [1], [0]])
# Khởi tạo trọng số
np.random.seed(42)
input_size = 2
hidden_size = 2
output_size = 1

W1 = np.random.randn(input_size, hidden_size)
b1 = np.zeros((1, hidden_size))
W2 = np.random.randn(hidden_size, output_size)
b2 = np.zeros((1, output_size))

# Huấn luyện
lr = 0.1
epochs = 10000
```

```python
for epoch in range(epochs):
    # Forward pass
    z1 = np.dot(X, W1) + b1
    a1 = sigmoid(z1)
    z2 = np.dot(a1, W2) + b2
    y_pred = sigmoid(z2)
    # Loss (MSE)
    loss = np.mean((y - y_pred) ** 2)
    # Backward pass
    d_loss = y_pred - y
    d_output = d_loss * sigmoid_deriv(y_pred)
    d_hidden = d_output.dot(W2.T) * sigmoid_deriv(a1)
    # Cập nhật trọng số
    W2 -= a1.T.dot(d_output) * lr
    b2 -= np.sum(d_output, axis=0, keepdims=True) * lr
    W1 -= X.T.dot(d_hidden) * lr
    b1 -= np.sum(d_hidden, axis=0, keepdims=True) * lr
    if epoch % 1000 == 0:
        print(f"Epoch {epoch}, Loss: {loss:.4f}")
# Kiểm tra kết quả
print("\nKết quả dự đoán sau huấn luyện:")
for i in range(4):
    print(f"Input: {X[i]} → Output: {y_pred[i][0]:.4f} → Rounded:
{round(y_pred[i][0])}")
```

5.4 Ứng dụng giải bài toán xấp xỉ hàm

Mạng nhiều lớp (MLP) huấn luyện bằng Backpropagation

Một trong những ứng dụng mạnh mẽ nhất của mạng nơ-ron nhân tạo (ANN), đặc biệt là **mạng Backpropagation**, là **xấp xỉ hàm phi tuyến**

Mục tiêu của xấp xỉ hàm:

•Tìm một **hàm gần đúng** $\hat{f}(x)$ sao cho $\hat{f}(x) \approx f(x)$ với mọi x∈D

•Thường áp dụng cho các hàm phi tuyến, không có công thức giải chính xác, hoặc không xác định rõ

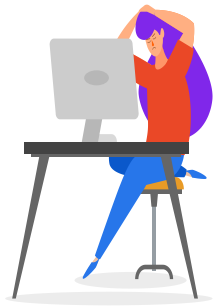
Cách thực hiện:

1.Tạo tập dữ liệu đầu vào – đầu ra: $(x_i, f(x_i))$

2.Huấn luyện mạng nơ-ron MLP:

1. Input layer: chứa biến x
2. Hidden layers: đủ số lượng để nắm bắt đặc trưng phi tuyến
3. Output layer: giá trị $\hat{f}(x)$

3.Đánh giá độ chính xác xấp xỉ qua sai số (MSE, RMSE, MAE)



5.4 Ứng dụng giải bài toán xấp xỉ hàm

Ví dụ minh hoạ

Ví dụ 1: Xấp xỉ hàm sin

•**Bài toán**: Tìm mạng nơ-ron xấp xỉ $f(x) = \sin(x), x \in [0, 2\pi]$

•**Thực hiện**:

- Tạo tập dữ liệu mẫu: 100 điểm từ $[0, 2\pi]$
- Huấn luyện MLP với 1 input, 1 output, 1–2 hidden layers

•**Kết quả**: Mạng có thể vẽ được đường cong gần khớp với đồ thị hàm sin

5.4 Ứng dụng giải bài toán xấp xỉ hàm

Ví dụ minh hoạ

```
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# Dữ liệu huấn luyện
X = np.linspace(0, 2*np.pi, 100)
y = np.sin(X)

# Xây dựng model
model = Sequential([
    Dense(10, input_dim=1, activation='tanh'),
    Dense(10, activation='tanh'),
    Dense(1)
])
model.compile(optimizer='adam', loss='mse')
model.fit(X, y, epochs=300, verbose=0)
```

```
# Dự đoán
y_pred = model.predict(X)

# Kết quả
plt.plot(X, y, label='Hàm gốc sin(x)')
plt.plot(X, y_pred, label='Hàm xấp xỉ',
linestyle='dashed')
plt.legend()
plt.title("Xấp xỉ hàm sin(x)")
plt.show()
```

5.4 Ứng dụng giải bài toán xấp xỉ hàm

Ví dụ minh hoạ

Ví dụ 2: Xấp xỉ hàm bật (step function)

•**Hàm mục tiêu**:

$$f(x) = \begin{cases} 0, nếu\ x\ < 0 \\ 1, nếu\ x\ \geq 0 \end{cases}$$

•**Thử thách**: Đây là hàm không liên tục → mạng cần dùng sigmoid để xấp xỉ mượt hàm này

•**Ứng dụng**: Xây dựng bộ phân loại nhị phân

5.4 Ứng dụng giải bài toán xấp xỉ hàm

Ví dụ minh hoạ

```
# Dữ liệu step function
X = np.linspace(-1, 1, 100)
y = np.where(X < 0, 0, 1)  # Step function

# Model mạng nơ-ron
model = Sequential([
    Dense(10, input_dim=1, activation='tanh'),
    Dense(1, activation='sigmoid')
])
model.compile(optimizer='adam', loss='mse')
model.fit(X, y, epochs=300, verbose=0)

# Dự đoán
y_pred = model.predict(X)

# Vẽ kết quả
plt.plot(X, y, label='Hàm bật gốc')
plt.plot(X, y_pred, label='Hàm xấp xỉ sigmoid',
linestyle='dashed')
plt.legend()
plt.title("Xấp xỉ hàm bật")
plt.show()
```

5.4 Ứng dụng giải bài toán xấp xỉ hàm

Ví dụ minh hoạ

Ví dụ 3: Xấp xỉ hàm tuyến tính nhiều biến

- **Hàm mục tiêu**: $f(x_1, x_2) = 3x_1 + 2x_2 + 1$
- **Mục đích**: Kiểm tra khả năng học của mạng nơ-ron với bài toán đơn giản
- **Kết quả**: Mạng có thể học đúng trọng số gần như chính xác sau vài epoch

5.4 Ứng dụng giải bài toán xấp xỉ hàm

Ví dụ minh hoạ

```
from sklearn.model_selection import train_test_split

# Tạo dữ liệu ngẫu nhiên
X = np.random.rand(1000, 2)
y = 3 * X[:, 0] + 2 * X[:, 1] + 1

# Tách tập
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2)

# Mô hình
model = Sequential([
    Dense(10, input_dim=2, activation='relu'),
    Dense(1)
])
model.compile(optimizer='adam', loss='mse')
model.fit(X_train, y_train, epochs=200, verbose=0)
```

```
# Đánh giá
loss = model.evaluate(X_test, y_test, verbose=0)
print("MSE trên tập test:", loss)

# Kiểm tra một vài mẫu
test_sample = np.array([[1, 2], [0.5, 0.5]])
prediction = model.predict(test_sample)
print("Dự đoán:", prediction)
print("Giá trị thực:", 3*test_sample[:,0] +
2*test_sample[:,1] + 1)
```