Mạng nơ ron nhân tạo

Ngô Minh Nhựt

Bộ môn Công nghệ Tri thức

2024

Biểu diễn mô hình và lan truyền tới

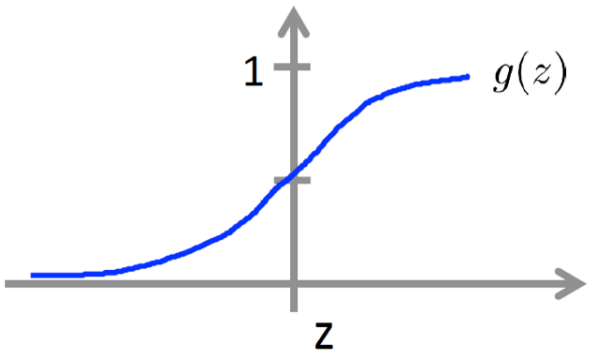
Phần 1

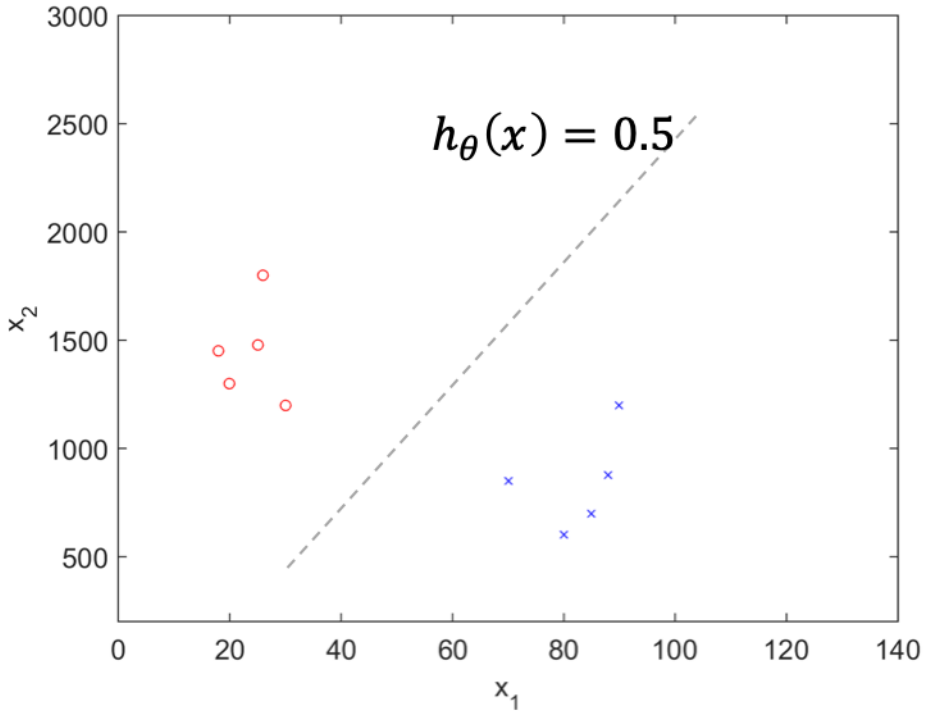
Hồi qui logistic

- Bộ phân lớp tuyến tính

$$h_\theta(x) = g(\theta^T x)$$

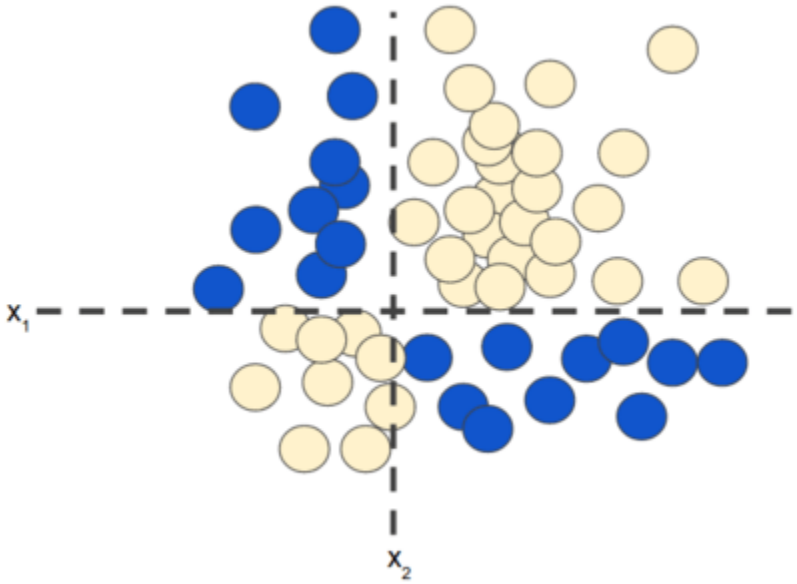
$$g(z) = \frac{1}{1 + e^{-z}}$$





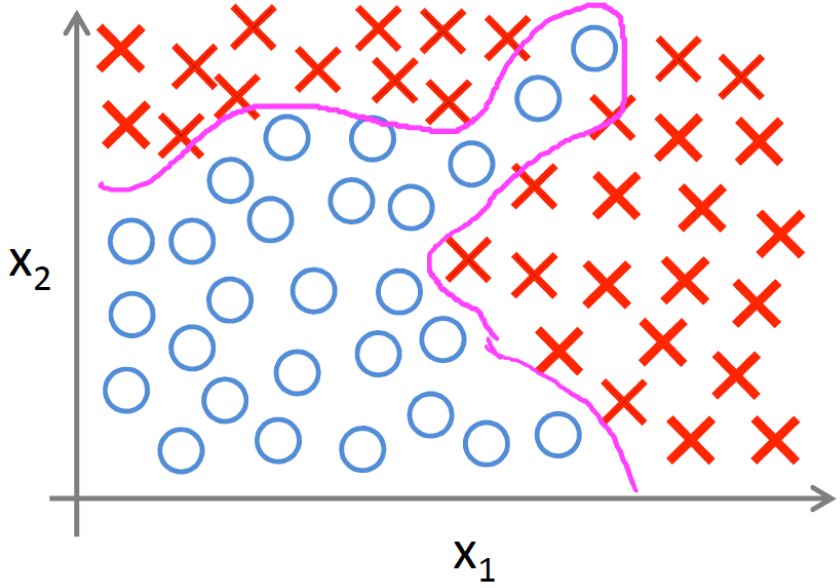
Phân lớp phi tuyến

- Phi tuyến (nonlinear): dữ liệu không thể phân lớp bằng mô hình tuyến tính (hyperplane)



Source: Google Crash Course

Phân lớp phi tuyến



$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \theta_4 x_1^2 x_2 + \theta_5 x_1^3 x_2 + \theta_6 x_2 x_2^2 + \ldots)$$

Feature mapping

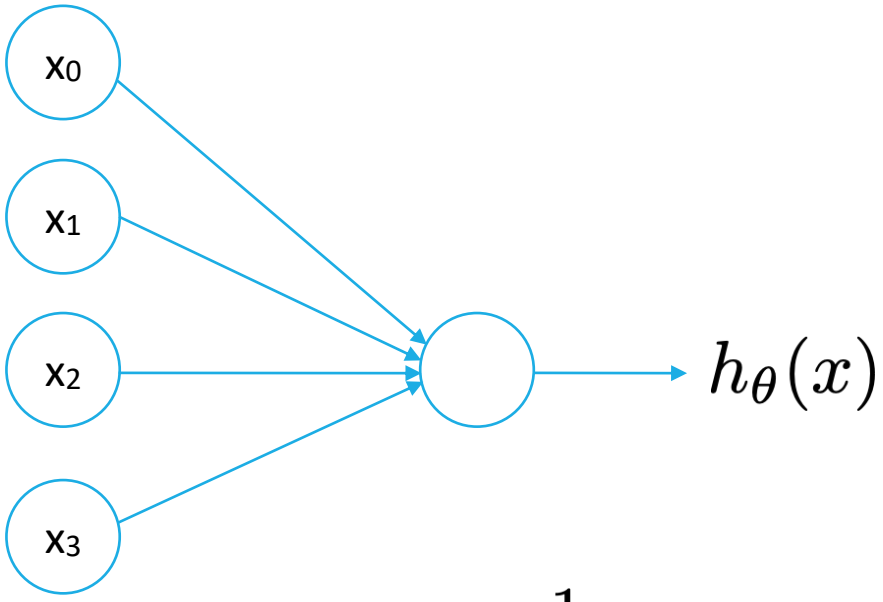
- x_1 = size
- x_2 = number of bedrooms
- x_3 = number of floors
- x_4 = age
- …

- Ánh xạ như thế nào?
- Bao nhiêu ánh xạ?
- Bao nhiêu đặc trưng?

Phân lớp phi tuyến với hồi qui logistic cần rất nhiều feature

Source: Andrew Ng

Hồi qui logistic

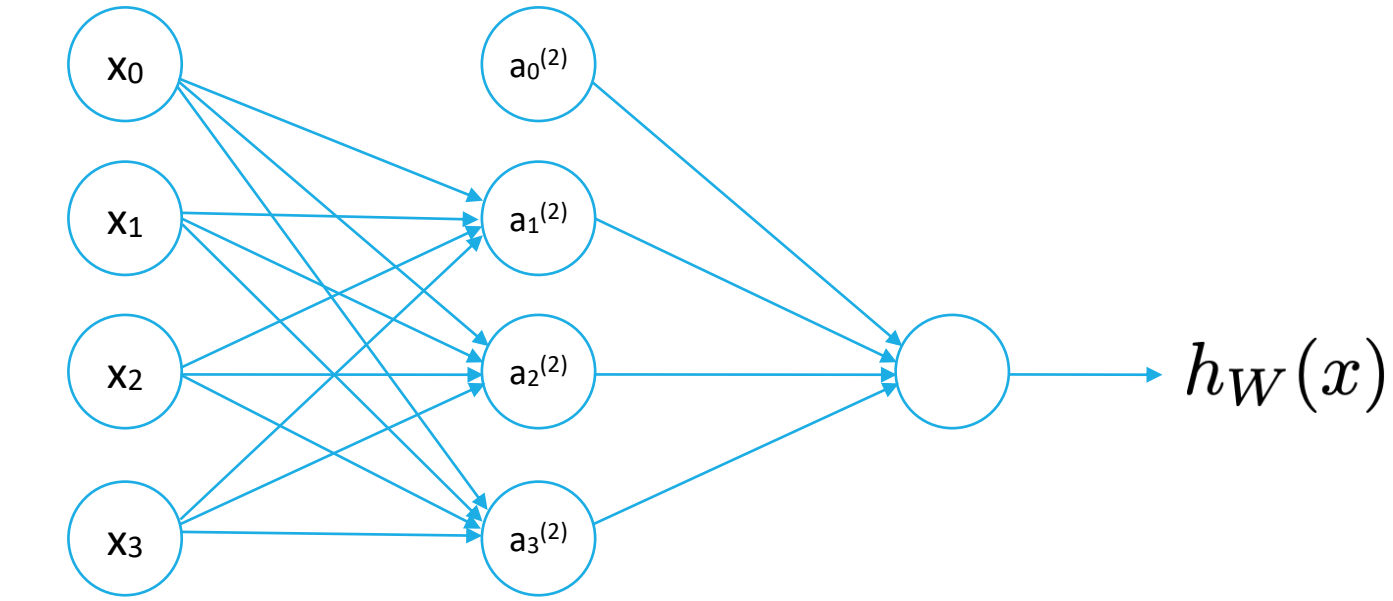


$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

Hàm truyền sigmoid

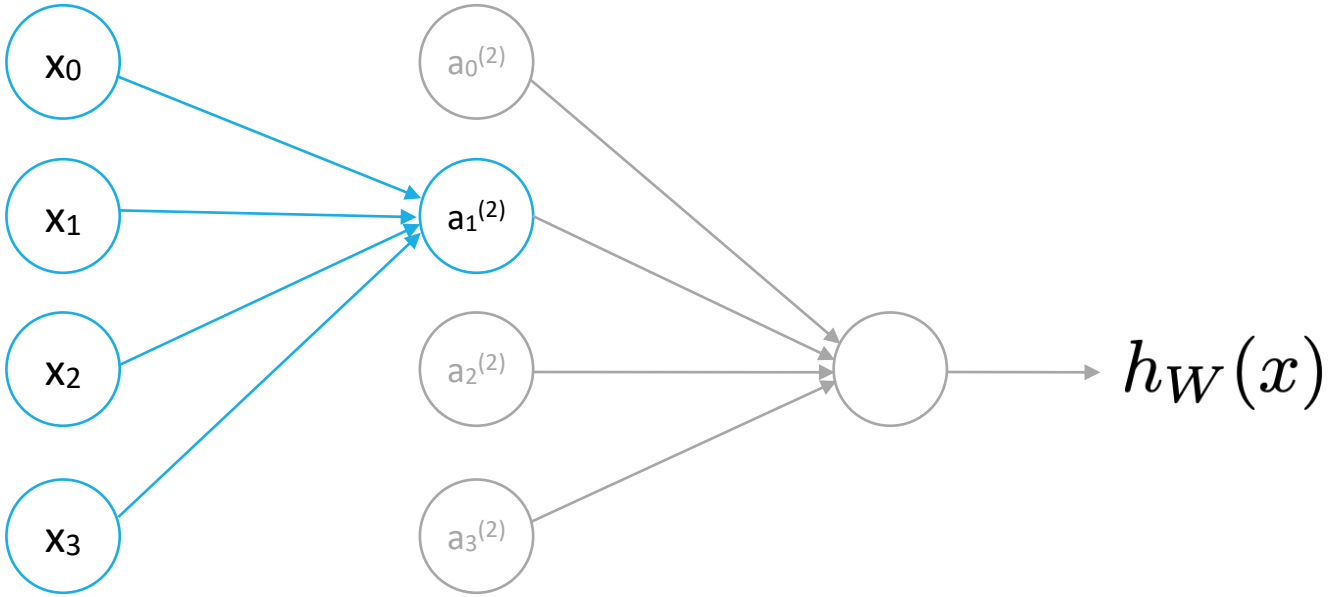
Mạng nơ ron


x_0
x_1
x_2
x_3
$a_0^{(2)}$
$a_1^{(2)}$
$a_2^{(2)}$
$a_3^{(2)}$
$h_W(x)$
Lớp 1
Lớp 2
Lớp 3
“input layer”
“output layer”

Hàm truyền tại nút

- Activation/transfer function

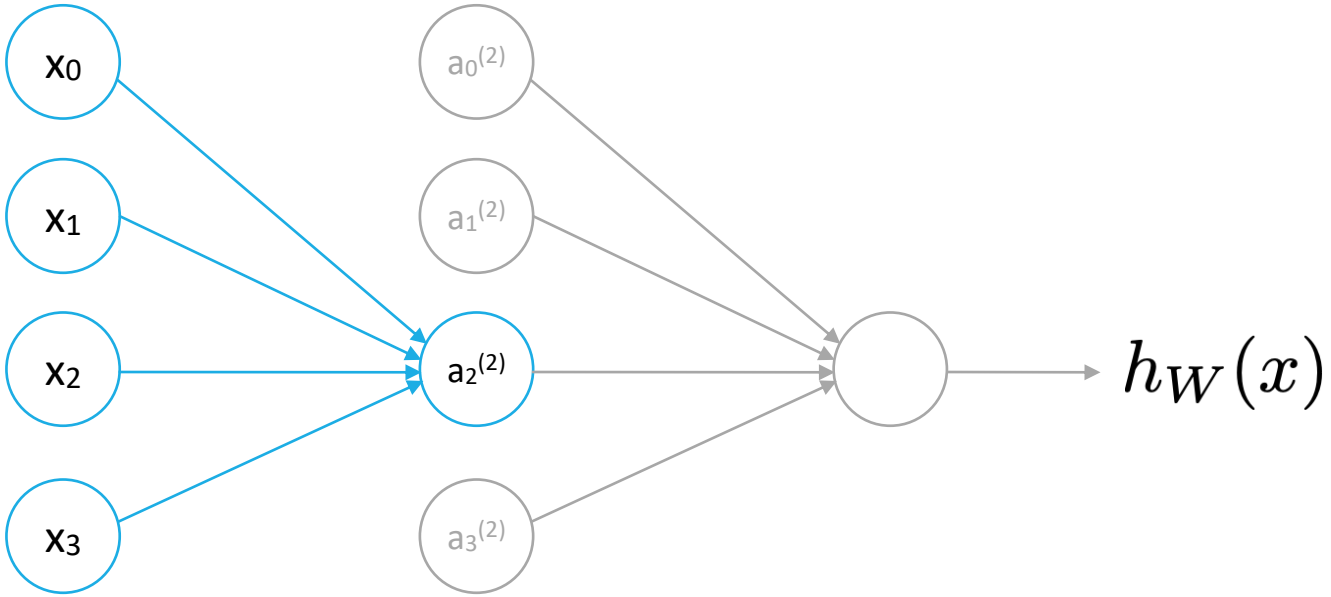
$$g(z) = \frac{1}{1 + e^{-z}}$$



$$a_1^{(2)} = g\left(W_{10}^{(1)}x_0 + W_{11}^{(1)}x_1 + W_{12}^{(1)}x_2 + W_{13}^{(1)}x_3\right)$$

Hàm truyền tại nút

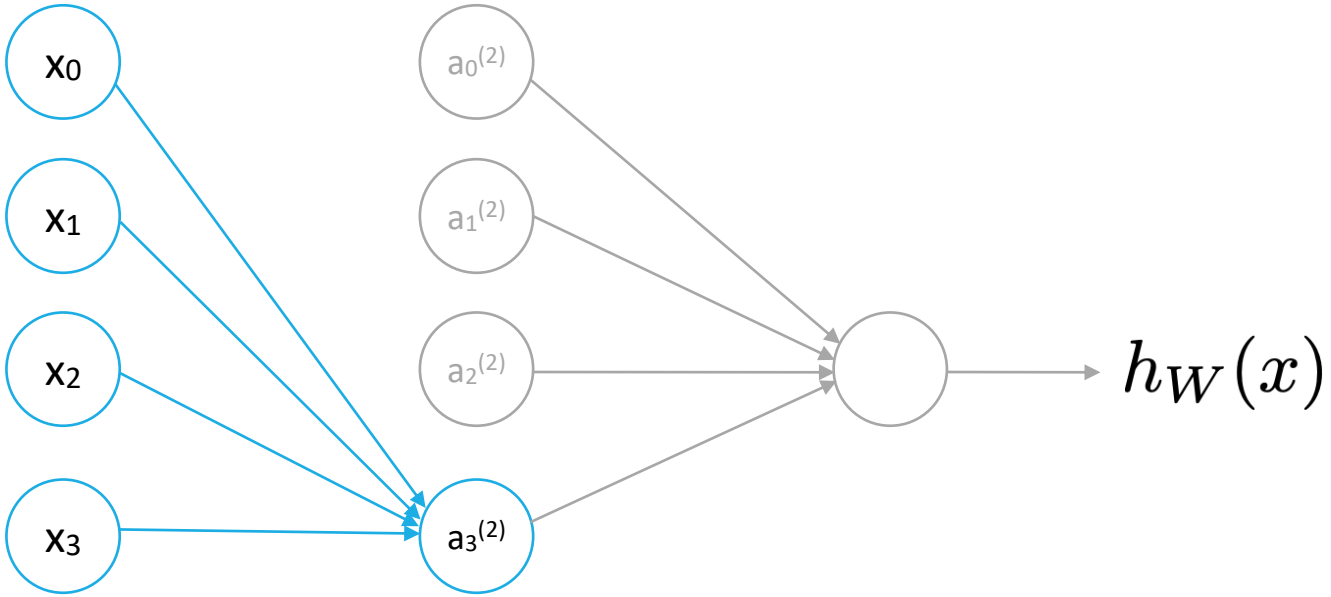
$$g(z) = \frac{1}{1 + e^{-z}}$$



$$a_2^{(2)} = g\left(W_{20}^{(1)}x_0 + W_{21}^{(1)}x_1 + W_{22}^{(1)}x_2 + W_{23}^{(1)}x_3\right)$$

Hàm truyền tại nút

$$g(z) = \frac{1}{1 + e^{-z}}$$

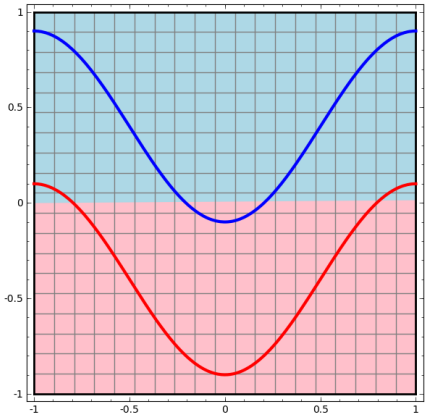


$$a_3^{(2)} = g\left(W_{30}^{(1)}x_0 + W_{31}^{(1)}x_1 + W_{32}^{(1)}x_2 + W_{33}^{(1)}x_3\right)$$

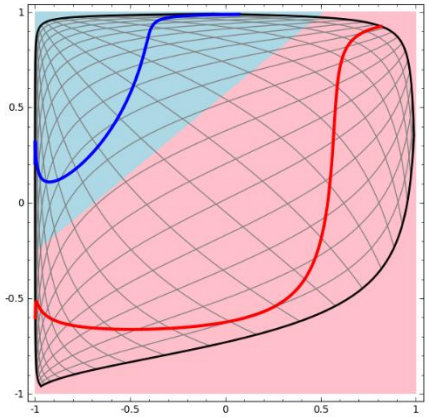
Hàm truyền tại nút

Vì sao mạng nơ ron cần hàm truyền?

- Tính phi tuyến (nonlinearity) của hàm truyền giúp chuyển đổi dữ liệu sang không gian (representation) mới
- Dữ liệu ở dạng mới được kỳ vọng phân tách tuyến tính



Original



New representation

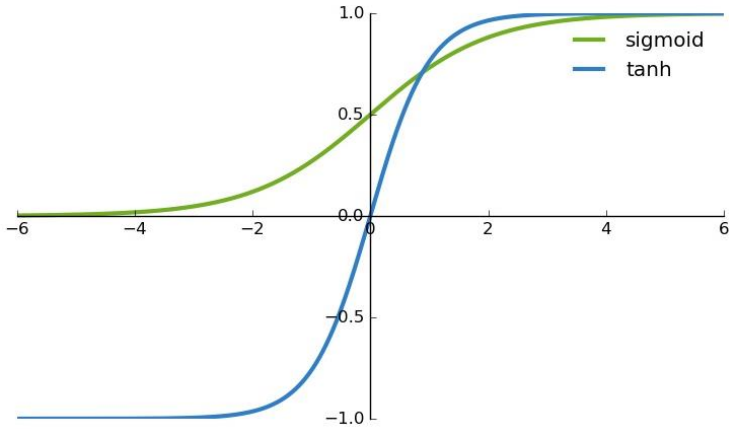
Không có biến đổi phi tuyến, mạng nơ rơn hoạt động như một mô hình tuyến tính.

Source: https://colah.github.io/posts/2014-03-NN-Manifolds-Topology/

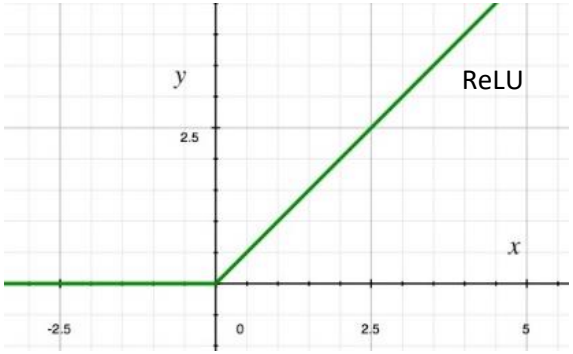
Các hàm truyền phổ biến

- Hàm sigmoid

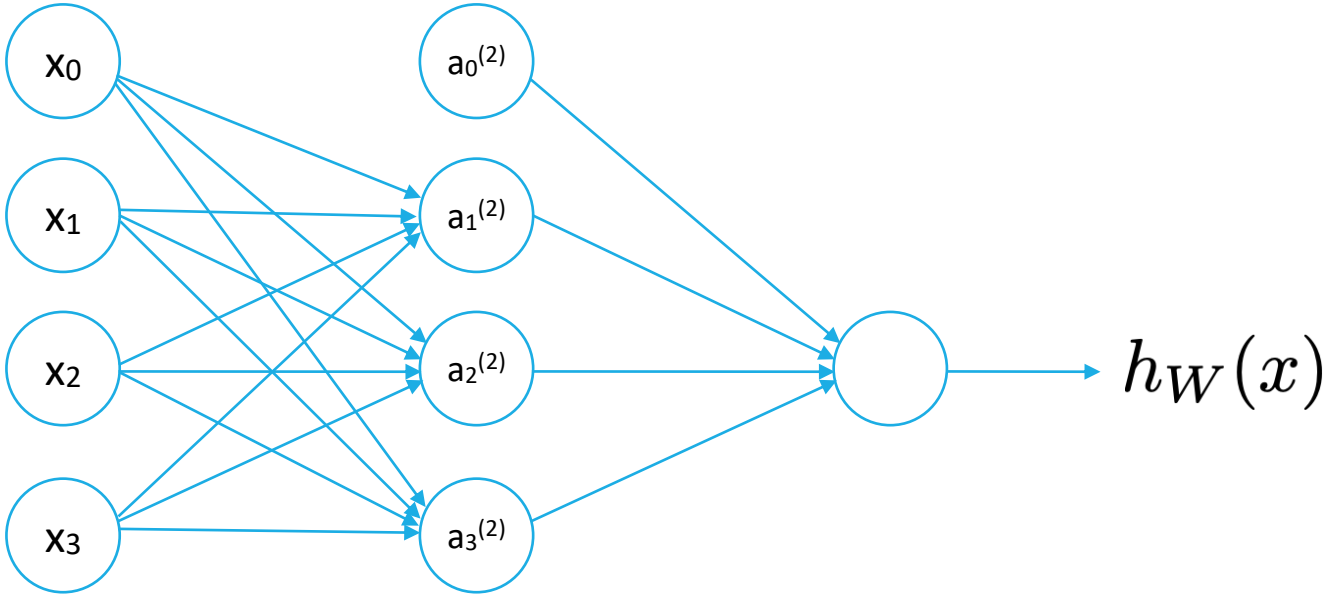
$$g(z) = \frac{1}{1 + e^{-z}}$$



- Hàm tanh
  - $\tanh(z) = \frac{e^{2z}-1}{e^{2z}+1}$
  - Derivative: $1 - \tanh^2(z)$
- Hàm ReLU (rectified linear unit)
  - $g(z) = \max(0, z)$



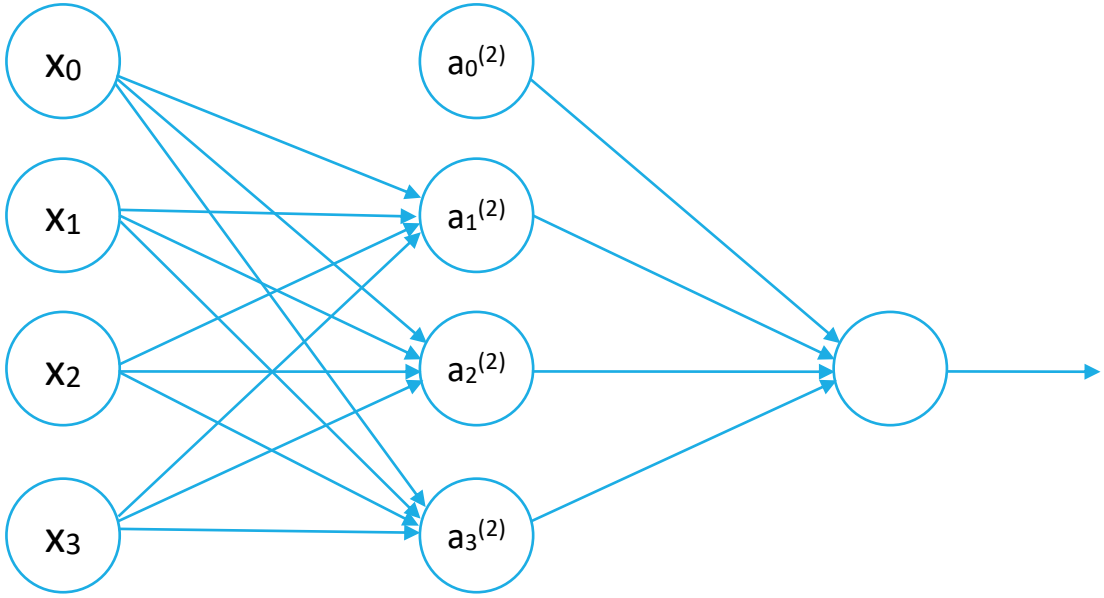
Ma trận trọng số



- $a_i^{(j)}$: giá trị (activation) của nút i trong lớp j
- $W^{(j)}$: ma trận trọng số để ánh xạ giá trị ở lớp j sang lớp j+1
- Nếu mạng nơ ron có s_j nút ở lớp j và s_{j+1} nút ở lớp j+1, $W^{(j)}$ có kích thước:

  s_{j+1} x (s_j + 1)

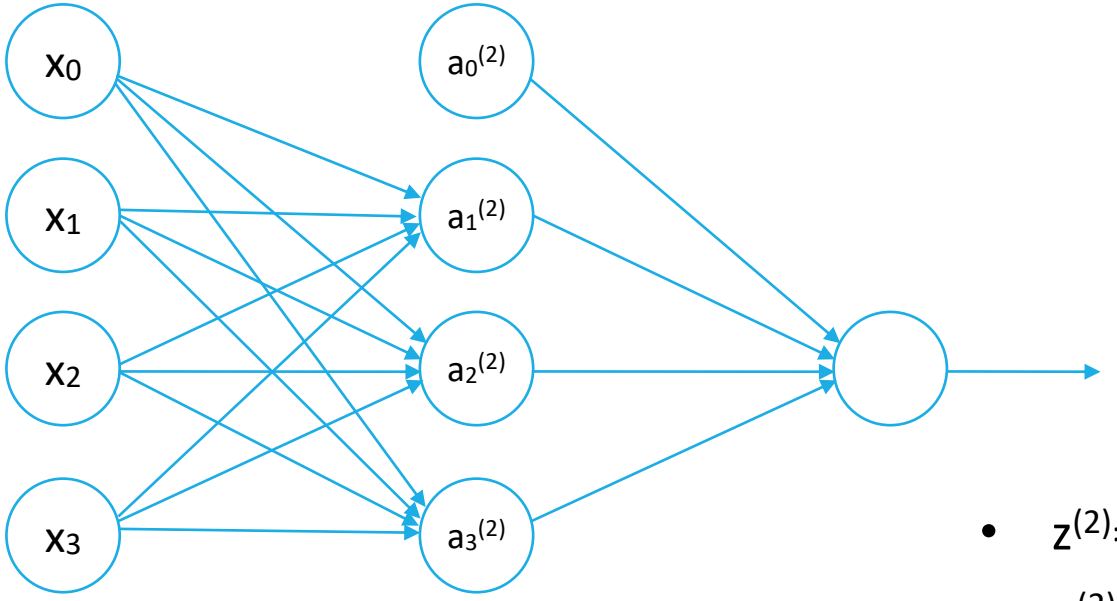
Lan truyền tới



$$a_1^{(2)} = g\left(W_{10}^{(1)}x_0 + W_{11}^{(1)}x_1 + W_{12}^{(1)}x_2 + W_{13}^{(1)}x_3\right)$$
$$a_2^{(2)} = g\left(W_{20}^{(1)}x_0 + W_{21}^{(1)}x_1 + W_{22}^{(1)}x_2 + W_{23}^{(1)}x_3\right)$$
$$a_3^{(2)} = g\left(W_{30}^{(1)}x_0 + W_{31}^{(1)}x_1 + W_{32}^{(1)}x_2 + W_{33}^{(1)}x_3\right)$$

$$h_W(x) = a_1^{(3)} = g\left(W_{10}^{(2)}a_0^{(2)} + W_{11}^{(2)}a_1^{(2)} + W_{12}^{(2)}a_2^{(2)} + W_{13}^{(2)}a_3^{(2)}\right)$$

Biểu diễn bằng vector



$$a_1^{(2)} = g\left(W_{10}^{(1)}x_0 + W_{11}^{(1)}x_1 + W_{12}^{(1)}x_2 + W_{13}^{(1)}x_3\right)$$

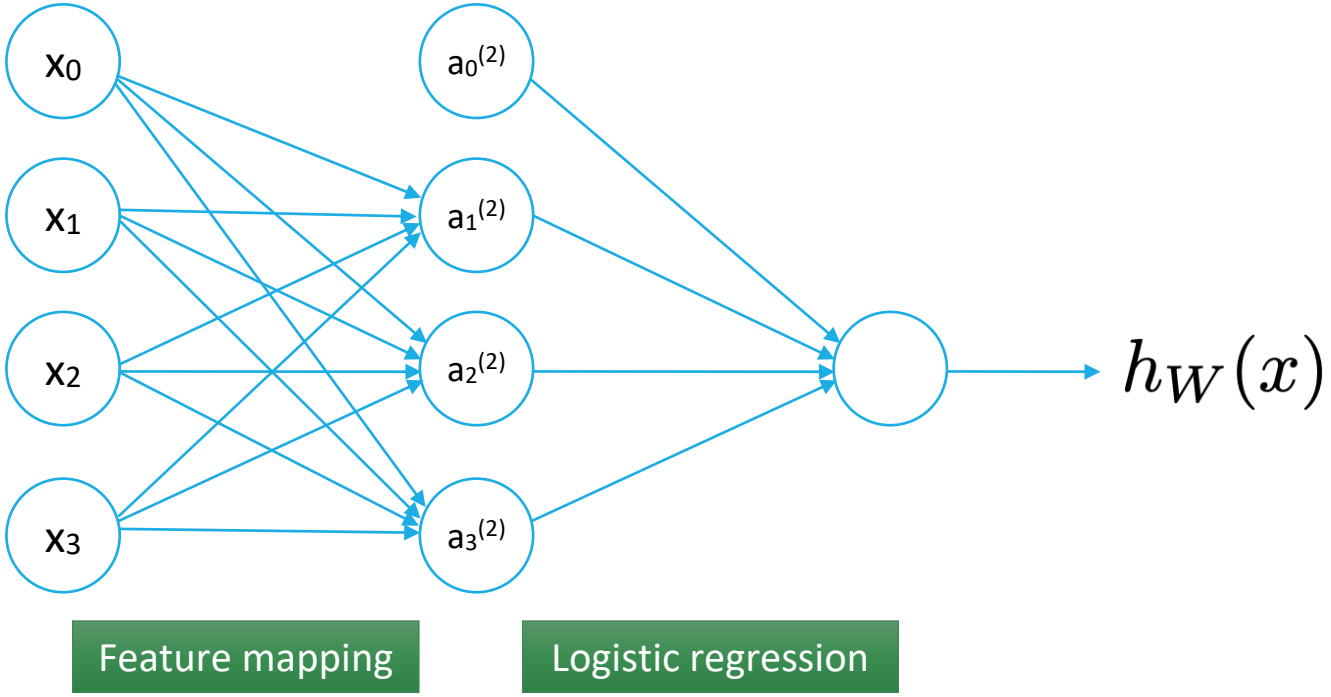
$$a_2^{(2)} = g\left(W_{20}^{(1)}x_0 + W_{21}^{(1)}x_1 + W_{22}^{(1)}x_2 + W_{23}^{(1)}x_3\right)$$

$$a_3^{(2)} = g\left(W_{30}^{(1)}x_0 + W_{31}^{(1)}x_1 + W_{32}^{(1)}x_2 + W_{33}^{(1)}x_3\right)$$

$$h_W(x) = a_1^{(3)} = g\left(W_{10}^{(2)}a_0^{(2)} + W_{11}^{(2)}a_1^{(2)} + W_{12}^{(2)}a_2^{(2)} + W_{13}^{(2)}a_3^{(2)}\right)$$

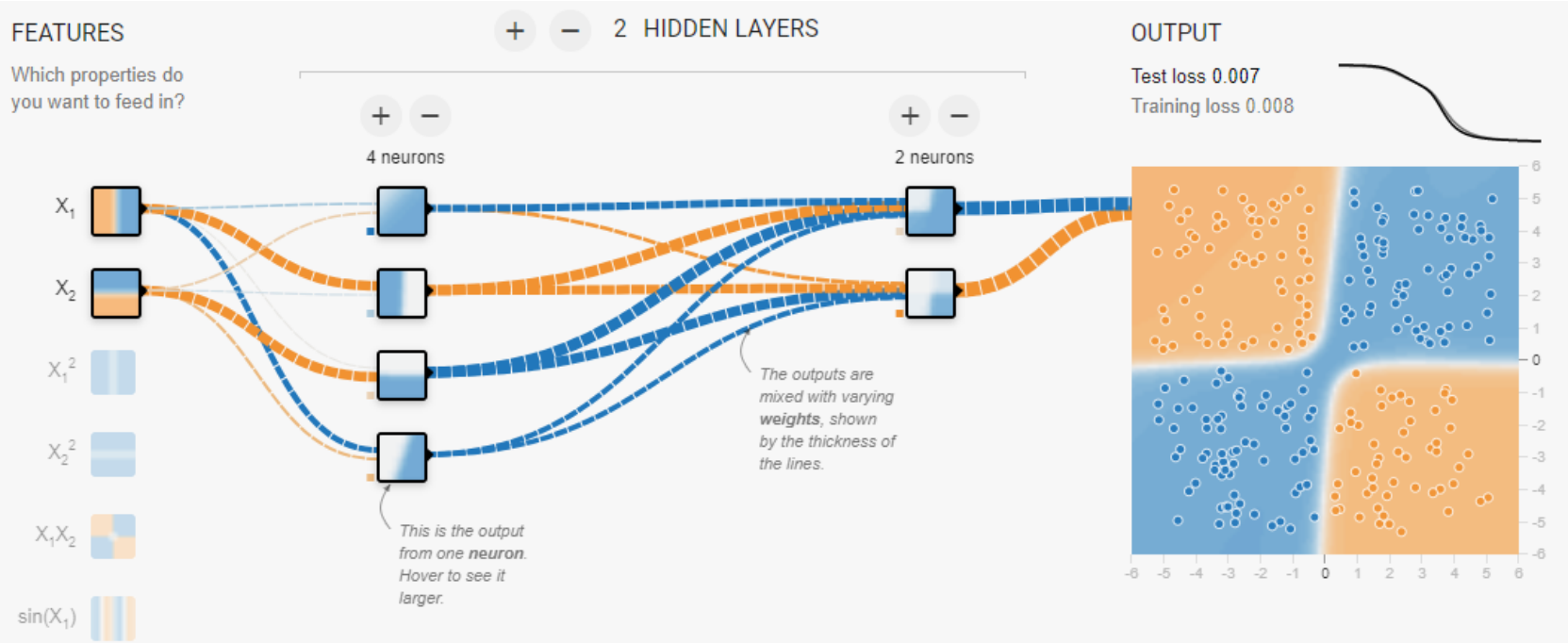
- $z^{(2)}=W^{(1)}x$
- $a^{(2)} = g(z^{(2)})$
- Thêm $a_0^{(2)} = 1$
- $z^{(3)} = W^{(2)}a^{(2)}$
- $h_w(x) = a^{(3)} = g(z^{(3)})$

Mạng nơ ron tự học đặc trưng



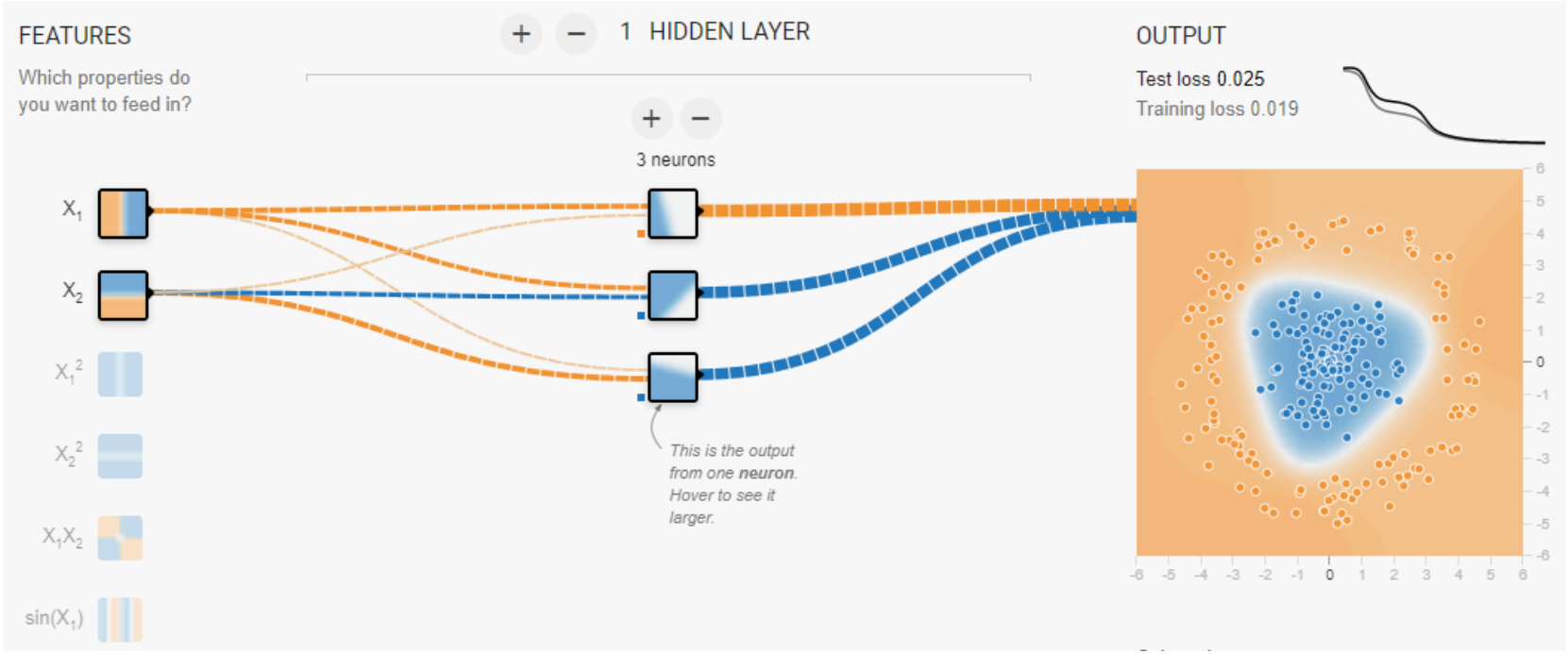
$$h_W(x) = a_1^{(3)} = g\left(W_{10}^{(2)}a_0^{(2)} + W_{11}^{(2)}a_1^{(2)} + W_{12}^{(2)}a_2^{(2)} + W_{13}^{(2)}a_3^{(2)}\right)$$

Mạng nơ ron tự học đặc trưng



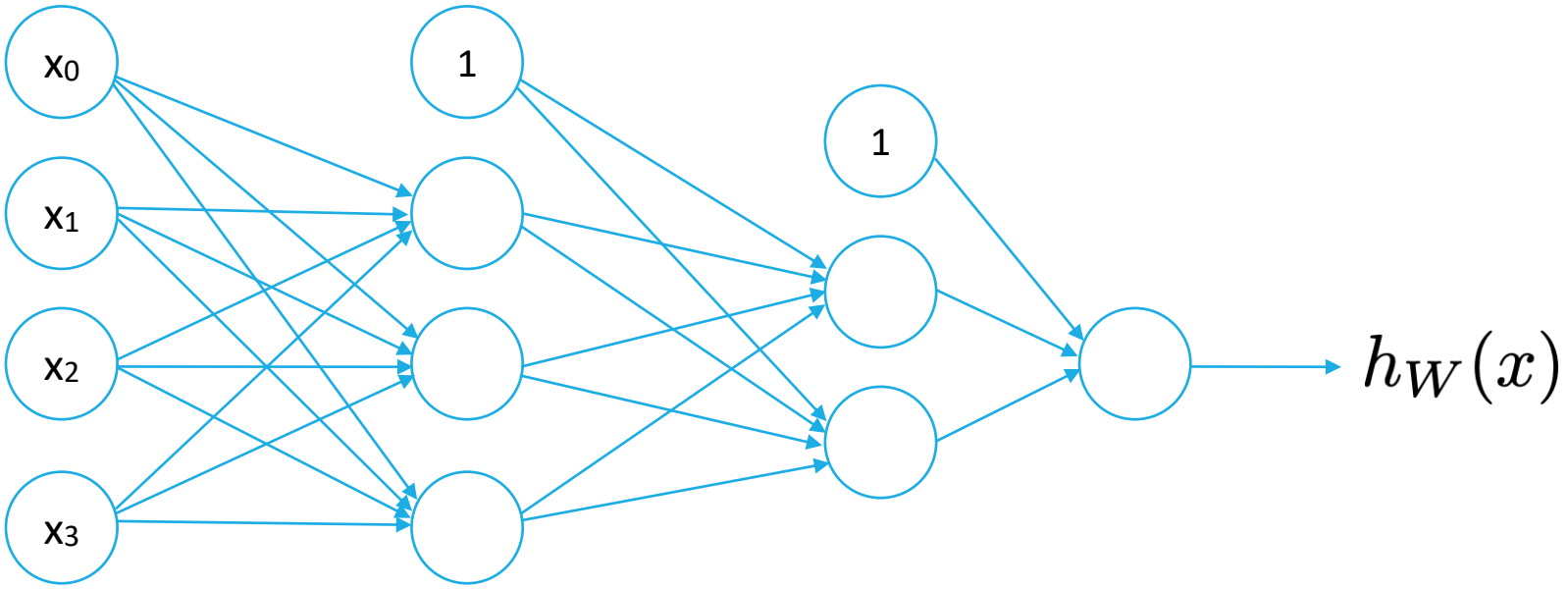
Source: https://playground.tensorflow.org

Mạng nơ ron tự học đặc trưng



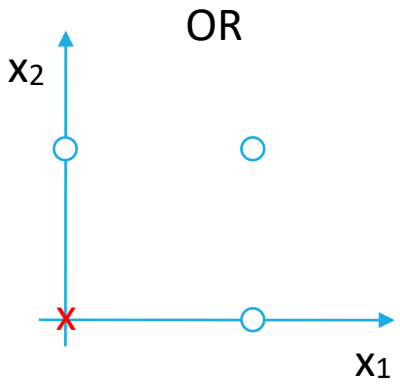
Source: https://playground.tensorflow.org

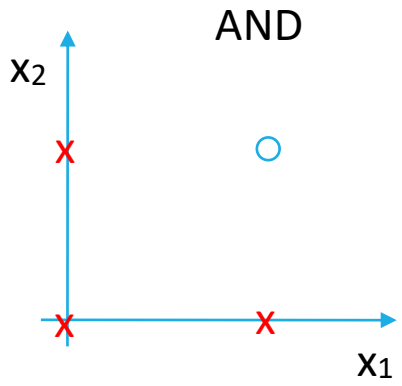
Các kiến trúc khác

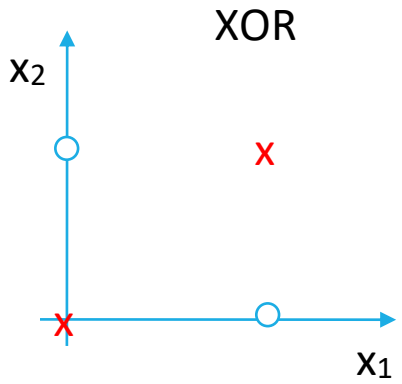


Bộ phân lớp phi tuyến

- Hàm tuyến tính
  - OR
  - AND
- Hàm phi tuyến XOR
  - XOR(x1, x2)
  - a1 = AND(x1, NOT(x2))
  - a2 = AND(NOT(x1), x2)
  - y = OR(a1, a2)

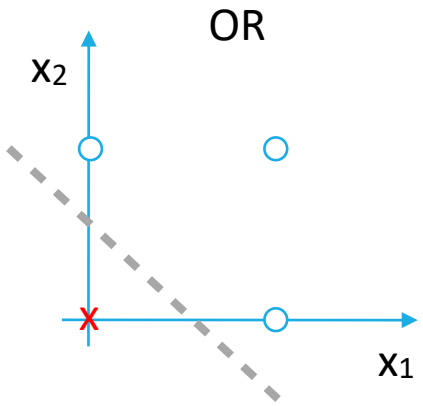


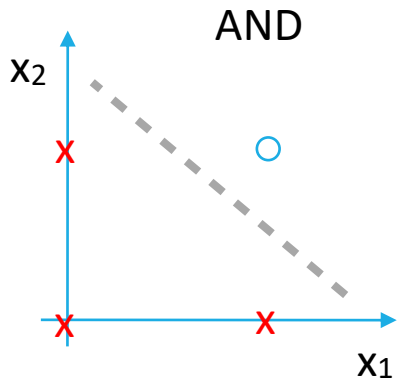


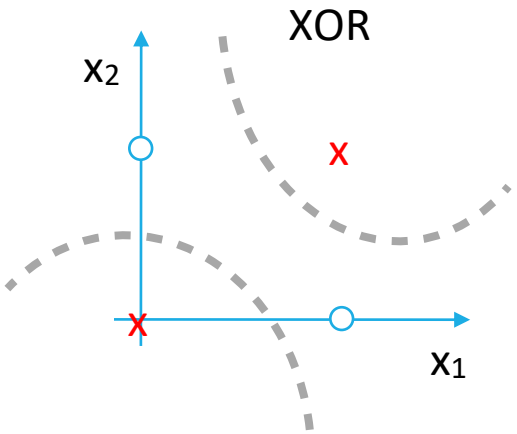


Bộ phân lớp phi tuyến

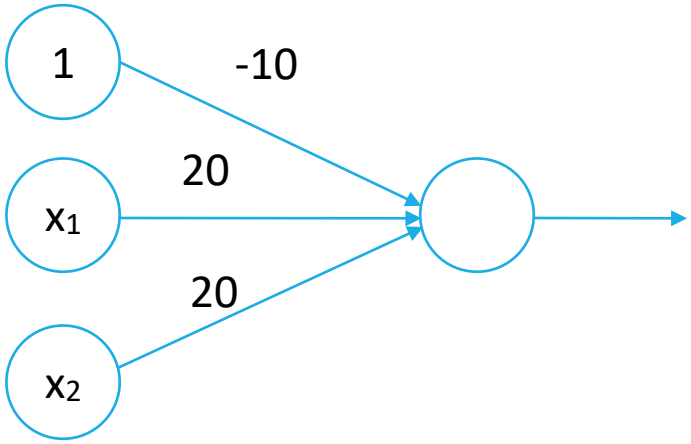
- Hàm tuyến tính
  - OR
  - AND
- Hàm phi tuyến XOR
  - XOR(x1, x2)
  - a1 = AND(x1, NOT(x2))
  - a2 = AND(NOT(x1), x2)
  - y = OR(a1, a2)



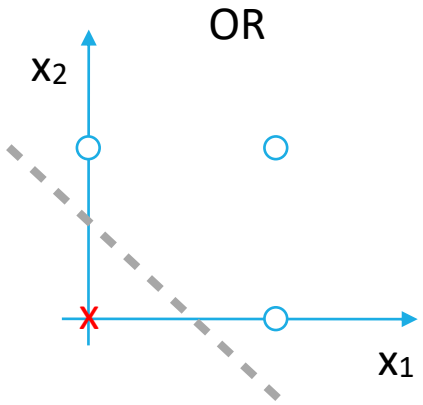




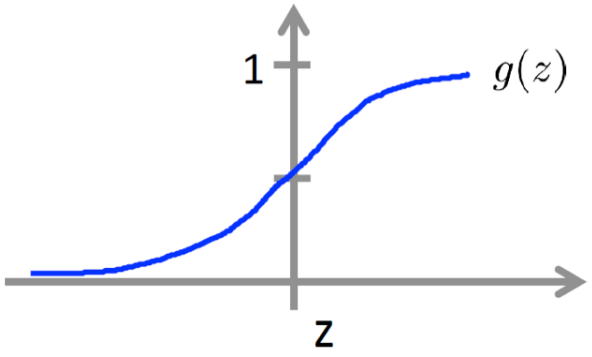
Hàm OR



$h_W(x)$

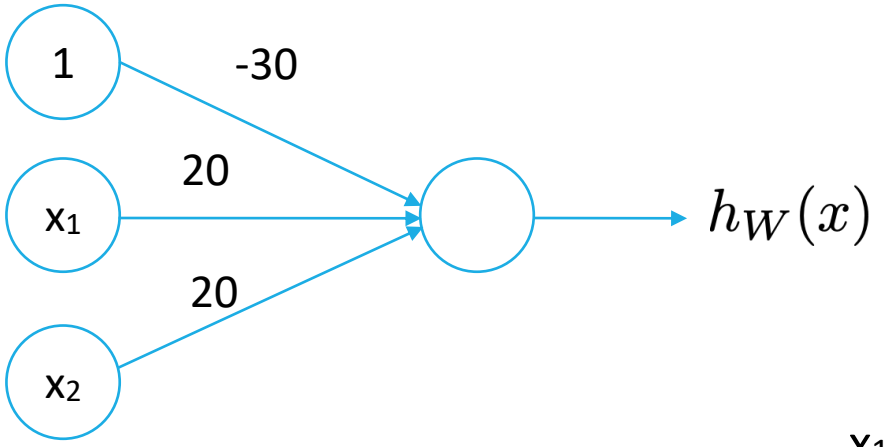


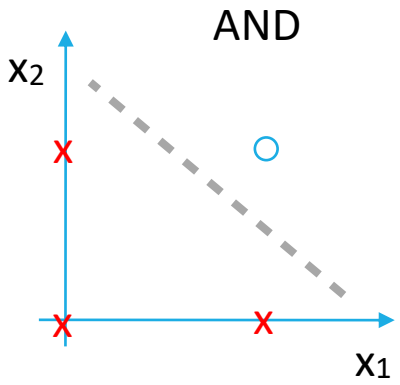
$$h_W(x) = g(-10 + 20x_1 + 20x_2)$$



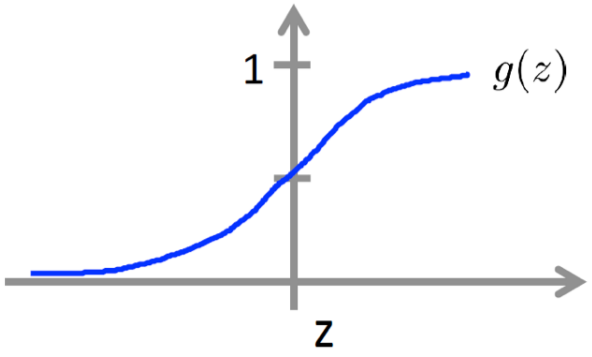
x_1	x_2	$h_W(x)$
0	0	0
0	1	1
1	0	1
1	1	1

Hàm AND



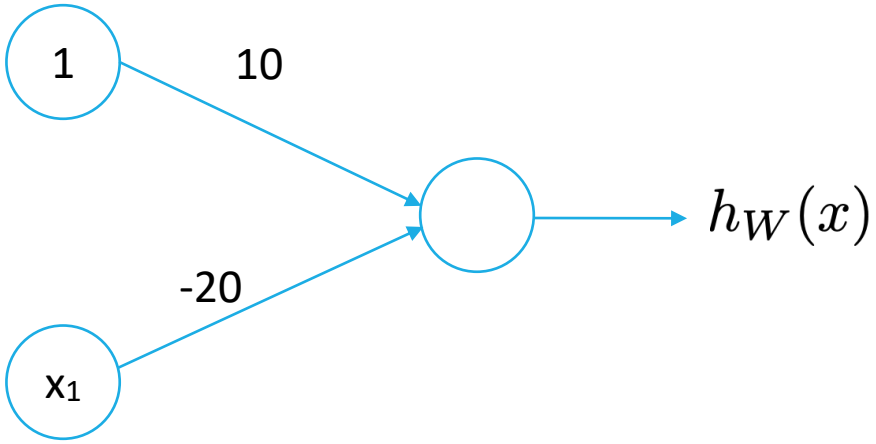


$$h_W(x) = g(-30 + 20x_1 + 20x_2)$$

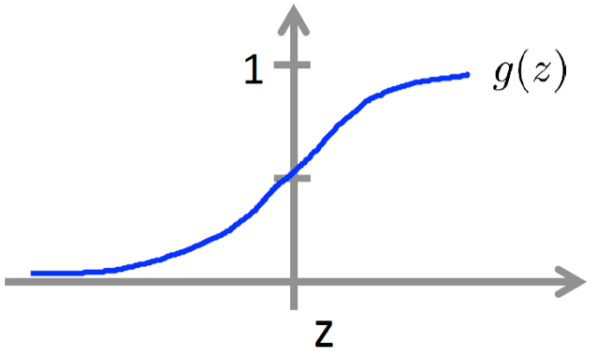


x1	x2	$h_W(x)$
0	0	0
0	1	0
1	0	0
1	1	1

Hàm NOT

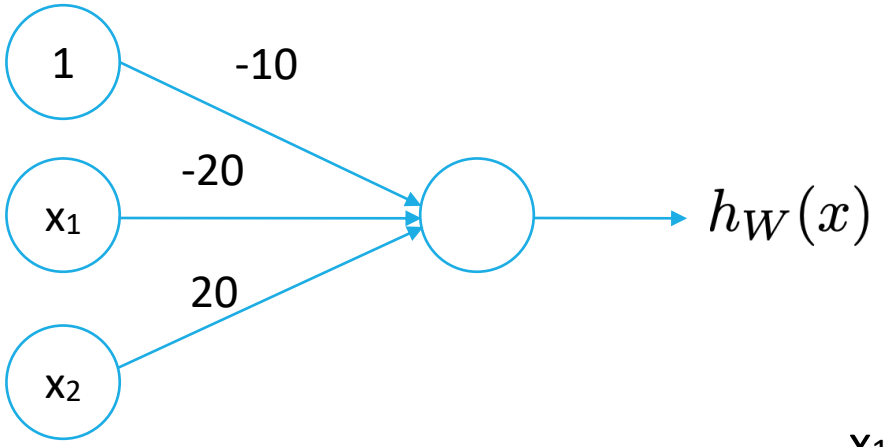


$$h_W(x) = g(10 - 20x_1)$$

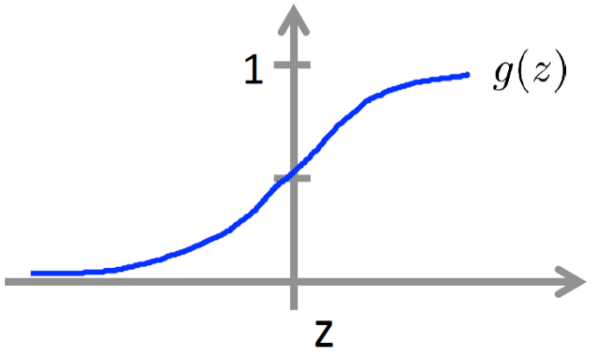


x_1	$h_W(x)$
0	1
1	0

Hàm AND(NOT(x1), x2)

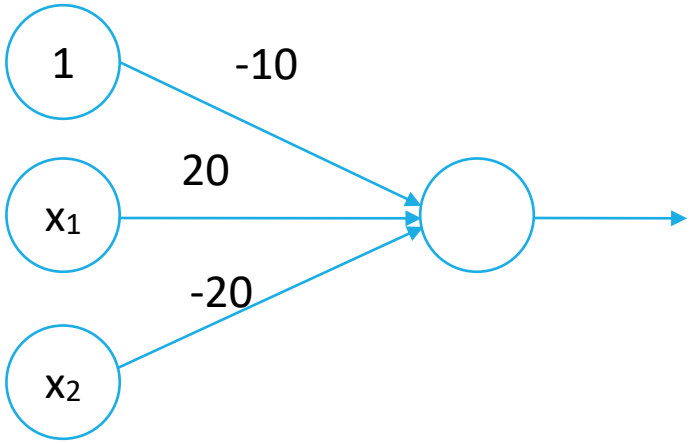


$$h_W(x) = g(-10 - 20x_1 + 20x_2)$$



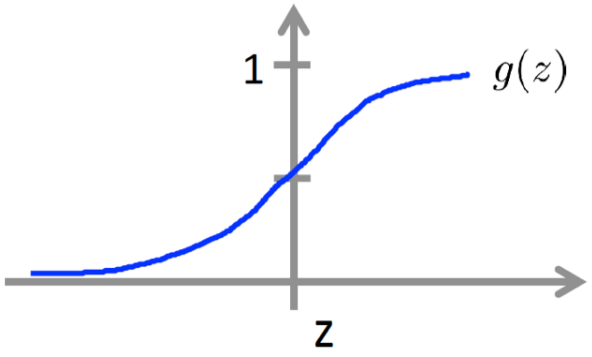
x_1	x_2	$h_W(x)$
0	0	0
0	1	1
1	0	0
1	1	0

Hàm AND(x1, NOT(x2))



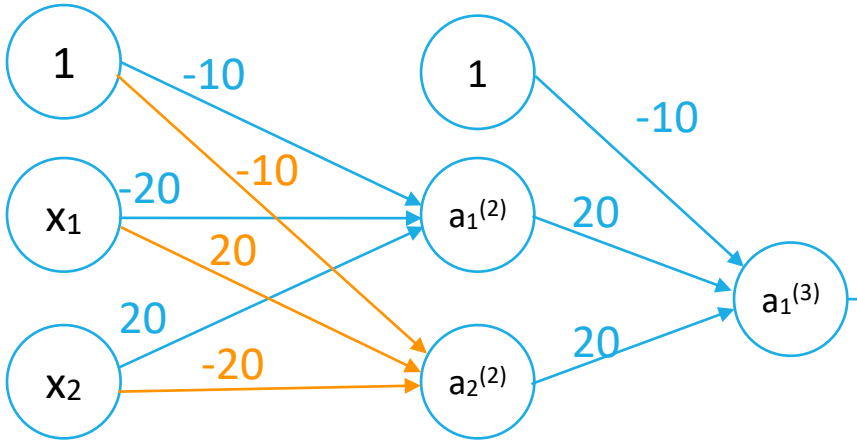
$h_W(x)$

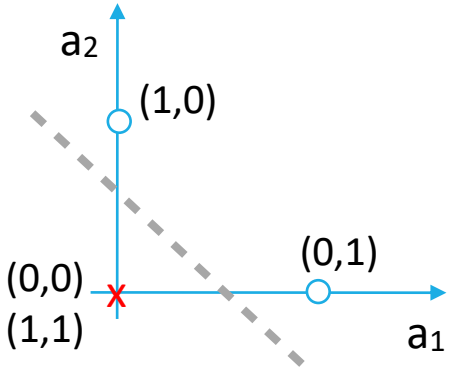
$$h_W(x) = g(-10 + 20x_1 - 20x_2)$$



x_1	x_2	$h_W(x)$
0	0	0
0	1	0
1	0	1
1	1	0

Hàm XOR





x_1	x_2	$h_W(x)$
0	0	0
0	1	1
1	0	1
1	1	0

XOR(x_1, x_2) = OR(AND(NOT(x_1), x_2), AND(x_1, NOT(x_2)))

Bài tập với playground

Playground: https://playground.tensorflow.org

In this exercise, we will train our first little neural net. Neural nets will give us a way to learn nonlinear models without the use of explicit feature crosses.

Task 1: The model as given combines our two input features into a single neuron. Will this model learn any nonlinearities? Run it to confirm your guess.

Task 2: Try increasing the number of neurons in the hidden layer from 1 to 2, and also try changing from a Linear activation to a nonlinear activation like ReLU. Can you create a model that can learn nonlinearities? Can it model the data effectively?

Task 3: Try increasing the number of neurons in the hidden layer from 2 to 3, using a nonlinear activation like ReLU. Can it model the data effectively? How does model quality vary from run to run?

Task 4: Continue experimenting by adding or removing hidden layers and neurons per layer. Also feel free to change learning rates, regularization, and other learning settings. What is the smallest number of neurons and layers you can use that gives test loss of 0.177 or lower?

Does increasing the model size improve the fit, or how quickly it converges? Does this change how often it converges to a good model? For example, try the following architecture:

- First hidden layer with 3 neurons.
- Second hidden layer with 3 neurons.
- Third hidden layer with 2 neurons.

Source: Google Crash Course

Huấn luyện mạng và thuật toán lan truyền ngược

Phần 2

Hàm chi phí

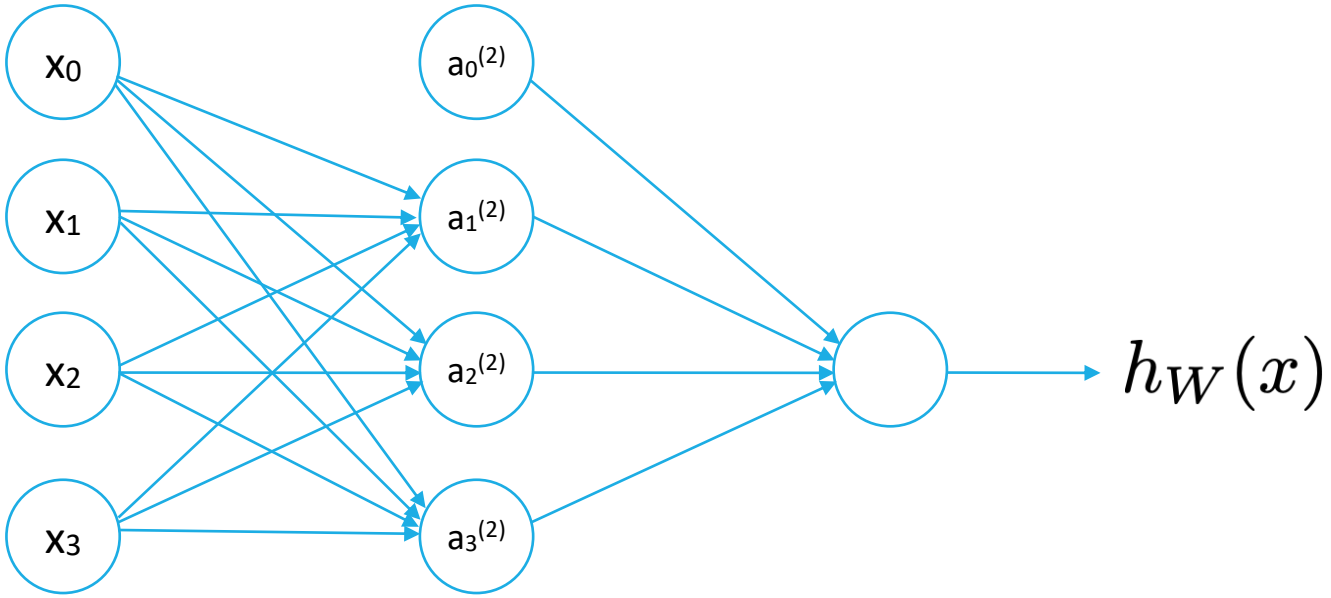
$$J(W) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log(h_W(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_W(x^{(i)}))\right]$$

Gradient

$$\frac{dJ}{dW} = \left[\frac{dJ}{dW_{10}^{(1)}}, \frac{dJ}{dW_{11}^{(1)}}, \ldots, \frac{dJ}{dW_{10}^{(L-1)}}, \ldots, \frac{dJ}{dW_{s_{l+1}s_l}^{(L-1)}} \right]$$

Đạo hàm hàm hợp

$$\frac{dJ}{dW_{ij}^{(l)}} = \sum_{k=1}^{s_{(l+1)}} \frac{dJ}{dz_k^{(l+1)}} \frac{dz_k^{(l+1)}}{dW_{ij}^{(l)}}$$



$$\frac{dJ}{dz_i^{(l)}} = \sum_{j=1}^{s_{(l+1)}} \frac{dJ}{dz_j^{(l+1)}} \frac{dz_j^{(l+1)}}{dz_i^{(l)}}$$

$$= \sum_{j=1}^{s_{(l+1)}} \delta_j^{(l+1)} \frac{dz_j^{(l+1)}}{dz_i^{(l)}}$$

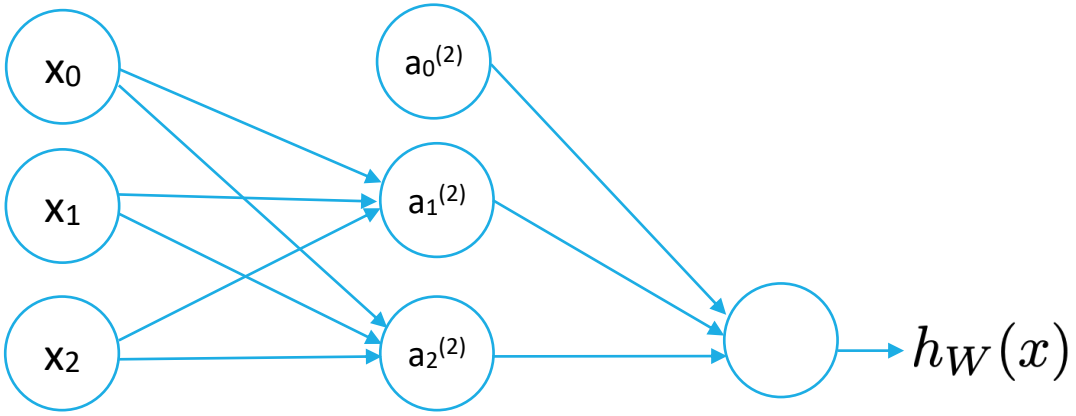
$$= \sum_{j=1}^{s_{(l+1)}} \delta_j^{(l+1)} \frac{d}{dz_i^{(l)}} \sum_{k=0}^{s_l} W_{jk}^{(l)} g(z_k^{(l)})$$

$$= \sum_{j=1}^{s_{(l+1)}} \delta_j^{(l+1)} W_{ji}^{(l)} g'(z_i^{(l)})$$

$$= g'(z_i^{(l)}) \sum_{j=1}^{s_{(l+1)}} (W^{(l)})_{ij}^T \delta_j^{(l+1)}$$

Định nghĩa

$$\delta_j^{(l)} = \frac{dJ}{dz_j^{(l)}}$$



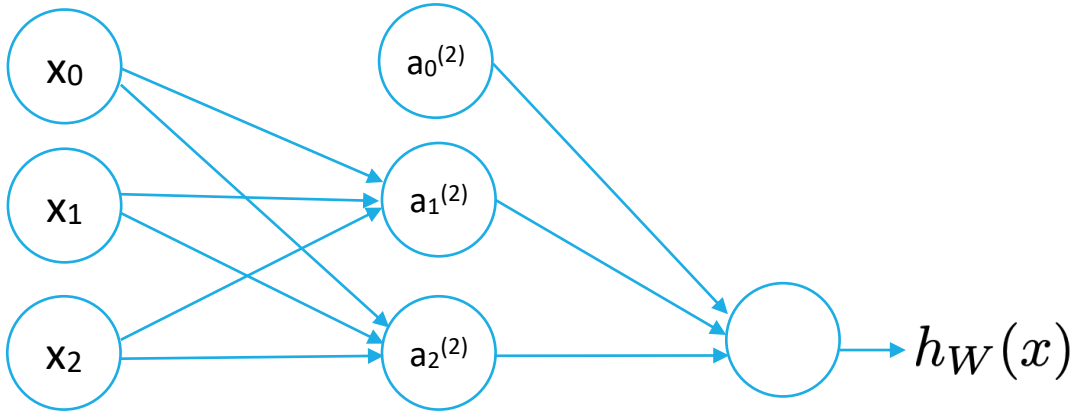
$$\frac{dJ}{dW_{ij}^{(l)}} = \sum_{k=1}^{s_{(l+1)}} \frac{dJ}{dz_k^{(l+1)}} \frac{dz_k^{(l+1)}}{dW_{ij}^{(l)}}$$

$$= \sum_{k=1}^{s_{(l+1)}} \delta_k^{(l+1)} \frac{d}{dW_{ij}^{(l)}} \sum_{t=0}^{s_l} W_{kt}^{(l)} a_t^{(l)}$$

$$= \delta_i^{(l+1)} \frac{d}{dW_{ij}^{(l)}} W_{ij}^{(l)} a_j^{(l)}$$

$$= \delta_i^{(l+1)} a_j^{(l)}$$

$$\frac{dJ}{dW^{(l)}} = \delta^{(l+1)} a^{(l)T}$$



Thuật toán lan truyền ngược

1. Áp dụng lan truyền tới để tính:

   $z^{(1)}$, …, $z^{(L)}$, $a^{(1)}$, …, $a^{(L)}$ và $J(z^{(L)})$

2. $\delta^{(L)} = \frac{dJ}{dz^{(L)}}$

3. for l = L-1 to 1

4. $\frac{dJ}{dz^{(l)}} = g'(z^{(l)})(W^{(l)T}\delta^{(l+1)})$

5. $\frac{dJ}{dW^{(l)}} = \delta^{(l+1)}a^{(l)T}$

6. end for

Regularization

- Hàm chi phí

$$J(W) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log(h_W(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_W(x^{(i)}))\right] + \frac{\lambda}{2m} \sum_{l=1}^{L-1} \sum_{j=1}^{s_{l+1}} \sum_{i=1}^{s_l} (W_{ji}^{(l)})^2$$

Regularization

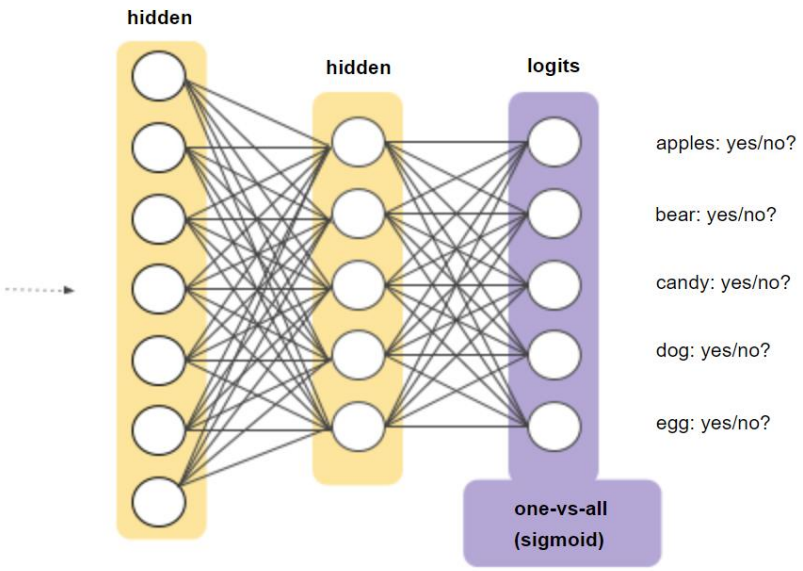
- Gradient

$$\frac{dJ}{dz^{(l)}} = g'(z^{(l)})(W^{(l)T}\delta^{(l+1)}) + \lambda W^{(l)} \qquad \text{if} \quad j \neq 0$$

$$\frac{dJ}{dz^{(l)}} = g'(z^{(l)})(W^{(l)T}\delta^{(l+1)}) \qquad \text{if} \quad j = 0$$

Mạng nơ ron cho phân lớp đa lớp

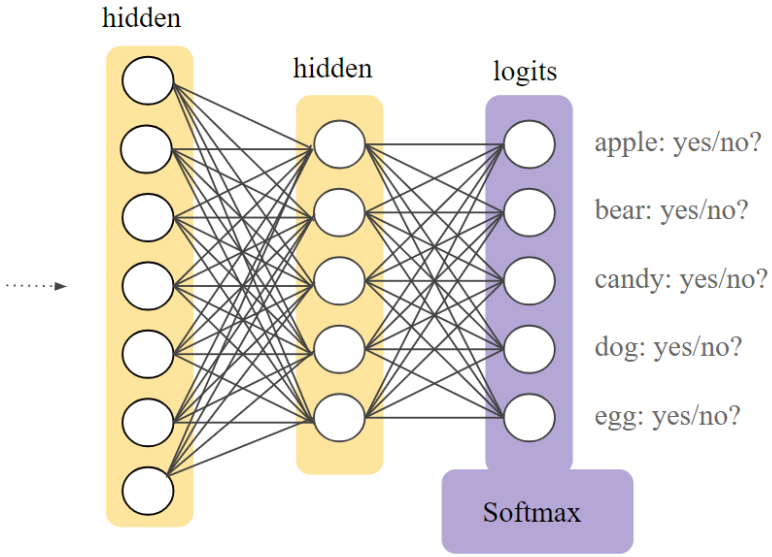
- One vs. All
  - Huấn luyện K bộ phân lớp



Source: Google Crash Course

Mạng nơ ron cho phân lớp đa lớp

- Softmax



$$P(y = j \mid \mathbf{x}) = \frac{e^{\mathbf{x}^\mathsf{T}\mathbf{w}_j}}{\sum_{k=1}^{K} e^{\mathbf{x}^\mathsf{T}\mathbf{w}_k}}$$

Source: Google Crash Course

Mạng nơ ron cho phân lớp đa lớp

- Softmax
  - Cross Entropy loss

$$-\sum_{c=1}^{K} y_c \log(p_c)$$

Thực hành

- Dự đoán giá nhà với California Housing Dataset
  - Intro to Neural Nets.ipynb
- Nhận dạng ký số viết tay với mạng nơ ron
  - Multi-class classification with MNIST.ipynb