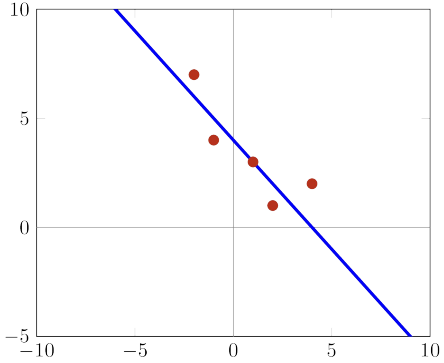
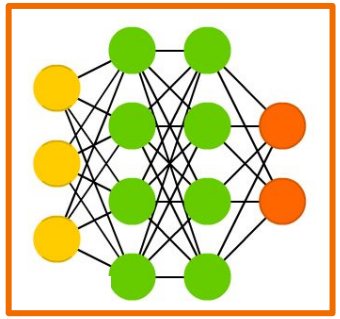
Deep Neural Network

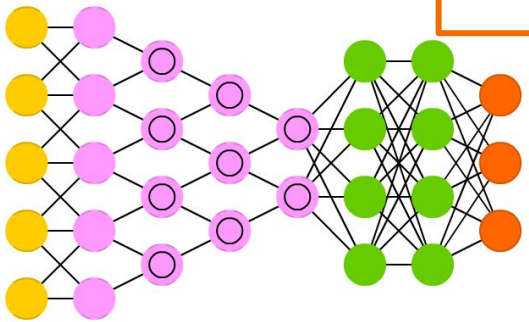
Trại Hè Toán và Khoa học MaSSP
Môn Tin học
Hà Nội, 06/2017

Tiến trình học

- Linear Regression
- Logistic Regression
- Deep Neural Network
- Convolutional Neural Network



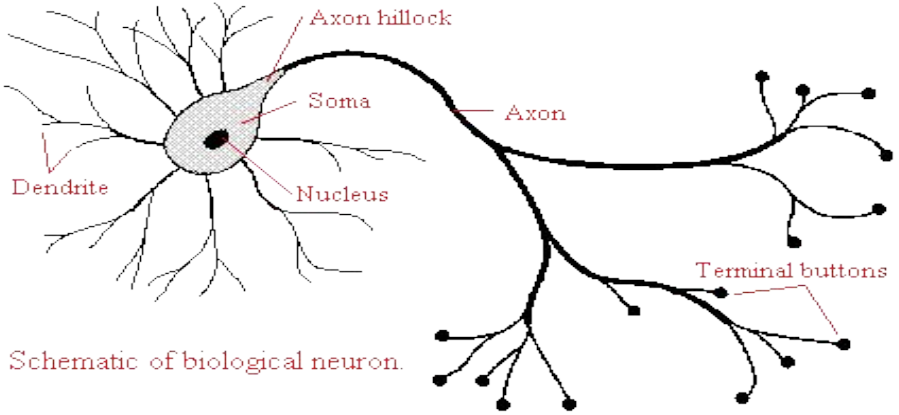


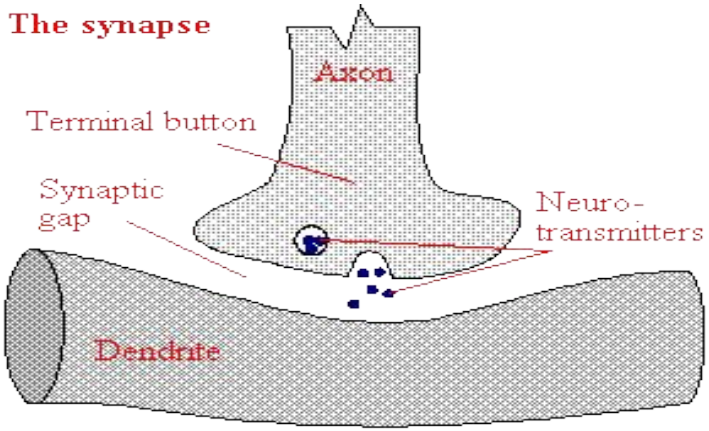


Giới thiệu Neural Network

- Khái niệm về NN có từ những năm đầu 1940s
- Cuối 1980s, NN trở nên phổ biến hơn nhờ phát triển vượt bậc của phần cứng
- Lấy cảm hứng từ mạng lưới thần kinh trong sinh học, mô phỏng lại bộ não

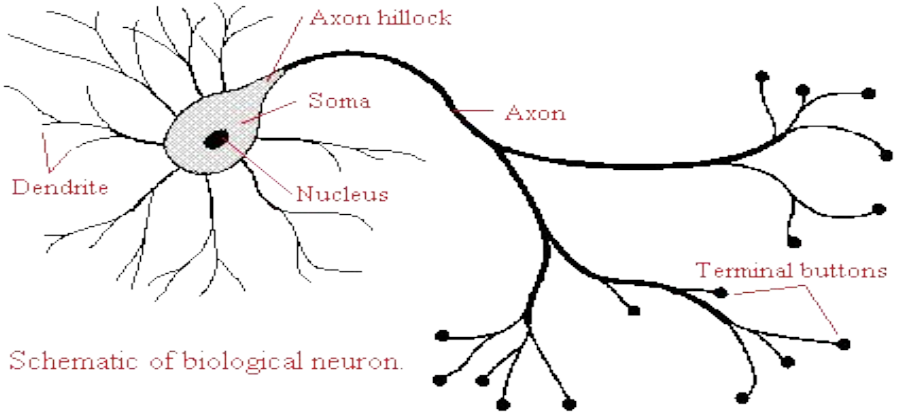
Neuron - Tế bào thần kinh

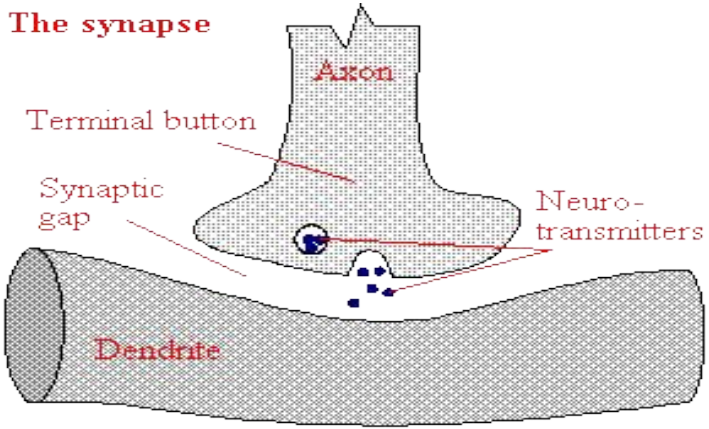




- Bộ não là một mạng lưới, chứa khoảng **10 tỷ** neuron
- Neuron sử dụng các **phản ứng sinh học** để nhận, xử lý, và truyền tín hiệu

Neuron - Tế bào thần kinh



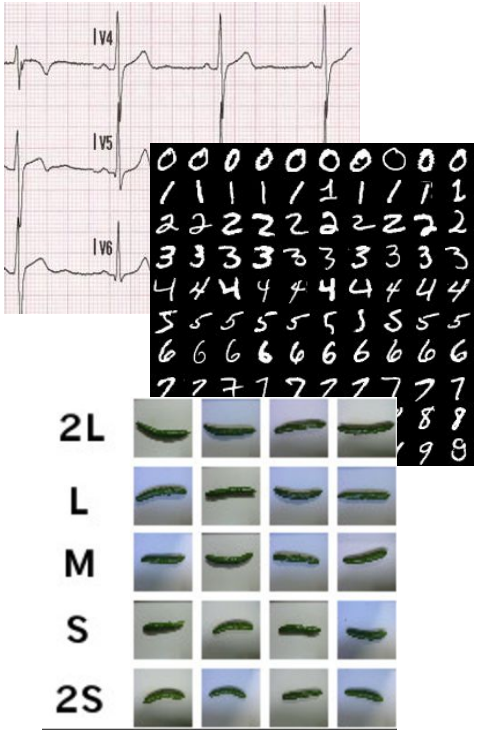


- Dendrite của mỗi neuron có thể kết nối tới hàng nghìn neuron lân cận

Giới thiệu Neural Network

NN có rất nhiều ứng dụng

- Phân loại
  - Nông nghiệp: phân loại nông sản
  - Y học: chuẩn đoán bệnh từ tín hiệu điện tâm đồ
- Nhận diện
  - Nhận diện chữ viết, giọng nói, hình ảnh
- Dự đoán
  - Tài chính: dự đoán giá cổ phiếu
  - Khí tượng dự báo thời tiết



Outline

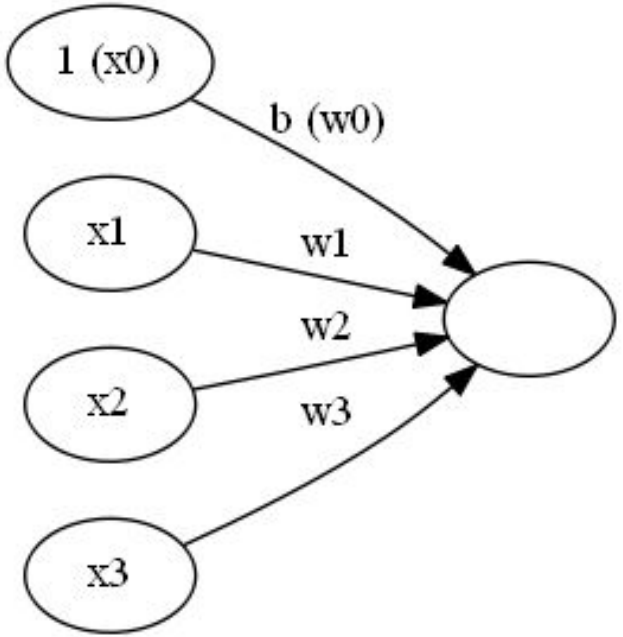
- Đơn vị đơn giản nhất của Neural Networks
  - Perceptron
  - Sigmoid Neuron
- Cấu trúc của Neural Networks
  - Lớp Input
  - Lớp Output
  - Lớp ẩn (Hidden)
- Thuật toán backpropagation

Perceptron

Một perceptron bao gồm:

- Inputs **x**'s (Binary hoặc số thực)
- Trọng số weights **w**'s (các số thực)
  - $\mathbf{w_i}$ tỉ lệ với độ quan trọng của $\mathbf{x_i}$ với output
- Thiên lệch bias **b** (số thực), hay trọng số $\mathbf{w_0}$ với input $\mathbf{x_0 = 1}$

...

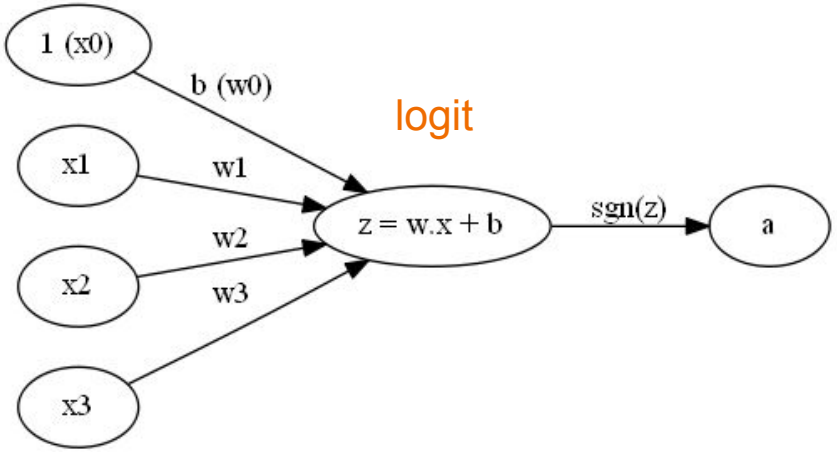


Perceptron

Một perceptron bao gồm:

- Inputs **x**'s (Binary hoặc số thực)
- Trọng số weights **w**'s
- Thiên lệch bias **b** (số thực)
- Binary output **a = sign(wx + b)**

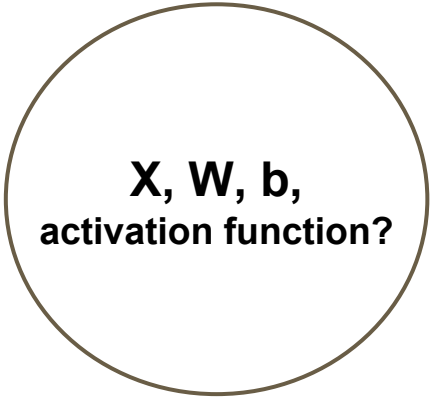
Sign là 1 loại activation function



$$\text{output} = \begin{cases} 0 & \text{if } w \cdot x + b \leq 0 \\ 1 & \text{if } w \cdot x + b > 0 \end{cases}$$

http://neuralnetworksanddeeplearning.com/chap1.html

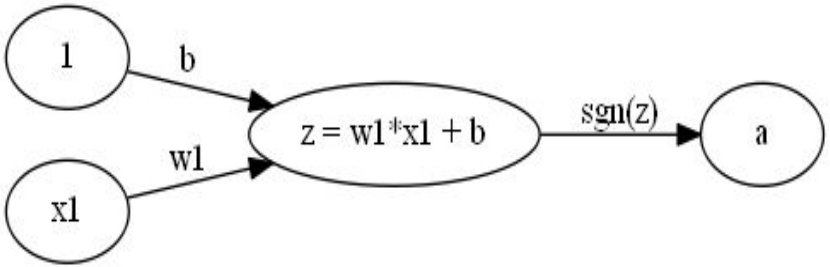
Perceptron - Ví dụ



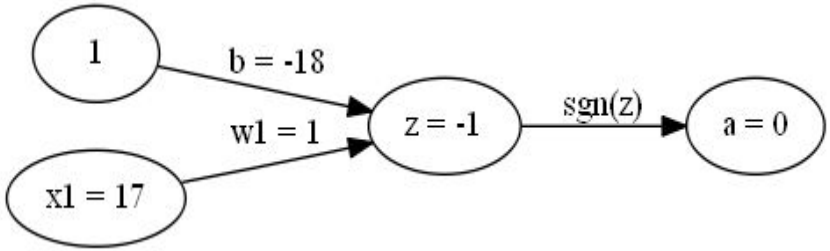
*“Người đủ **18** tuổi trở lên được lái xe mô tô hai bánh, xe mô tô ba bánh có dung tích xi-lanh từ 50 cm3 trở lên và các loại xe có kết cấu tương tự”*

Xây dựng một perceptron để cho output là **1** khi đủ tuổi, **0** khi chưa đủ tuổi lái xe?

Perceptron - Ví dụ



*"Người đủ **18** tuổi trở lên được lái xe mô tô hai bánh, xe mô tô ba bánh có dung tích xi-lanh từ 50 cm3 trở lên và các loại xe có kết cấu tương tự"*

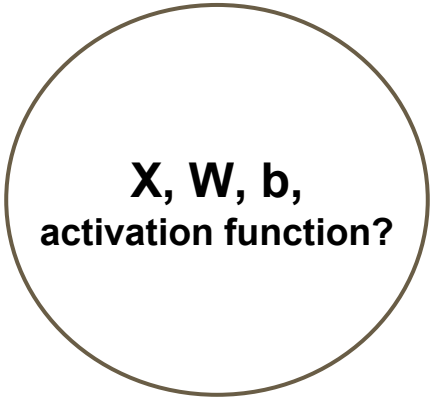


- Input: **x1**, chỉ số tuổi
- Trọng số **w1 = 1**
- Bias **b = -18**
- Activation function, tương tự hàm sign

Perceptron - Ví dụ

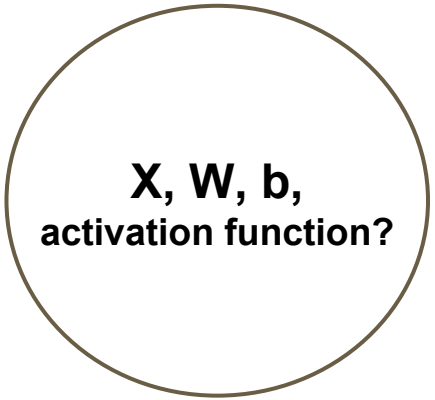
Perceptron - Ví dụ

- Tổng kích thước 3 chiều (dài, rộng, cao) không quá **203** cm



Perceptron - Ví dụ

- Tổng kích thước 3 chiều (dài, rộng, cao) không quá **203** cm

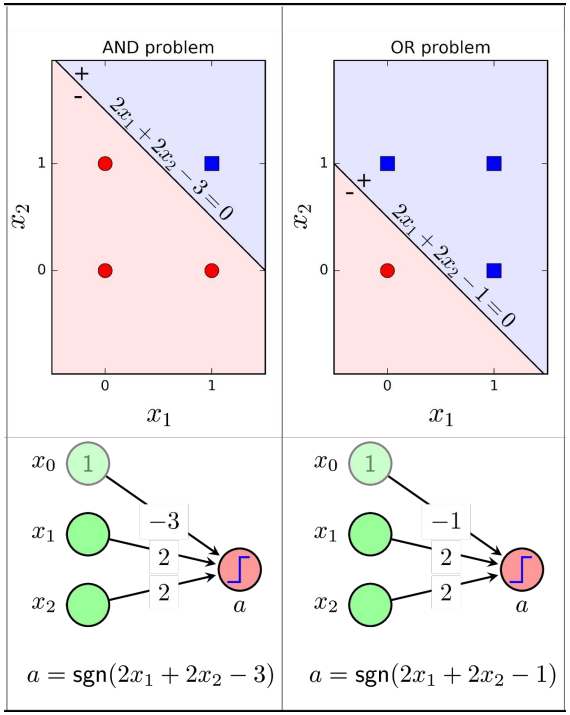


- Input: [**x1, x2, x3**], chỉ độ dài, rộng cao
- Trọng số **w = [-1, -1, -1]**
- Bias **b = 204**
- Activation function, tương tự hàm sign
  - Trả **1** khi **z > 0**
  - Trả **0** khi **z <= 0**

Perceptron - Ví dụ

- Nếu input **x** gồm có 2 thuộc tính, mỗi thuộc tính có giá trị là 0 hoặc 1 (binary)
  - Liệt kê tất cả các giá trị có thể của **x**?
  - **w** có kích thước như thế nào?

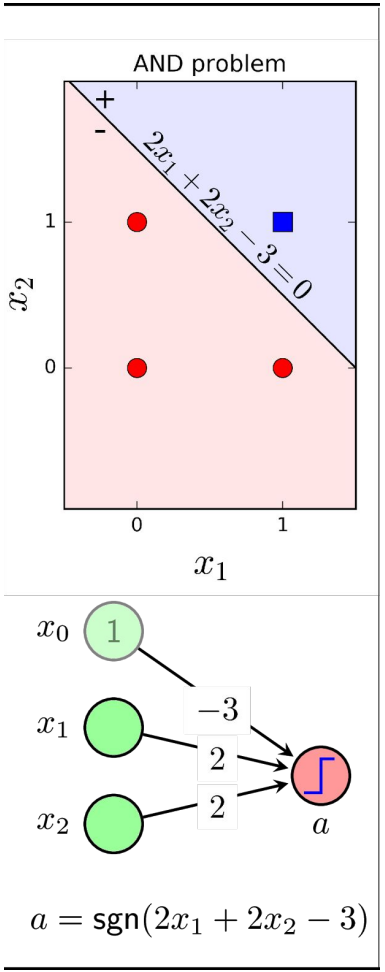
Perceptron - Ví dụ



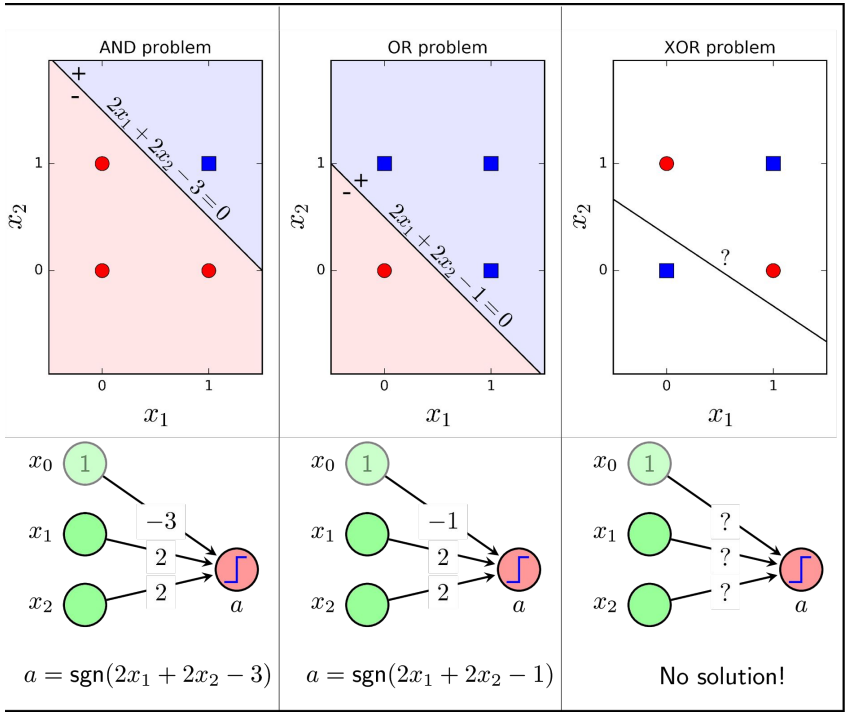
http://machinelearningcoban.com/2017/02/24/mlp/

Perceptron - Câu hỏi

Nếu ta lấy tất cả trọng số **weights** và thiên lệch **bias** nhân với một hằng số **c > 0**, thì perceptron có còn cho kết quả đúng không?



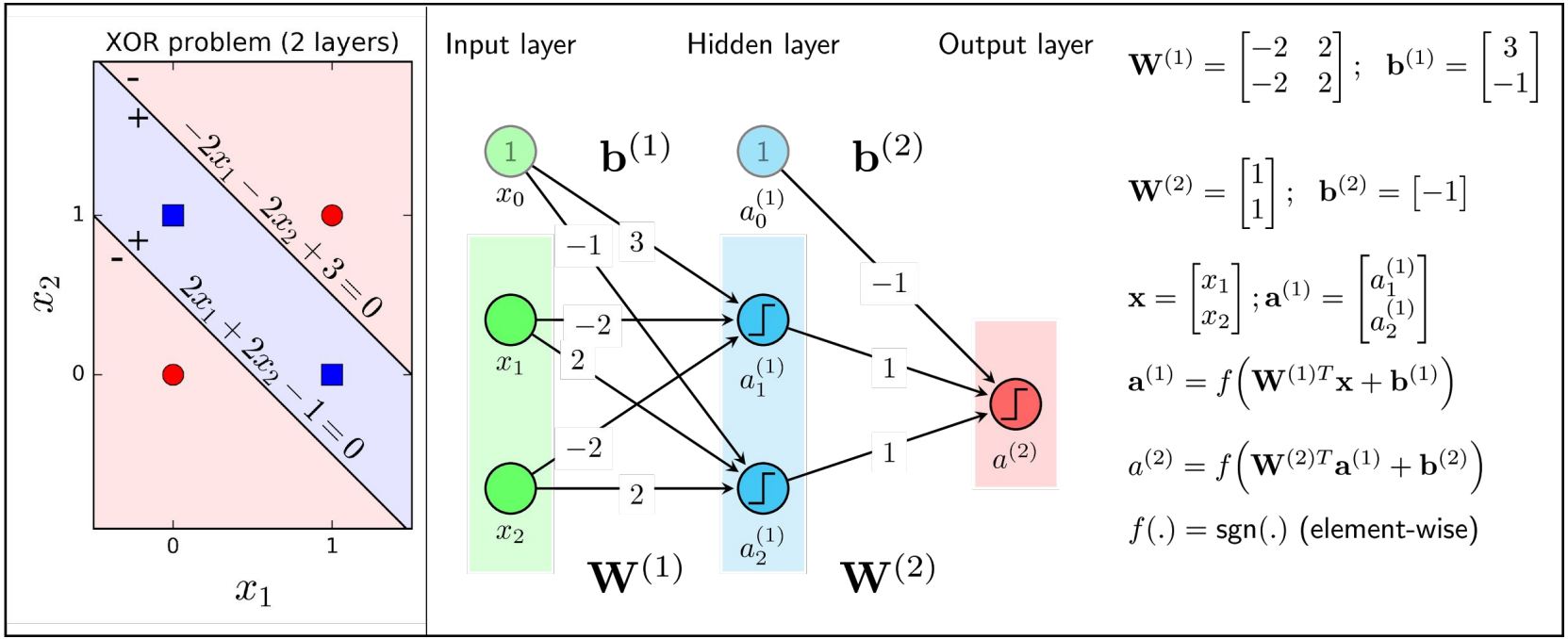
Perceptron - Bài toán XOR



Not linearly separable!

http://machinelearningcoban.com/2017/02/24/mlp/

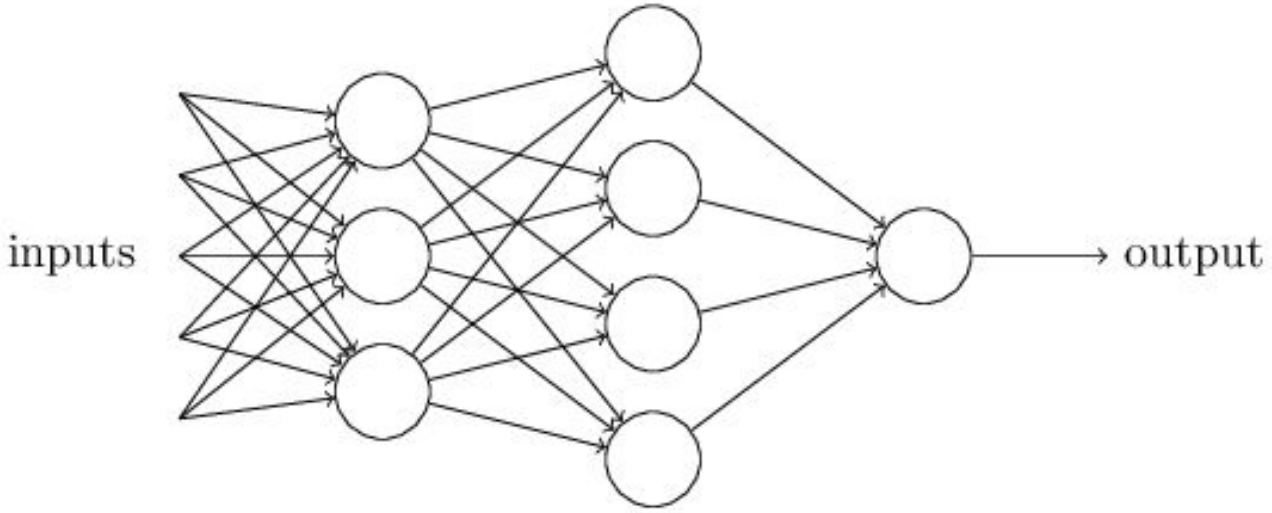
Dùng thêm 2 perceptrons!



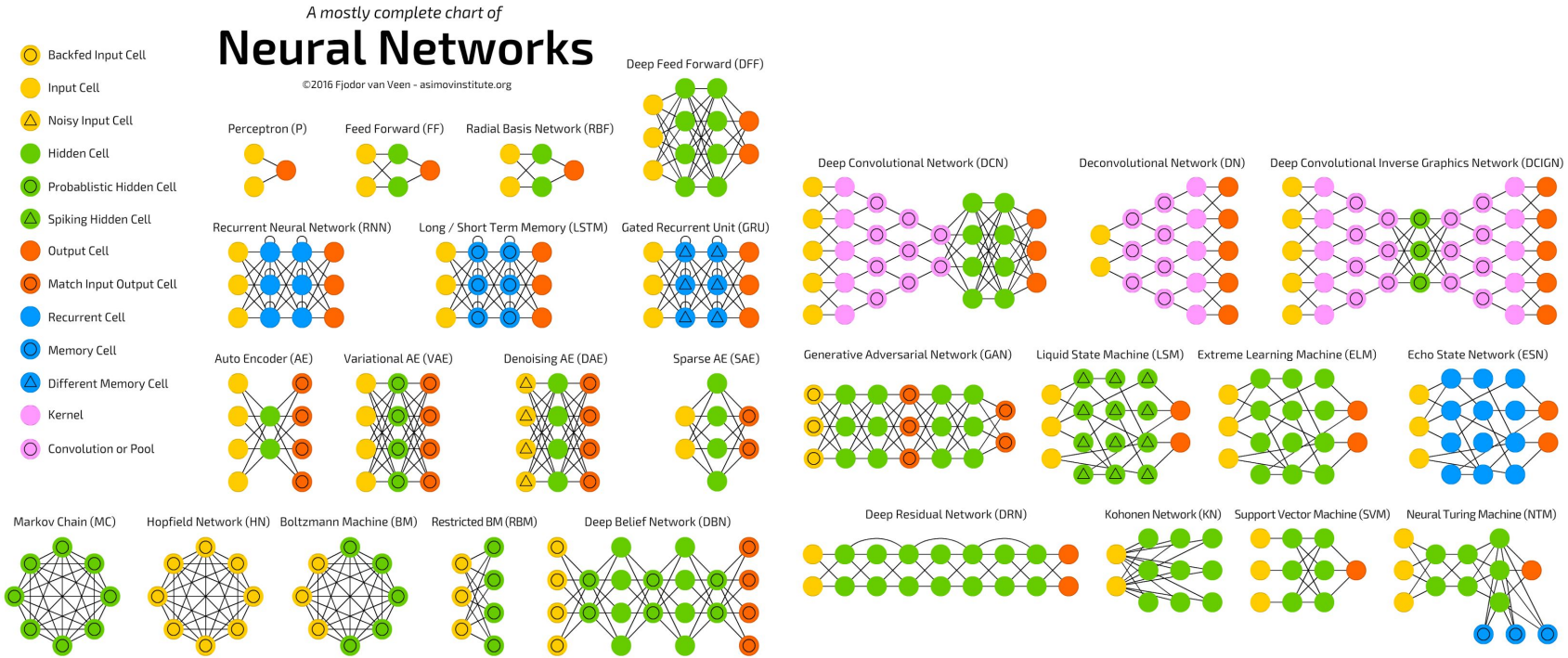
http://machinelearningcoban.com/2017/02/24/mlp/

Perceptron - Multilayer Perceptron

Ta có thể kết nối các Perceptron thành một **network** nhiều lớp, sao cho output của lớp này trở thành input của lớp sau



http://neuralnetworksanddeeplearning.com/chap1.html

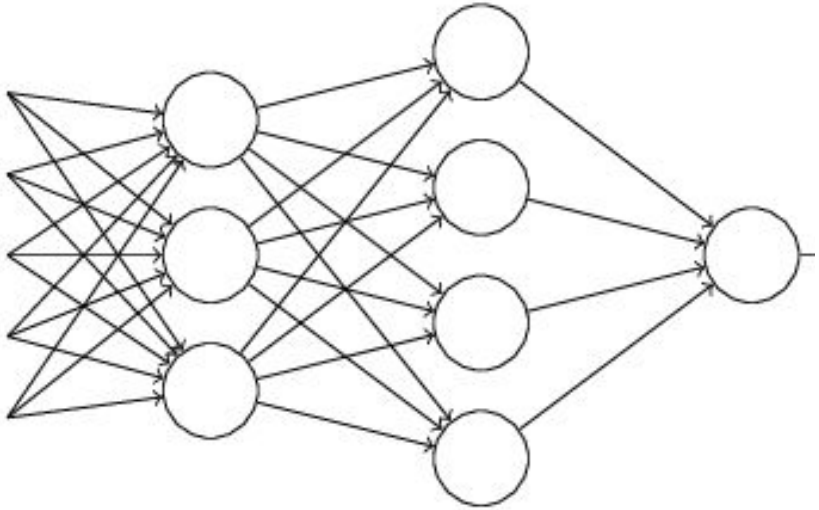


Michelle Fullwood, A gentle introduction to Deep Learning with Tensorflow, PyCon 2017

Perceptron - Câu hỏi

Giả sử tất cả các perceptron trong network này chỉ lấy giá trị **input là binary** để phù hợp với output của chúng cũng là binary.

Nếu ta lấy tất cả trọng số **weights** và thiên lệch **bias** nhân với một hằng số **c > 0**, thì network có cho kết quả output cuối cùng khác đi không?



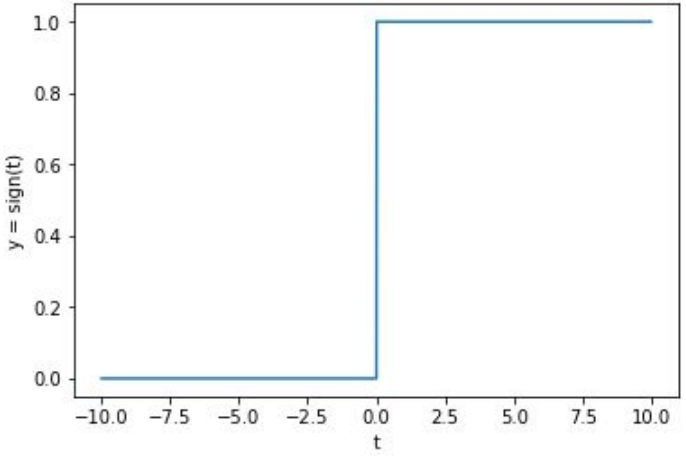
Questions?

Sigmoid Neuron

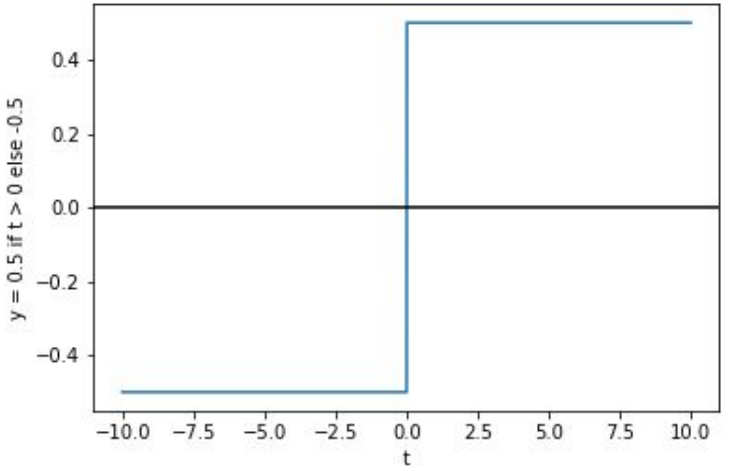
- Tương tự như perceptron, khác output và activation function

	Perceptron	**Sigmoid Neuron**
Output	0 hoặc 1	Giá trị giữa 0 và 1 $a = \sigma(w \cdot x + b)$
Loại activation function	Step (ví dụ sign)	Sigmoid (ví dụ sigmoid, tanh)

Step Functions

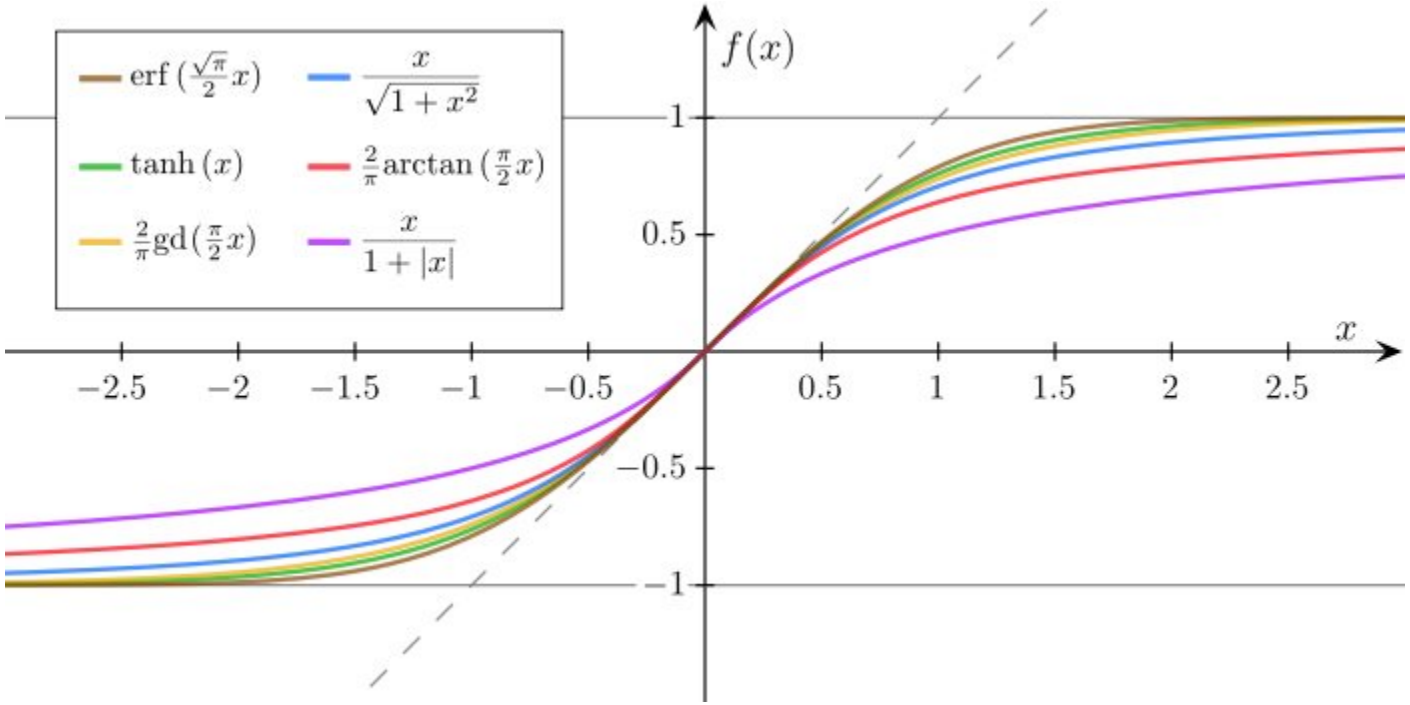


$$y = \begin{cases} 1, & \text{if } t > 0 \\ 0, & \text{otherwise} \end{cases}$$

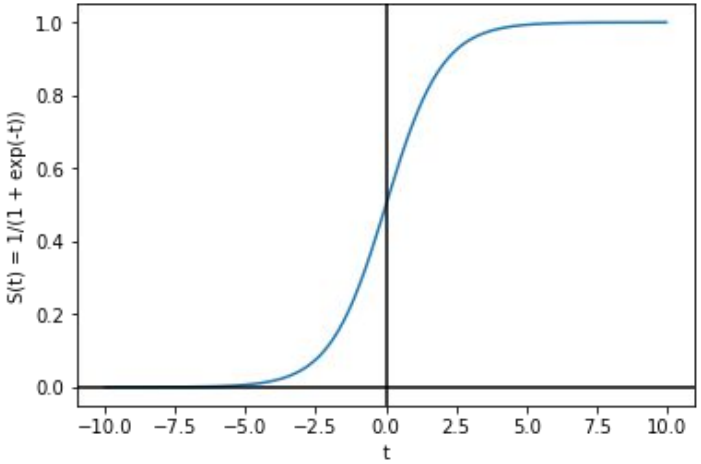


$$y = \begin{cases} 0.5, & \text{if } t > 0 \\ -0.5, & \text{otherwise} \end{cases}$$

Sigmoid Functions

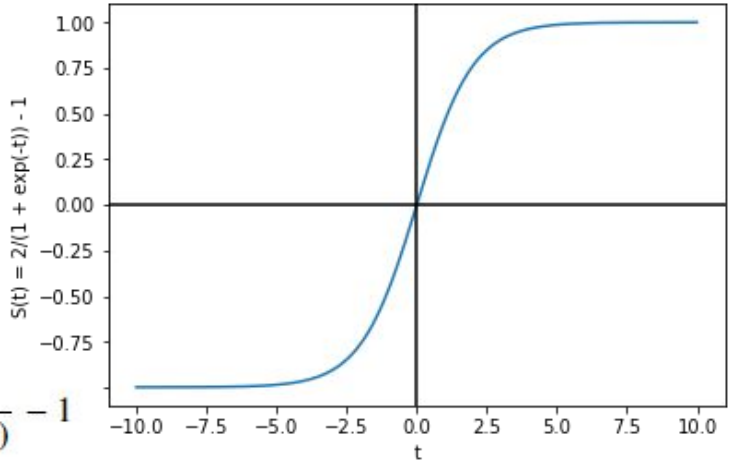


Sigmoid Functions



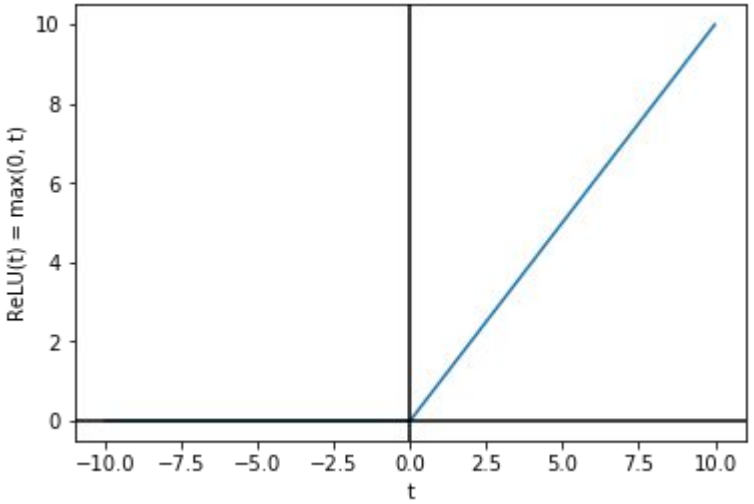
$$S(t) = \frac{1}{1 + exp(-t)}$$

$$tanh(t) = \frac{2}{1 + exp(-t)} - 1$$



So với perceptron, thay đổi **nhỏ** trong **weights** và **bias** của sigmoid neuron sẽ chỉ tạo một thay đổi **nhỏ** cho output

ReLU

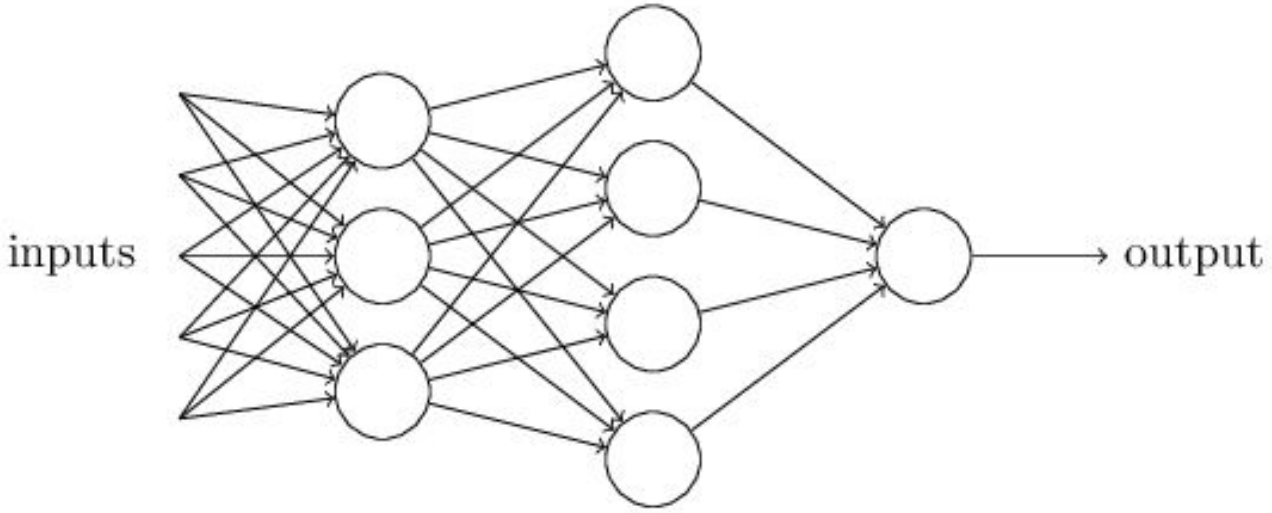


$$ReLU(t) = max(0, t)$$

- Ngoài step và sigmoid, ReLU cũng là một activation function
  - phổ biến trong NN
  - Đơn giản, tính đạo hàm nhanh (cần cho thuật toán Gradient Descent)

Sigmoid Neuron - Network

Tương tự như với perceptron, ta có thể kết nối các sigmoid neuron thành một network



http://neuralnetworksanddeeplearning.com/chap1.html

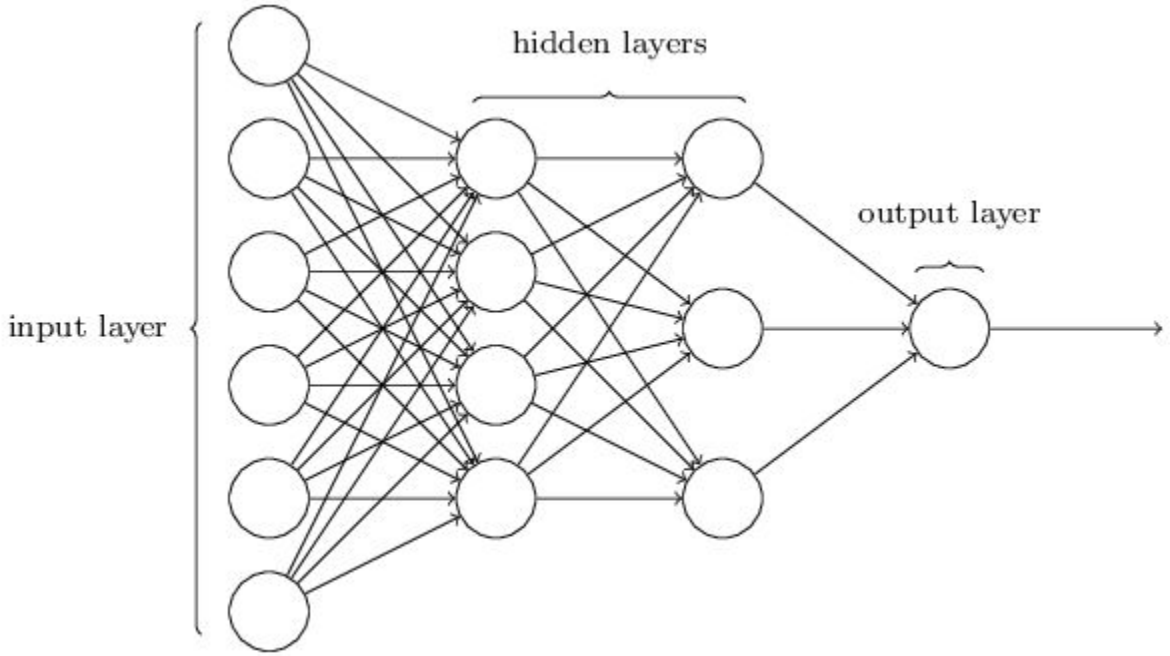
Sigmoid Neuron vs Perceptron Network

Khác với perceptron, output của sigmoid neuron không phải là binary mà là **liên tục**.

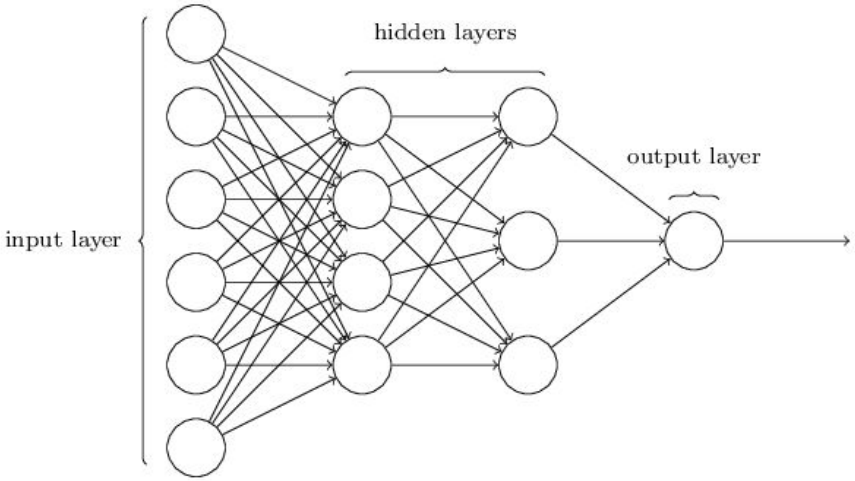
- Nếu ta lấy tất cả trọng số **w** và thiên lệch **b** nhân với một hằng số **c > 0**, thì mạng lưới của các *sigmoid neuron* có cho kết quả output khác đi không?
- Chứng minh rằng khi **c**$\rightarrow\infty$, thì network của *sigmoid neurons* sẽ giống như của *perceptrons*

Questions?

Cấu trúc của Neural Network

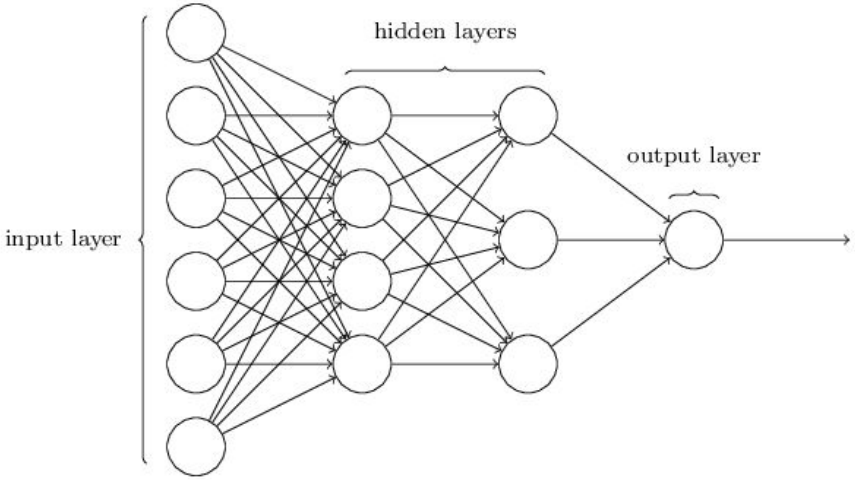


Cấu trúc của Neural Network



- Mỗi node = 1 neuron
- Mỗi cạnh = trọng số từ 1 neuron đến 1 neuron khác ở **lớp tiếp theo**
- Mỗi neuron trong một lớp được nối đến tất cả các neuron trong lớp tiếp theo
- Các neuron trong cùng một lớp **không** nối với nhau

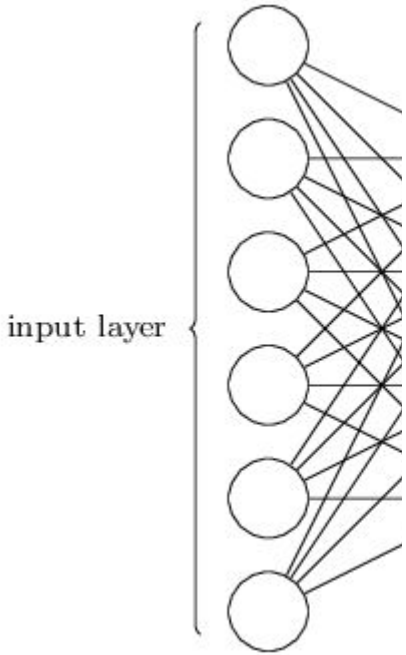
Cấu trúc của Neural Network



Những neural network khác nhau...

- Có số lớp khác nhau
- Có số neuron mỗi lớp khác nhau

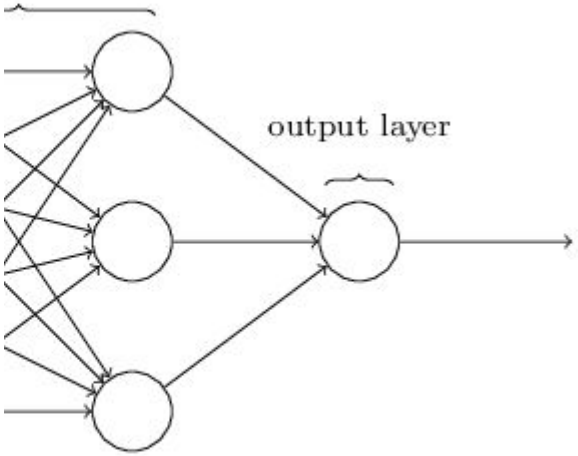
Input Layer



Trong lớp input:

- Mỗi neuron nối với tất cả neuron trong lớp thứ 2
- Ví dụ của lớp input trong
  - Phân loại chữ số viết tay
  - Lọc email rác

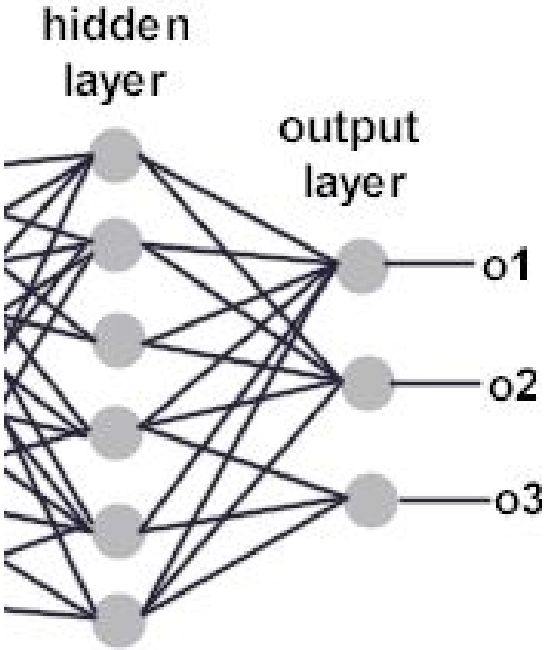
Output Layer



Trong lớp output cũng là lớp cuối cùng

- Có 1 hoặc nhiều output neuron tùy vào bài toán
  - Binary classification - một hình ảnh chữ viết tay có phải của số “9”?
  - Multi-class classification - hình ảnh này của số nào?

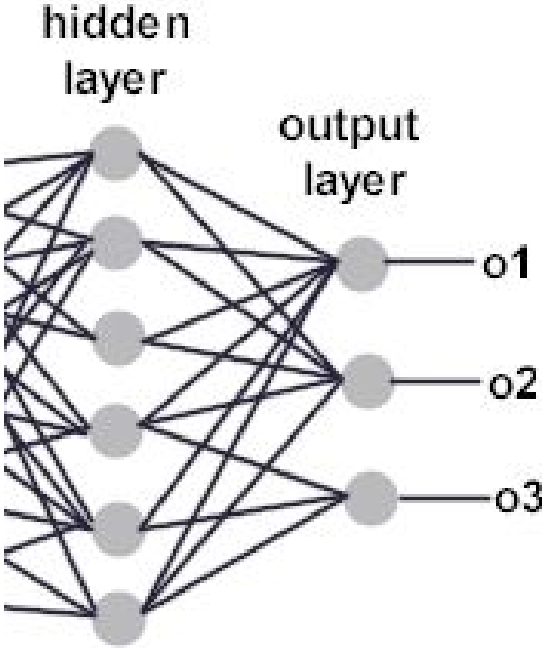
Output Layer - Activation Function



http://dms.irb.hr/tutorial/tut_nnets_short.php

- Lớp output thường dùng hàm **softmax** làm activation function
- Cho giá trị logit **o1**, **o2**, **o3**, hàm **softmax** trả về xác suất mỗi input thuộc về mỗi nhóm:
  - **P(x** thuộc nhóm $\mathbf{y_i}$) với **i = 1, 2, 3**

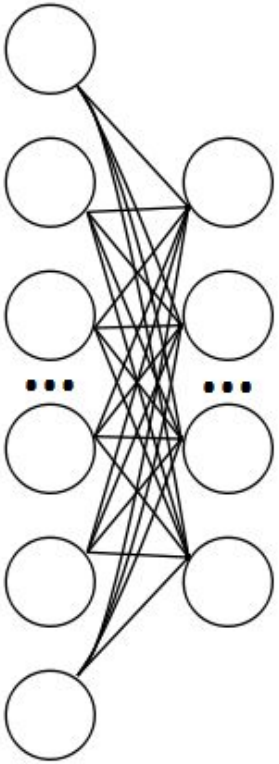
Output Layer - Softmax Regression



$$p(x \text{ belongs to class } y_i) = \frac{exp(o_i)}{exp(o_1) + exp(o_2) + exp(o_3)}$$

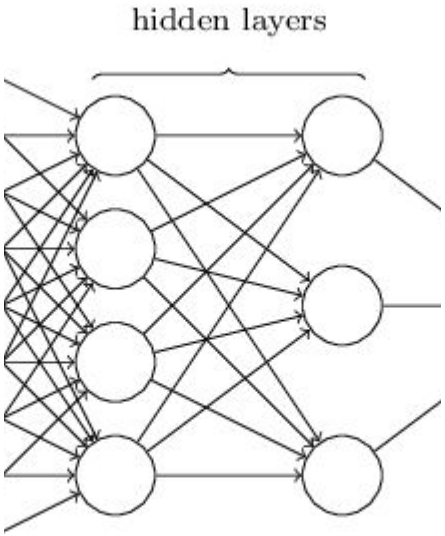
http://dms.irb.hr/tutorial/tut_nnets_short.php

Input layer + Output layer = Logistic Regression!



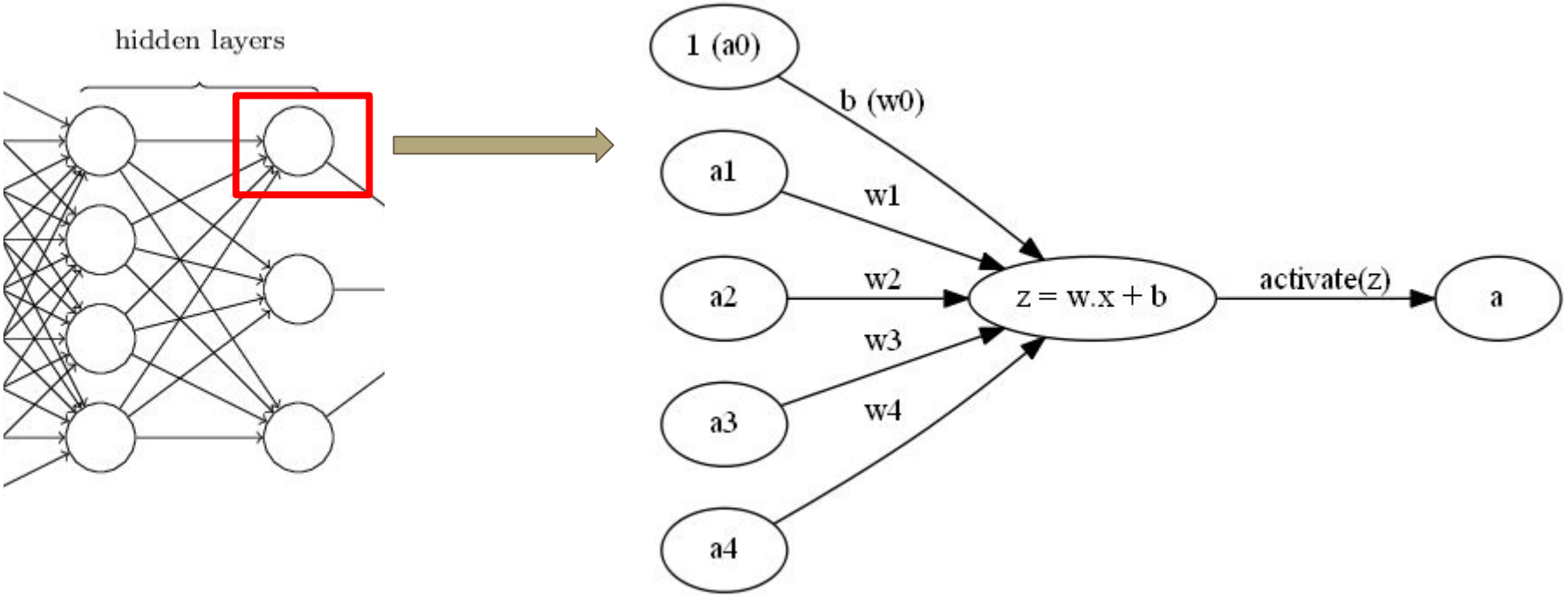
- **Linear Regression** và **Logistic Regression** là những ví dụ đơn giản nhất của Neural Network
  - 0 lớp ẩn
- >= **2** lớp ẩn → **Deep Neural Network**!

Hidden Layers

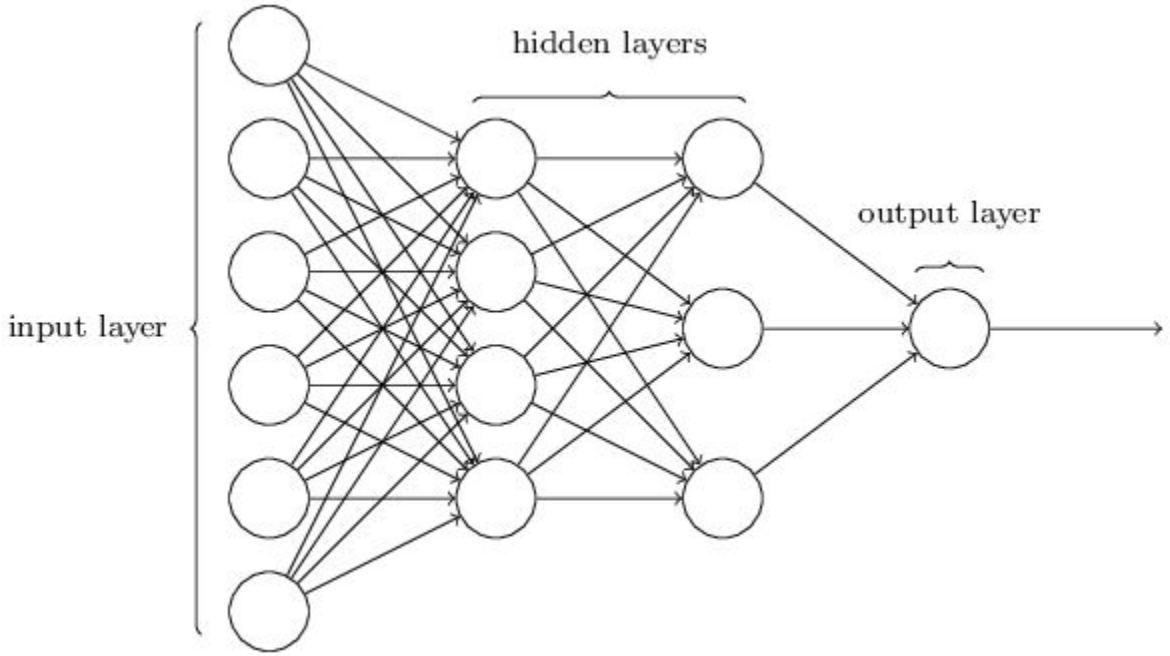


- **Lớp ẩn đầu tiên** có đầu vào là lớp input, đầu ra là lớp tiếp theo
- **Lớp ẩn cuối cùng** có đầu ra là lớp output
- Mỗi lớp ẩn có thể có **số neuron khác nhau**
- Mỗi lớp ẩn có thể sử dụng các activation function khác nhau, nhưng thường chỉ dùng một loại chung cho đơn giản

Hidden Layers



Cấu trúc của Neural Network



So far

- Đơn vị đơn giản nhất của Neural Networks
  - Perceptron
  - Sigmoid Neuron
- Cấu trúc của Neural Networks
  - Lớp Input
  - Lớp Output
  - Lớp ẩn (Hidden)
- Thuật toán backpropagation

Backpropagation algorithm

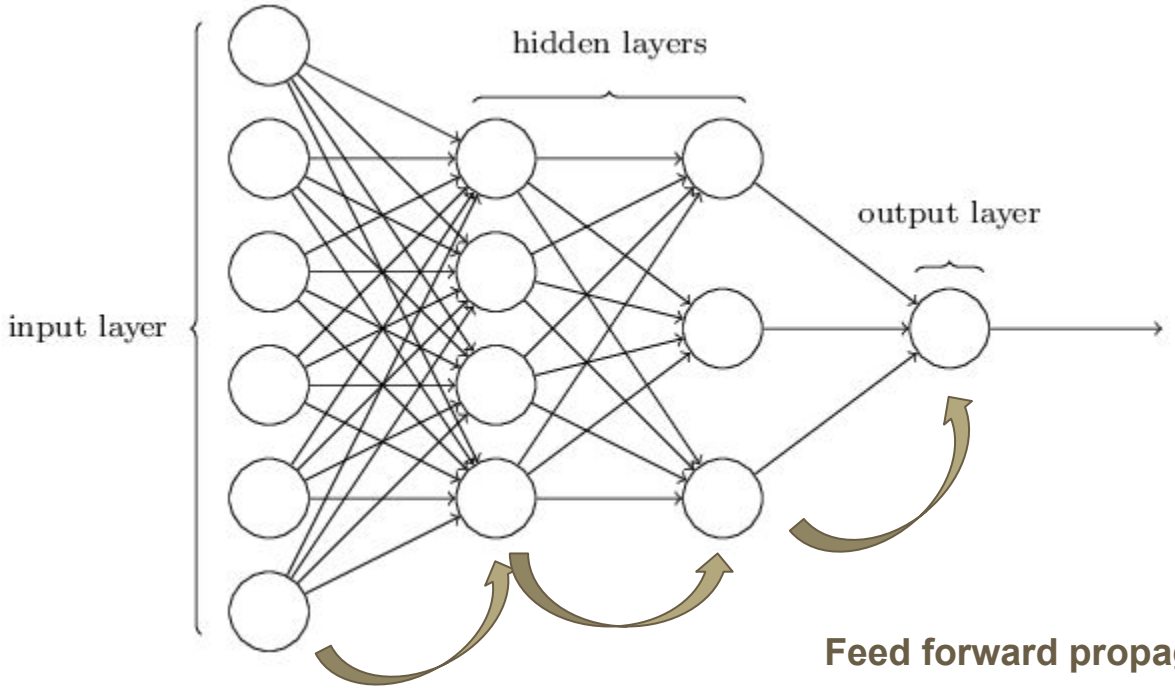
- Thuật toán để điều chỉnh tham số trong network
- Gồm 2 bước
  - **Feed forward**
  - **Back propagate**

Feedforward

Feedforward: tính output của các neuron trong NN từ trái sang phải (input layer đến output layer), thu được *output dự đoán*

=> Ta đã đi qua feedforward

Feed forward



Feed forward propagation direction

Back propagate

Back propagate: tính sai số trong từng lớp từ lớp output đến lớp ẩn đầu tiên, điều chỉnh tham số (weights, bias) cho các lớp này

Backpropagate

- Chiều ngược lại với **feedforward**
- Từ **feedforward**, dự đoán xác suất cho từng nhóm

	Chó	**Mèo**	**Chim**
Ảnh 1	0.5	0.4	0.1
Ảnh 2	0.2	0.6	0.2
Ảnh 3	0.0	0.1	0.9

Backpropagate

- Thực tế *output thật* sẽ lệch với output dự đoán (đặc biệt khi network mới bắt đầu quá trình học!)

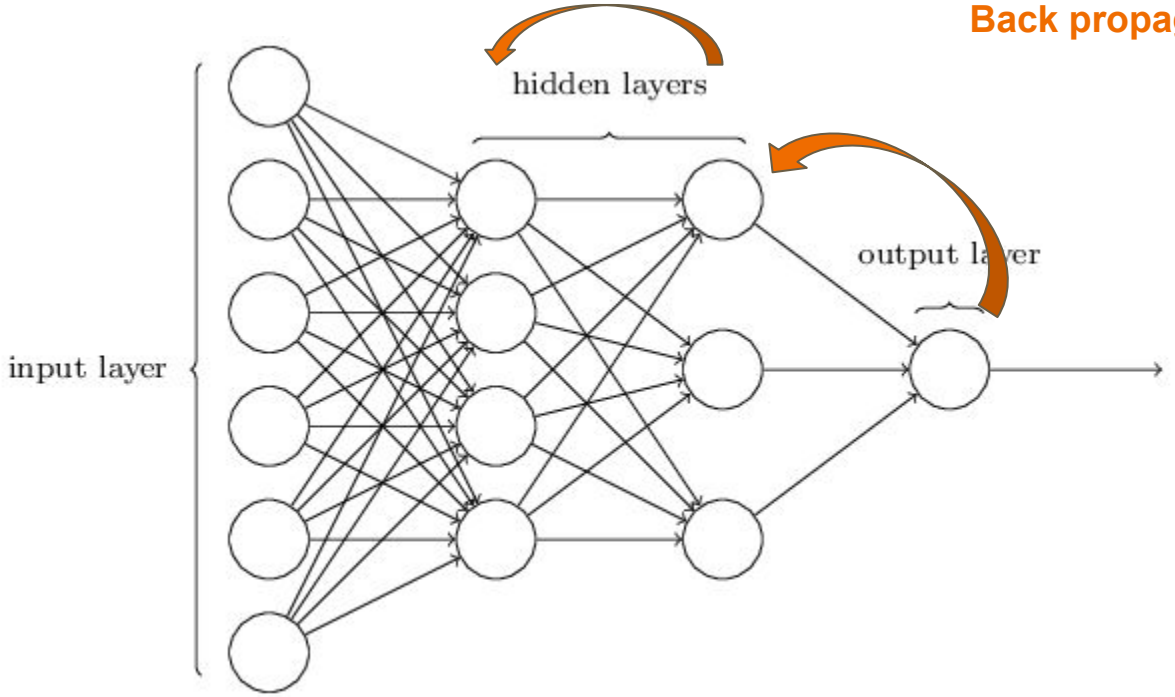
	Chó	Mèo	Chim	Output đúng
Ảnh 1	0.5	0.4	0.1	Mèo
Ảnh 2	0.2	0.6	0.2	Mèo
Ảnh 3	0.0	0.1	0.9	Chim

Backpropagate

Từ sai lệch giữa *output dự đoán* và *output đúng*, ta phải điều chỉnh lại tham số trong network sao cho sai lệch giảm đi

- Dùng cost function **cross-entropy** để tính độ sai lệch
- Dùng thuật toán optimization, ví dụ Gradient Descent để điều chỉnh **W, b** từng lớp

Backpropagate

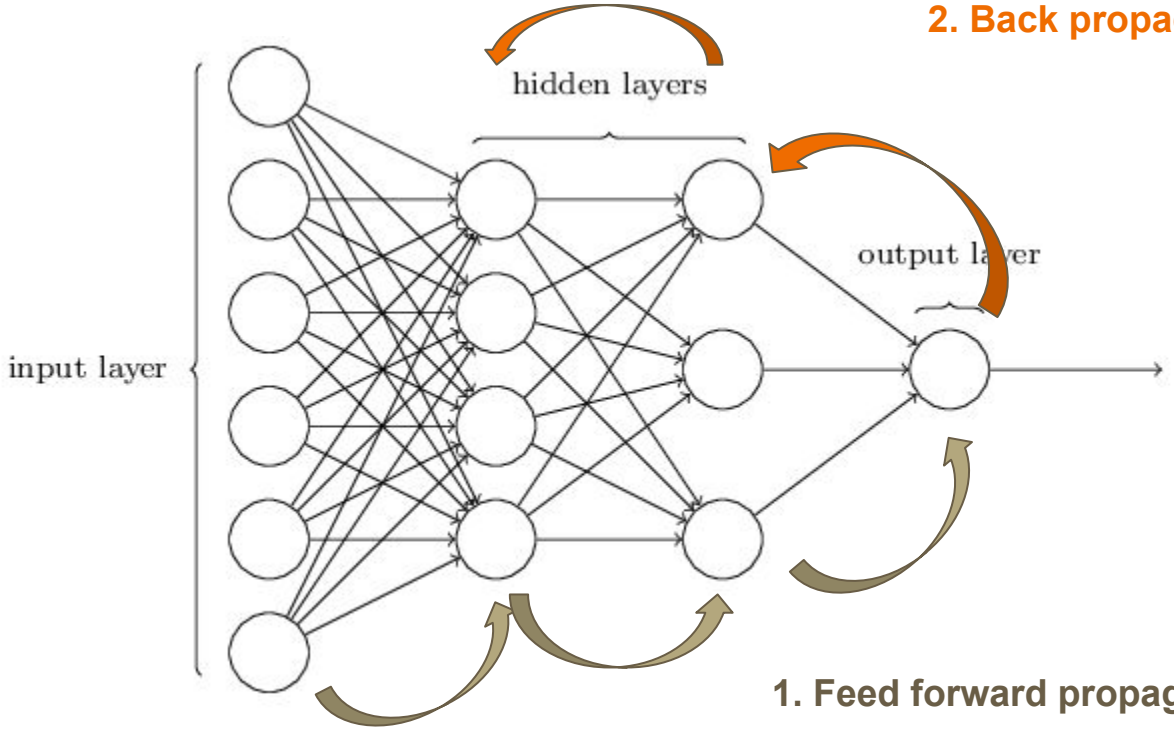


Backpropagate

- Công thức chi tiết về thuật toán trong bước này khá phức tạp do có nhiều tham số
- Hãy hỏi mentor nếu bạn muốn tìm hiểu thêm
- Thư viện Tensorflow sẽ giúp đơn giản hóa bước này! (:

```
with tf.name_scope("train_step") as scope:
        train_step = tf.train.GradientDescentOptimizer(0.025).minimize(cost)
```

Backpropagation algorithm



Questions?