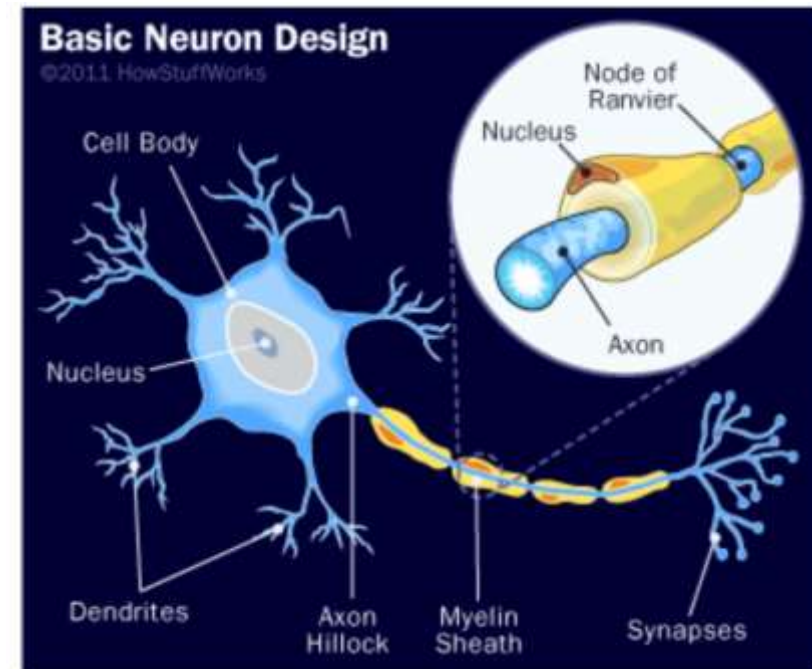


Neural network

Neural network

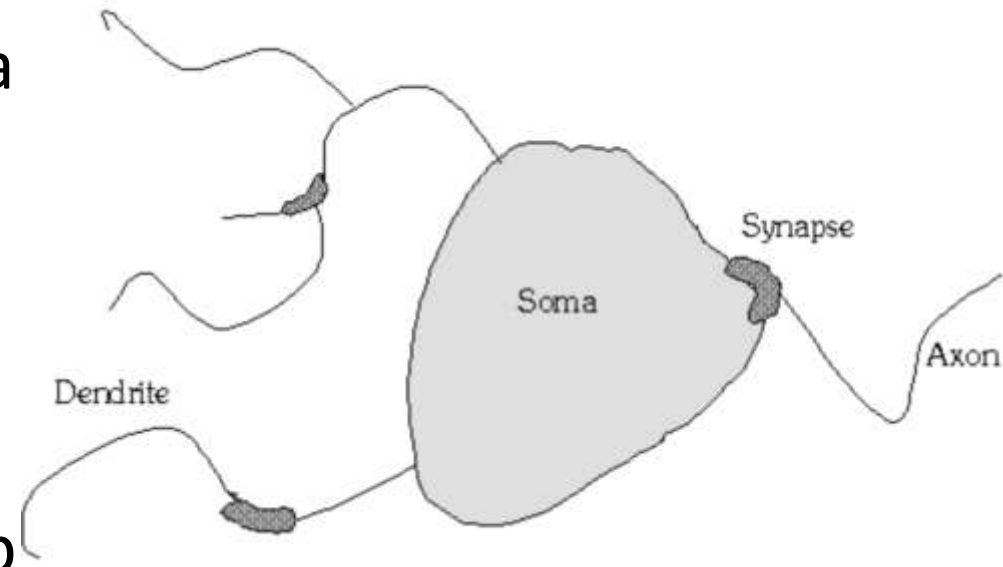
- Mạng neural nhân tạo (Artificial Neural Networks : ANN) ra đời xuất phát từ ý tưởng mô phỏng hoạt động của bộ não con người.
- Mạng neural trong một vài năm trở lại đây đã được nhiều người quan tâm và đã áp dụng thành công trong nhiều lĩnh vực khác nhau, như tài chính, y tế, địa chất, vật lý và công nghệ thông tin....
- Mạng neural nhân tạo dựa trên việc mô phỏng cấp thấp hệ thống neural sinh học.



Cấu tạo của neuron sinh học

Hoạt động của một neuron như sau:

- Neuron lấy tổng tất cả các điện thế vào mà nó nhận được, và phát ra một xung điện thế.
- Nếu tổng ấy lớn hơn một ngưỡng (threshold) nào đó. Các neuron nối với nhau ở các synapses.
- Synapse được gọi là mạnh khi nó cho phép truyền dẫn dễ dàng tín hiệu qua các neuron khác. Ngược lại, một synapse yếu sẽ truyền dẫn tín hiệu rất khó khăn.



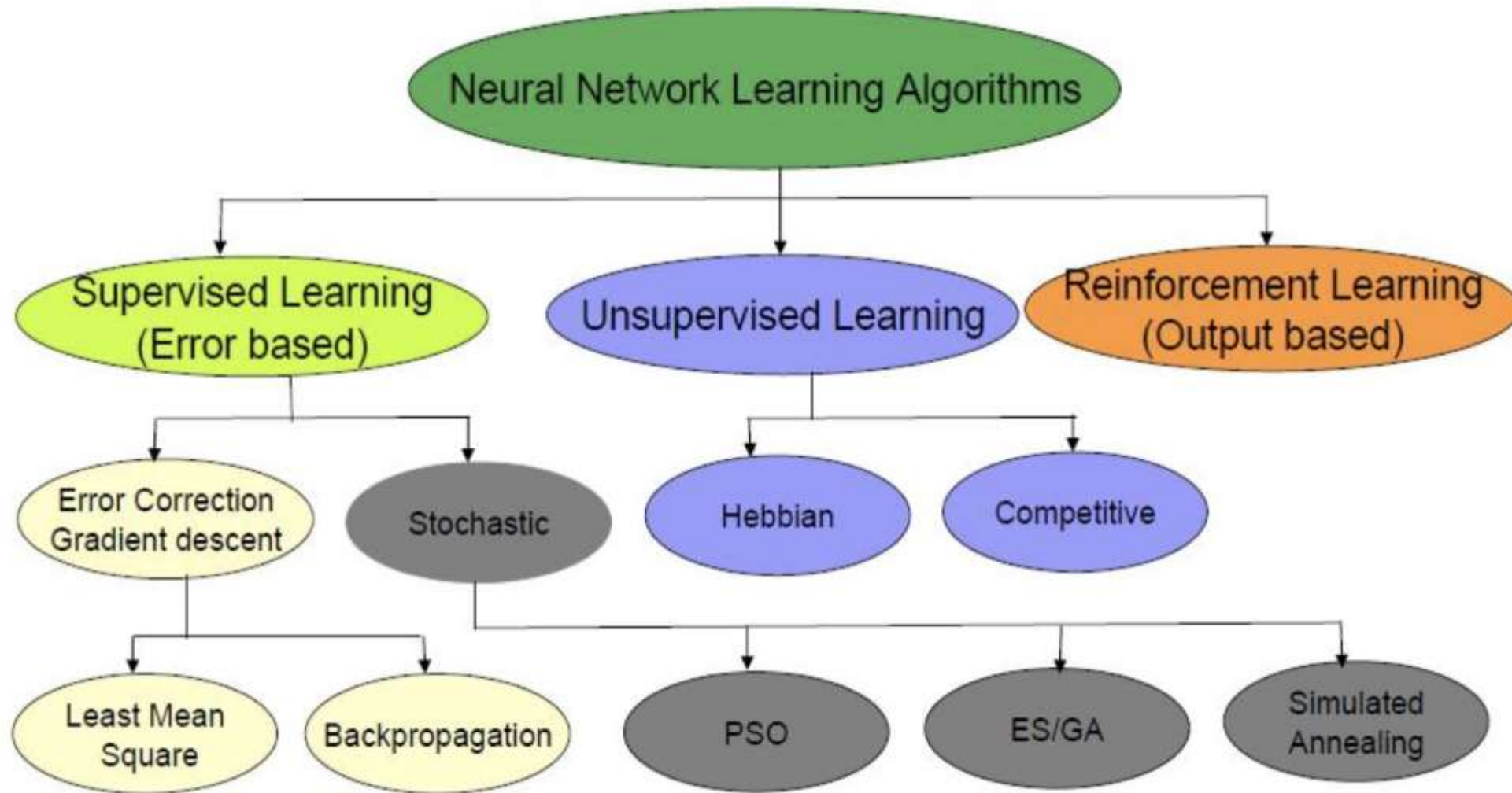
Các synapses đóng vai trò rất quan trọng trong sự học tập. Khi chúng ta học tập thì hoạt động của các synapses được tăng cường, tạo nên nhiều liên kết mạnh giữa các neuron.

--> người nào học càng giỏi thì càng có nhiều synapses và các synapses ấy càng mạnh mẽ, hay nói cách khác, thì liên kết giữa các nơron càng nhiều, càng nhạy bén.

Tại sao phải học mạng nơ-ron nhân tạo (Artificial neural network)

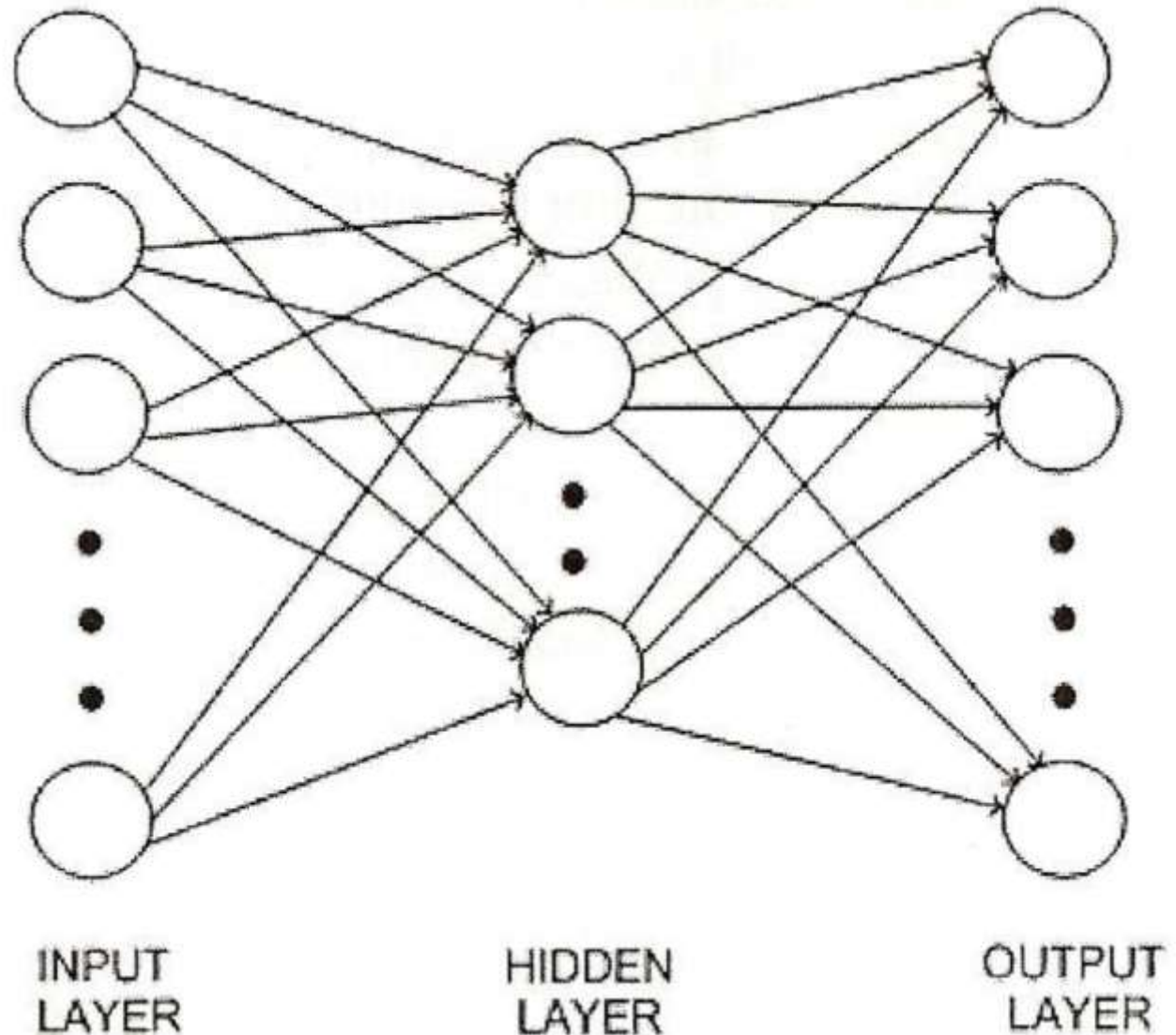
- Để hiểu được hoạt động của bộ não :
 - Bộ não con người có mạng nơ-ron thần kinh rất lớn và rất phức tạp, học mạng nơ-ron nhân tạo để tìm hiểu và mô phỏng bộ não.
- Để tìm hiểu tính toán song song lấy cảm hứng từ các tế bào thần kinh và cách kết nối của chúng.
- Để giải các bài toán thực tế bằng cách sử dụng thuật toán mới lấy cảm hứng từ bộ não con người.

Neural network cho máy học

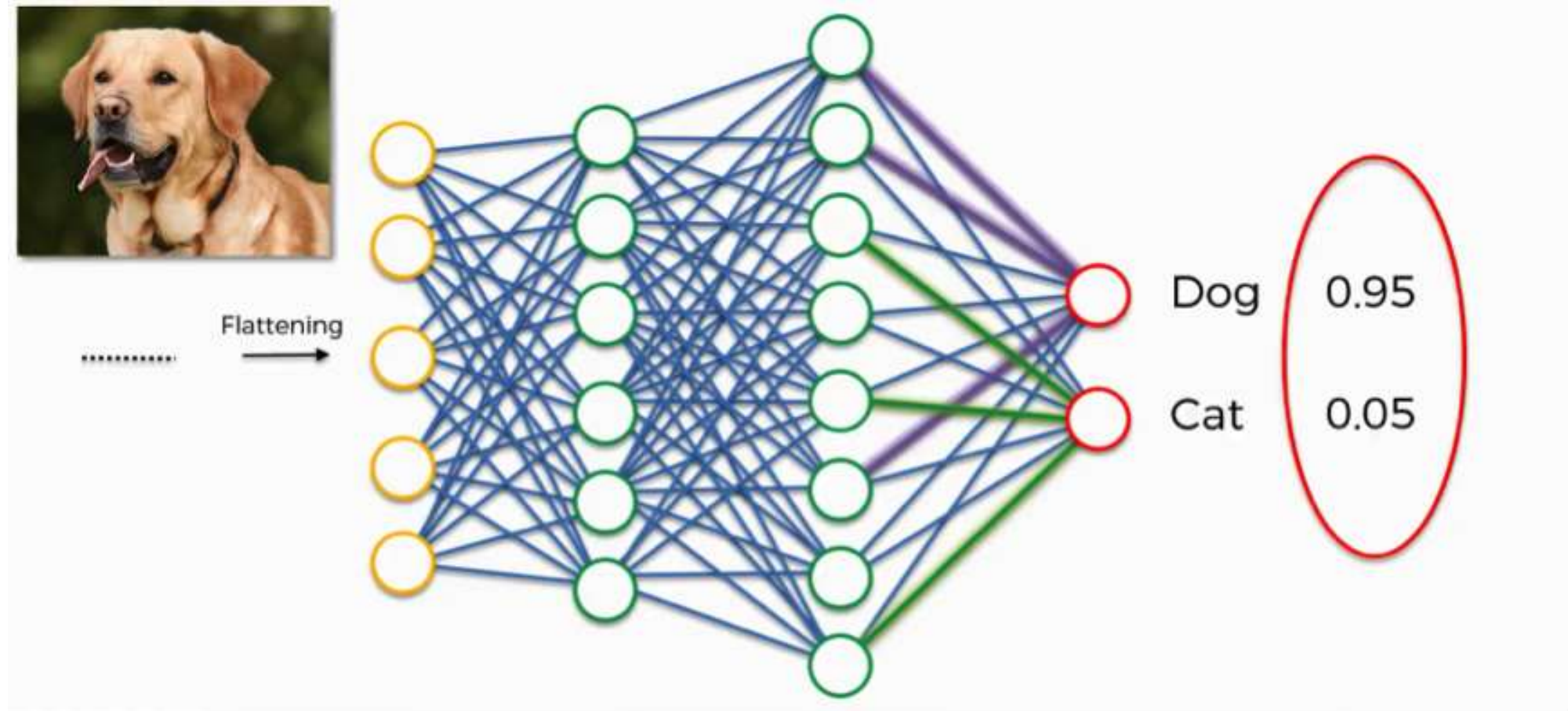


Neural network

- Có 3 loại layer:
- Các nơ-ron nhân tạo liên kết lại với nhau sẽ hình thành nên mạng nơ-ron với nhiều lớp gồm lớp đầu vào (input layer), các lớp ẩn (hidden layers) và lớp đầu ra (output layer)

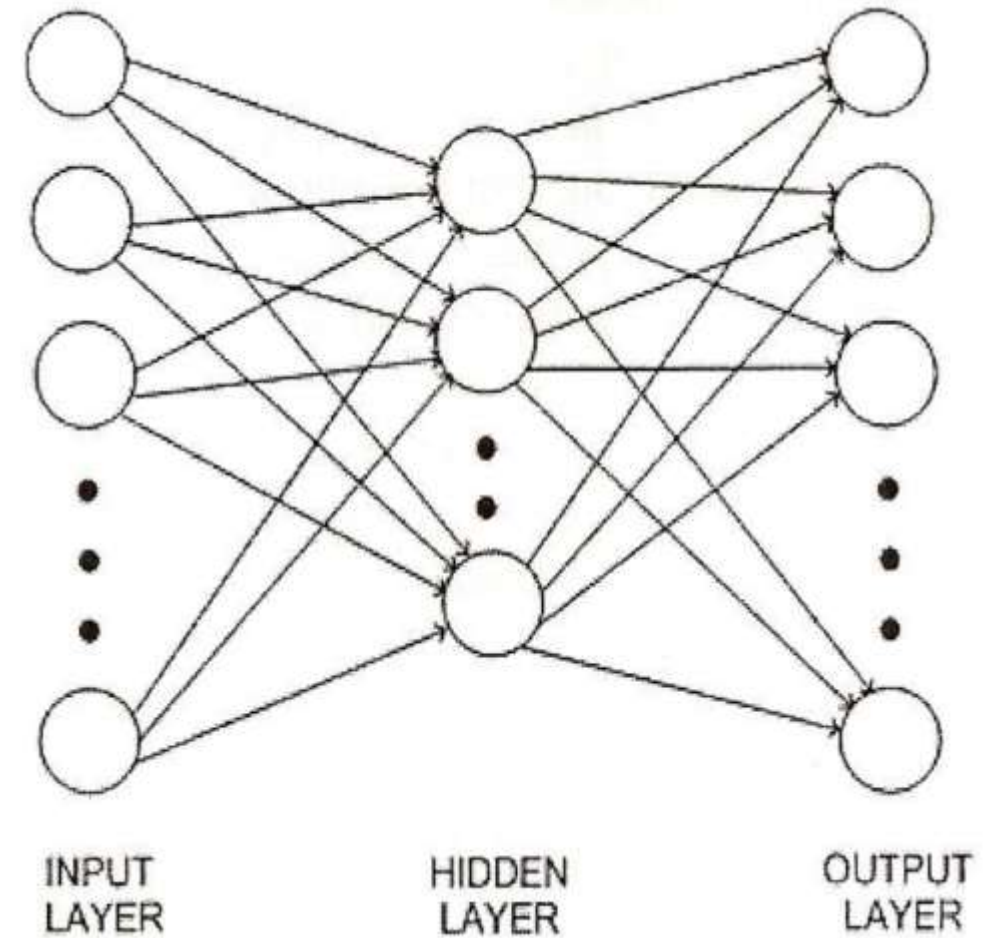


Neural network

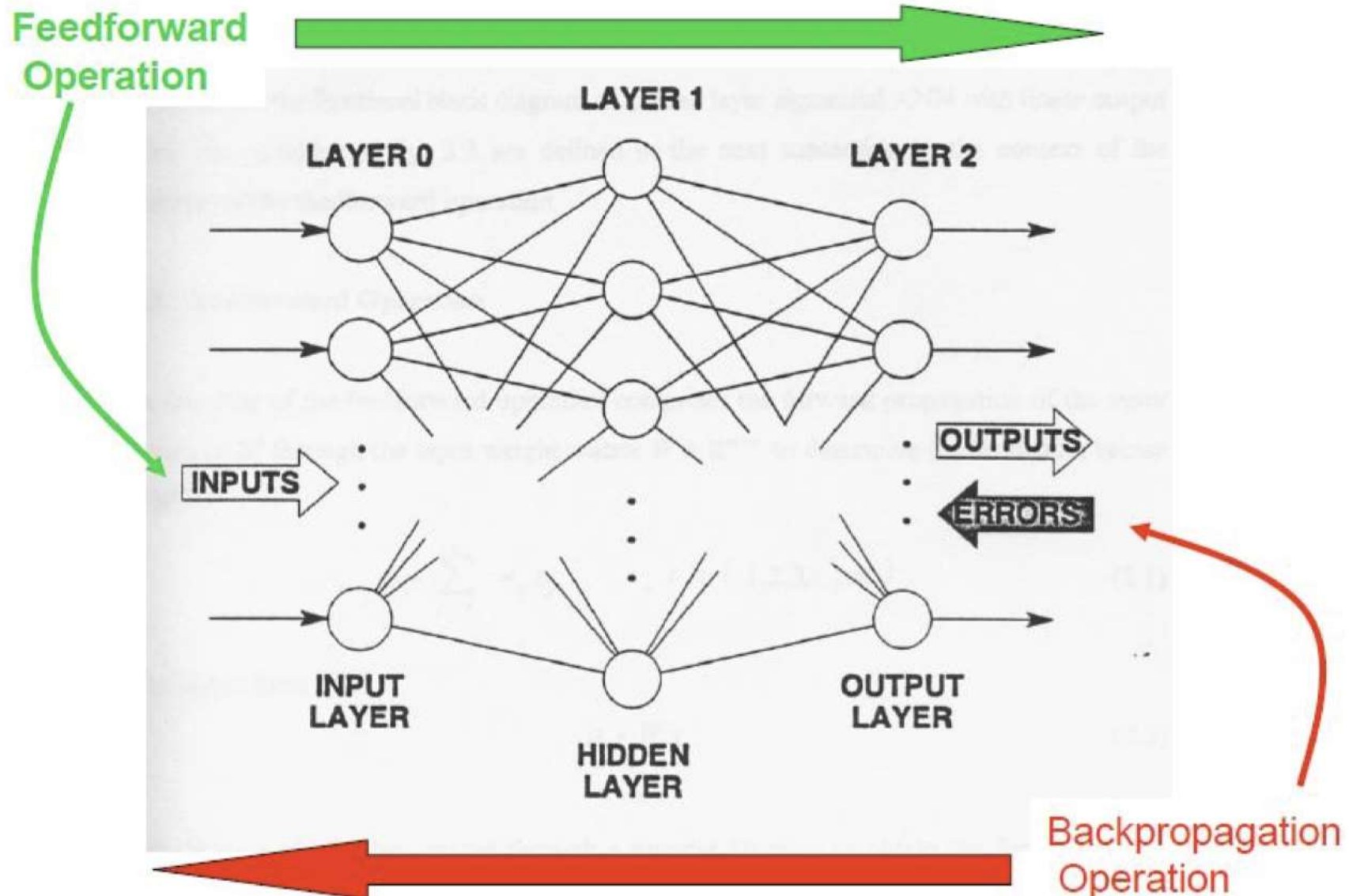


Cấu trúc neural network

- Có 2 cấu trúc mạng chính:
- Feed-Forward Network(
mạng nơ-ron truyền thẳng)



Feed-Forward Network(mạng nơ-ron truyền thẳng)



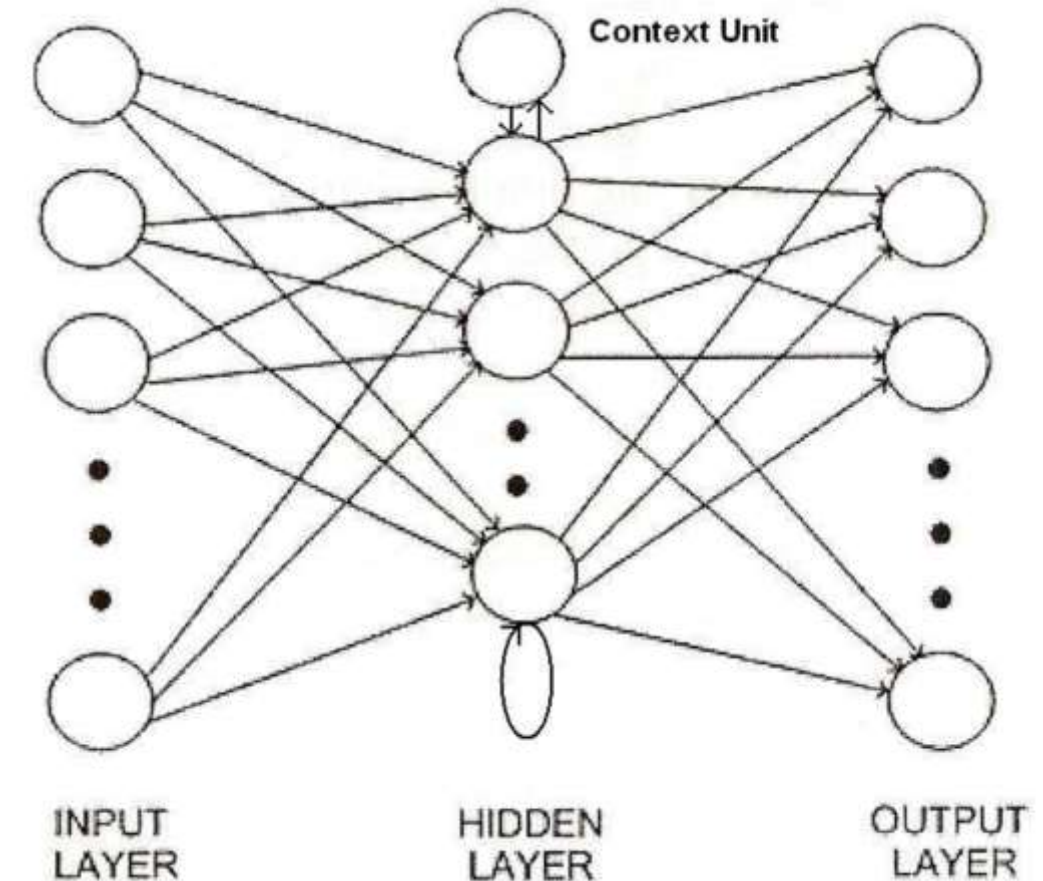
Recurrent Network(mạng hồi quy)

Ứng dụng trong lĩnh vực xử lý ngôn ngữ tự nhiên:

- Dịch máy(Machine Translation)
- Nhận dạng tiếng nói (Speech Recognition)
- Phát sinh câu mô tả cho hình ảnh (Generating Image Description)
- Gán nhãn từ loại (Part of Speech tagging)

Những cải tiến:

- Bidirectional Recurrent Neural Network
- Deep Recurrent Neural Network
- Long short-term memory



Cấu tạo của neuron nhân tạo

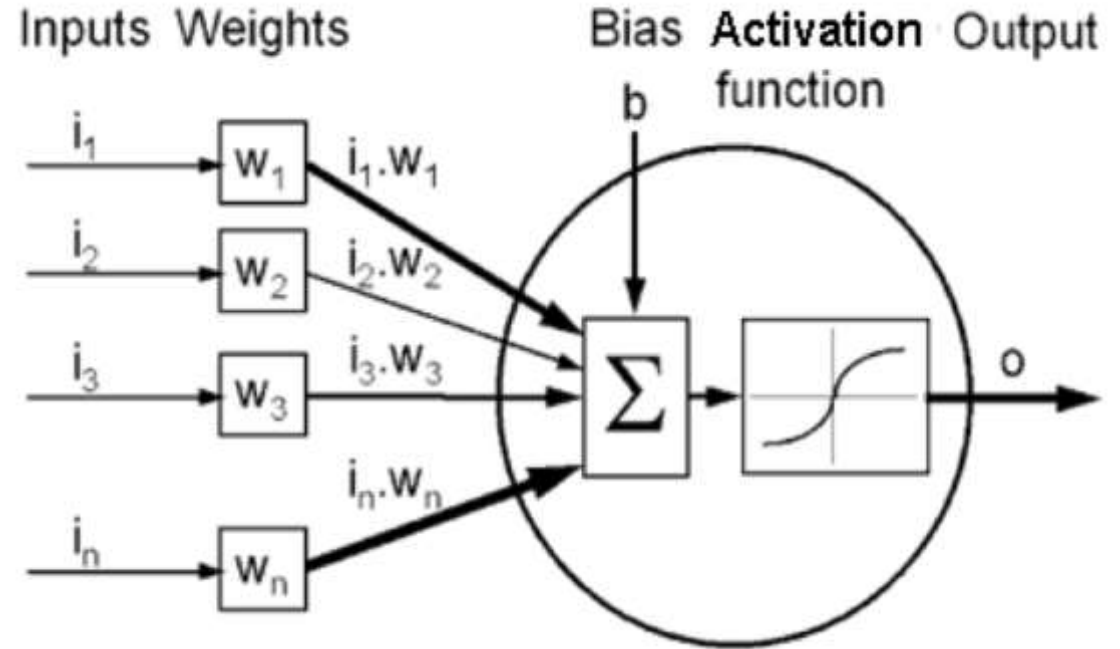
- Neuron nhân tạo là 1 đơn vị tính toán có nhiều đầu vào và 1 đầu ra.
- Neuron hoạt động như sau:
- Giả sử có n input, neuron sẽ có n trọng số w (weight), tương ứng với n đường truyền input.
- Neuron sẽ lấy tổng có trọng số của các input:

$$y = \sum_{j=1}^n w_j i_j$$

- Kết quả này sẽ so sánh với 1 ngưỡng t của neuron, và thông qua hàm kích hoạt (activation function) để giới hạn biên độ đầu ra của neuron.

$$y = f\left(\sum_{j=1}^n w_j i_j - t\right)$$

f - hàm kích hoạt, $b = -t$ được gọi là bias(độ lệch với ngưỡng ra) của neuron.



Nếu đưa thêm 1 input thứ 0 vào, thì $\text{input}=1$ và w sẽ bằng bias và neuron viết dưới dạng:

$$y = \sum_{j=0}^n w_j i_j$$

Hàm kích hoạt(Activation Functions)

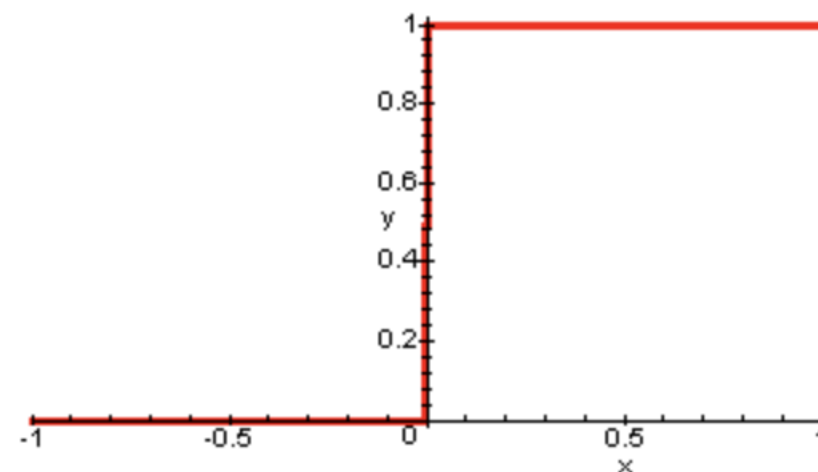
- Hàm kích hoạt nơ-ron:

$$h(\mathbf{x}) = f(\mathbf{w}^T \mathbf{x} + b)$$

- Hàm tuyến tính:
- Hàm ngưỡng:
- Hàm này giới hạn đầu ra, nếu $\mathbf{w}^T \mathbf{x} + b$ lớn hơn hoặc bằng 0 thì output=1, còn nếu nhỏ hơn thì output=0

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

Hàm này được gọi là hàm Heaviside hay hàm bước nhảy(step function)

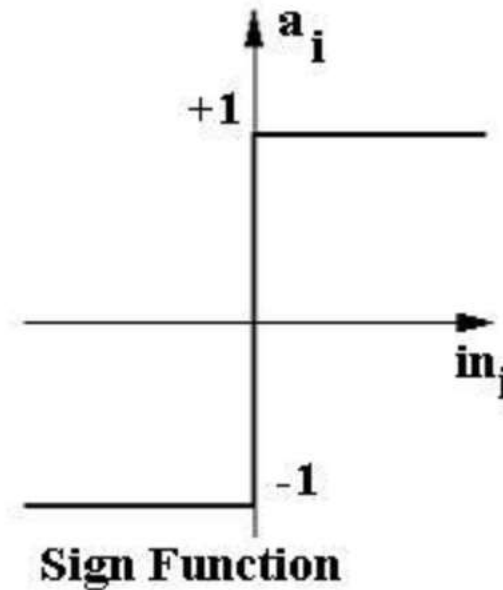


Hàm kích hoạt(Activation Functions)

- Hàm sgn: nếu $\mathbf{w}^T \mathbf{x} + b$ lớn hơn hoặc bằng 0 thì output=1, còn nếu nhỏ hơn thì output=-1

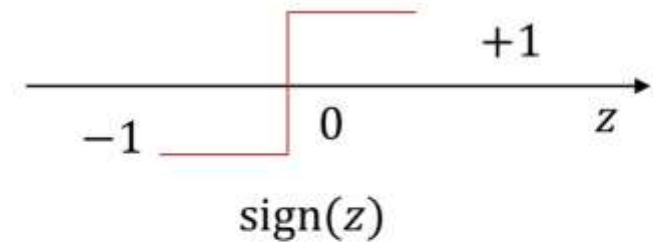
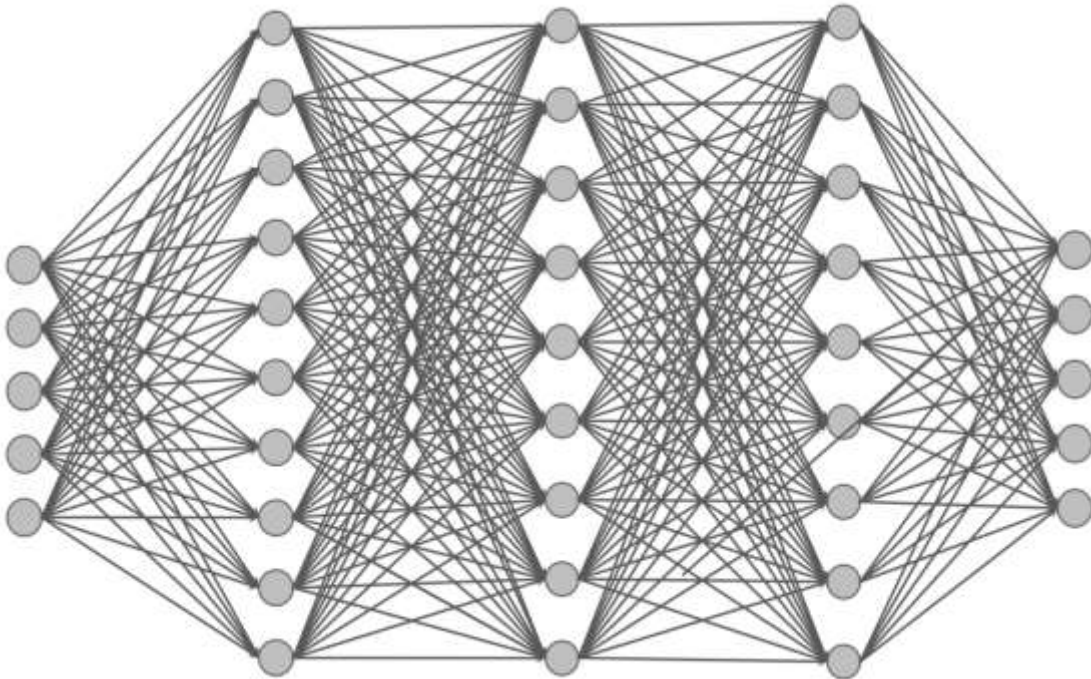
- $$f(x) = \begin{cases} 1 & \text{nếu } x \geq 0 \\ -1 & \text{nếu } x < 0 \end{cases}$$

- Sử dụng trong mô hình perceptron 1 lớp(sẽ học ở phần sau)

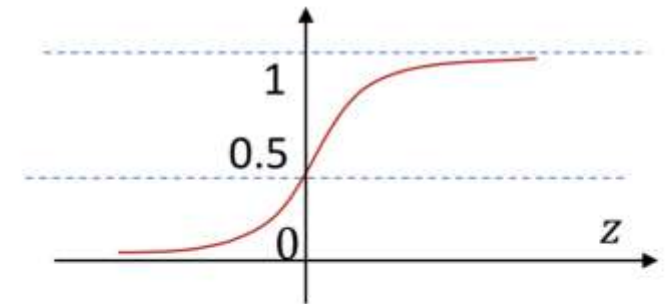


Hàm kích hoạt(Activation Functions)

- Mô hình tuyến tính chỉ dùng cho mạng neural 1 lớp, quá yếu khi xử lý dữ liệu nhiều chiều.
- Sử dụng những hàm không tuyến tính để giải quyết bài toán phức tạp với mô hình mạng neural nhiều lớp.



small step, big leap
in modelling



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

$$\frac{d\sigma(z)}{dz} = \sigma(z)(1 - \sigma(z))$$

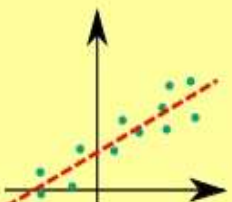
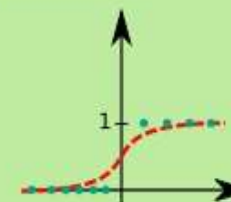
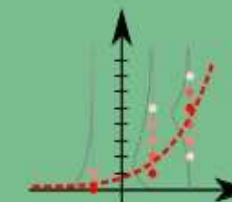
The Three Regression Types

a short guide

Generalized Linear Models (GLM) extend the ordinary linear regression and allow the response variable y to have an error distribution other than the normal distribution.

GLMs are:

- Easy to understand
- Simple to fit and interpret in any statistical package
- Sufficient in a lot of practical applications

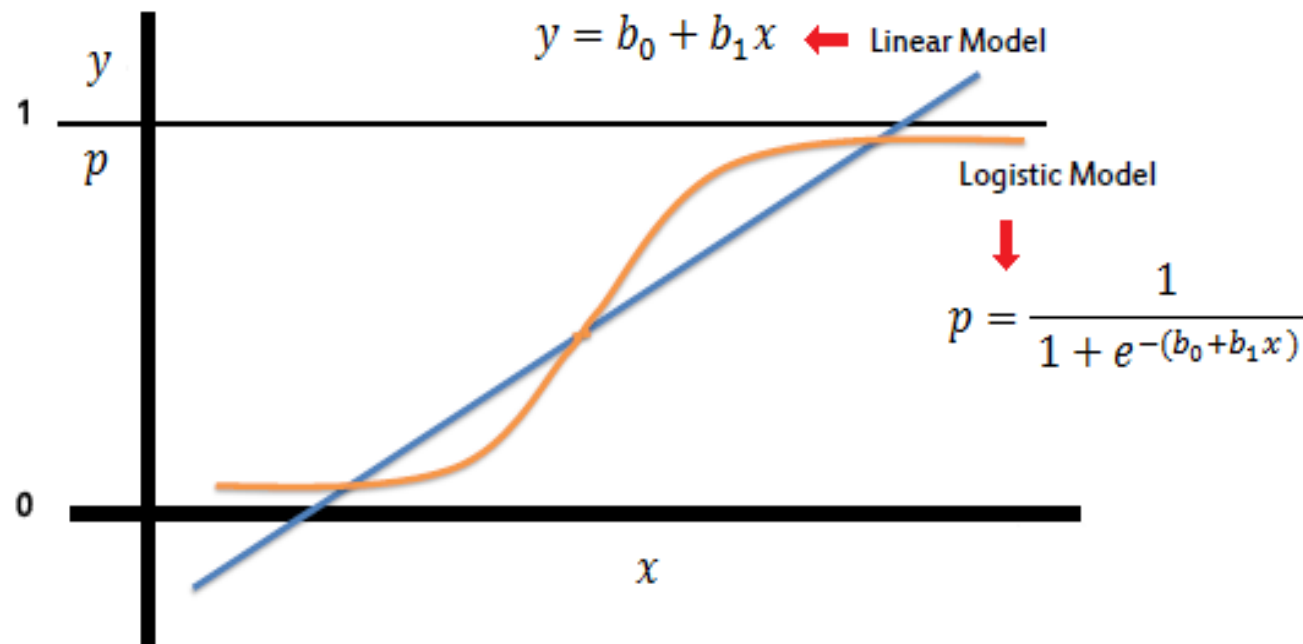
LINEAR REGRESSION	LOGISTIC REGRESSION	POISSON REGRESSION
<ol style="list-style-type: none"> Econometric modelling Marketing Mix Model Customer Lifetime Value 	<ol style="list-style-type: none"> Customer Choice Model Click-through Rate Conversion Rate Credit Scoring 	<ol style="list-style-type: none"> Number of orders in lifetime Number of visits per user
		
Continuous \Rightarrow Continuous	Continuous \Rightarrow True/False	Continuous \Rightarrow 0,1,2,...
$y = \alpha_0 + \sum_{i=1}^N \alpha_i x_i$	$y = \frac{1}{1 + e^{-z}}$ $z = \alpha_0 + \sum_{i=1}^N \alpha_i x_i$	$y \sim \text{Poisson}(\lambda)$ $\ln \lambda = \alpha_0 + \sum_{i=1}^N \alpha_i x_i$
<code>lm(y ~ x1 + x2, data)</code>	<code>glm(y ~ x1 + x2, data, family=binomial())</code>	<code>glm(y ~ x1 + x2, data, family=poisson())</code>
1 unit increase in x increases y by α	1 unit increase in x increases log odds by α	1 unit increase in x multiplies y by e^α

MarketingDistillery.com is a group of practitioners in the area of e-commerce marketing.

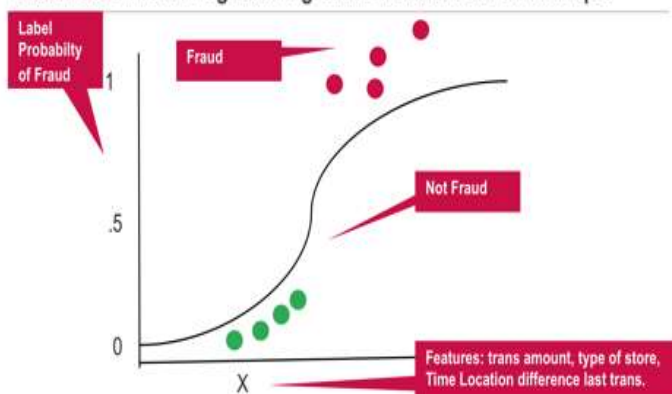
Our fields of expertise include: marketing strategy and optimization, customer tracking and on-site analytics, predictive analytics, econometrics, data warehousing and big data systems, marketing channel insights in Paid Search, Social, SEO, CRM and brand.

Marketing
DISTILLERY

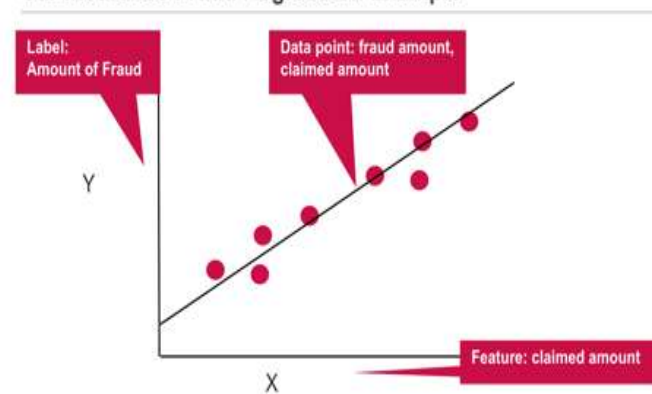
(cc-by) Kamil Bartocha, MarketingDistille



Credit Card Fraud Logistic Regression Classification Example



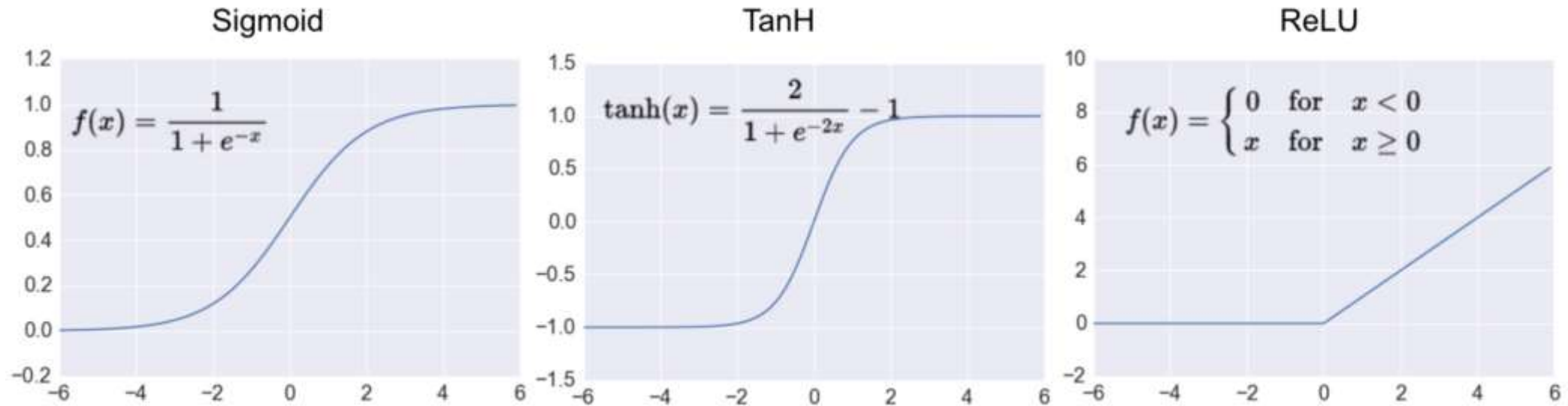
Car Insurance Fraud Regression Example



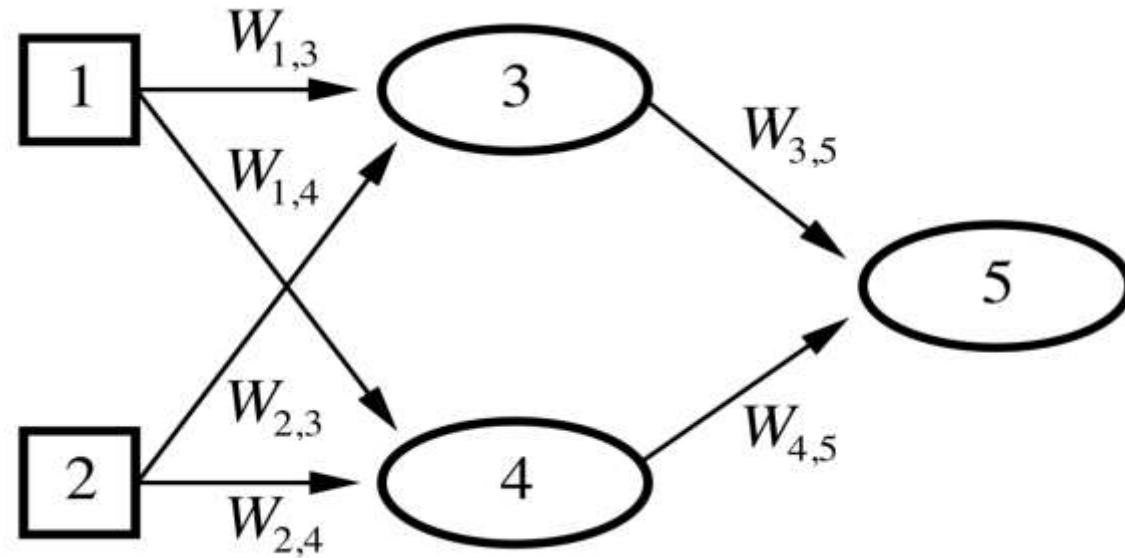
$$\text{AmntFraud} = \text{intercept} + \text{coeff} * \text{claimedAmnt}$$

Hàm kích hoạt(Activation Functions)

- Hàm không tuyến tính:



Feed-forward example



Giá trị đầu ra nhận được:

$$\begin{aligned} a_5 &= g(W_{3,5} \cdot a_3 + W_{4,5} \cdot a_4) \\ &= g(W_{3,5} \cdot g(W_{1,3} \cdot a_1 + W_{2,3} \cdot a_2) + W_{4,5} \cdot g(W_{1,4} \cdot a_1 + W_{2,4} \cdot a_2)) \end{aligned}$$

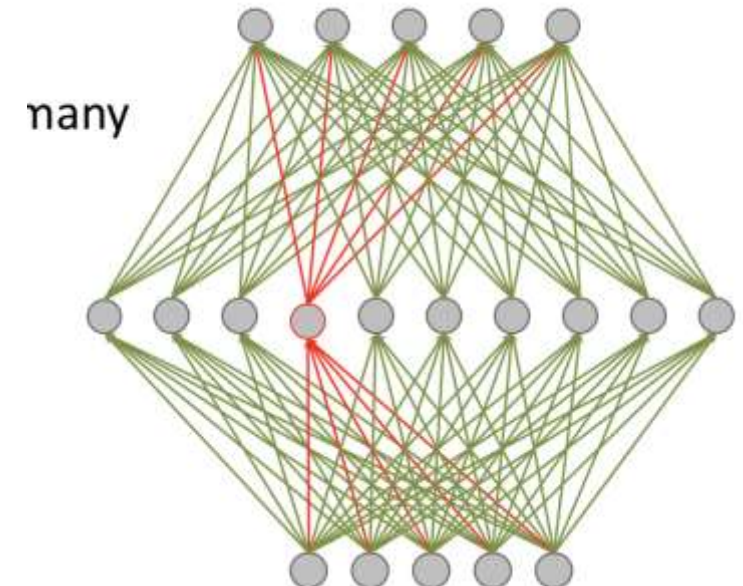
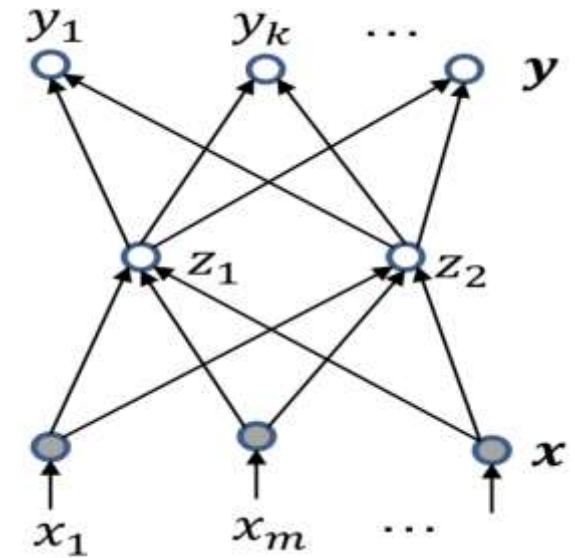
Hàm mất mát và vấn đề tối ưu hoá

- Với giá trị đầu vào x_t và đầu ra mong muốn y_t , $t = 1, \dots, n$ tìm trọng số mạng w sao cho :

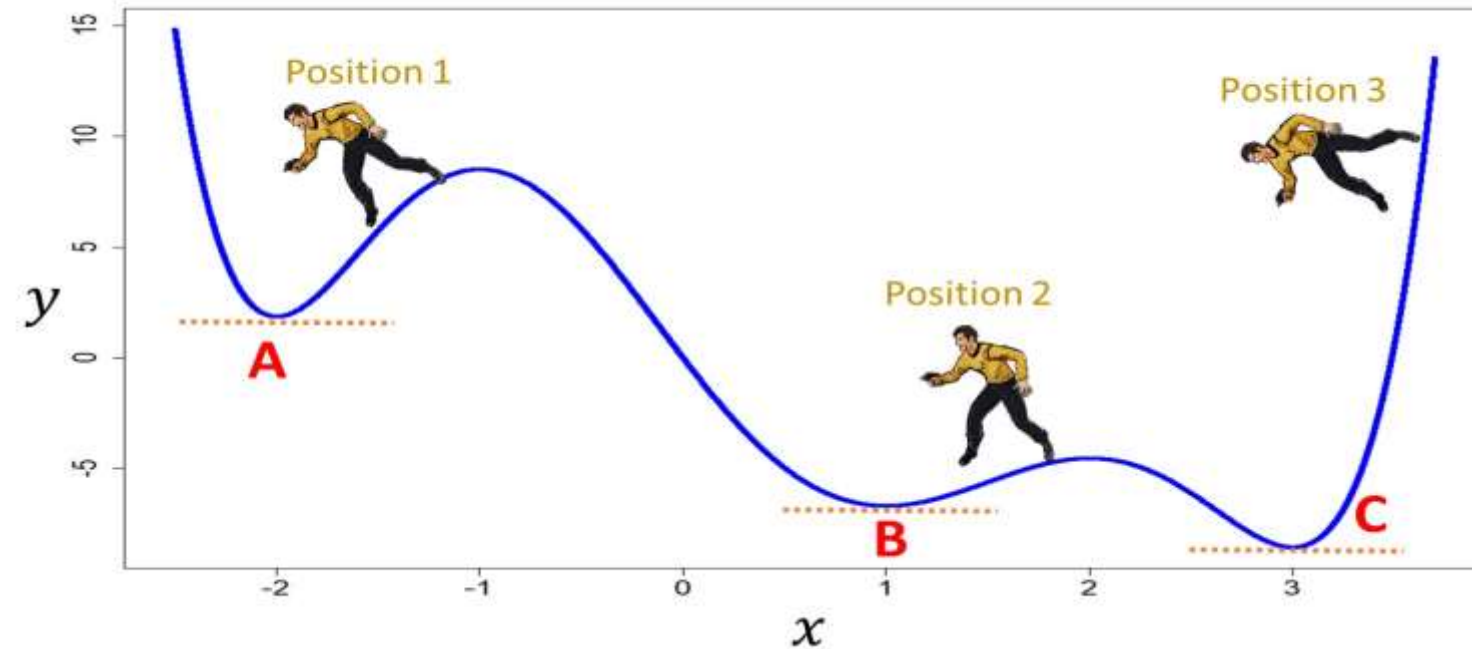
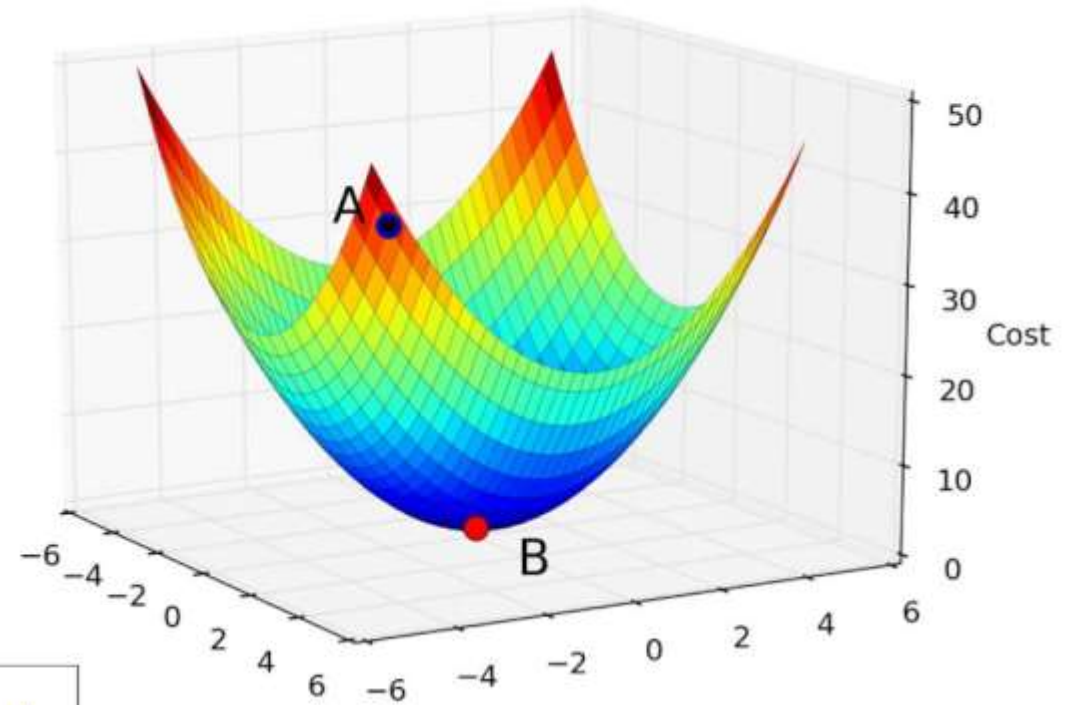
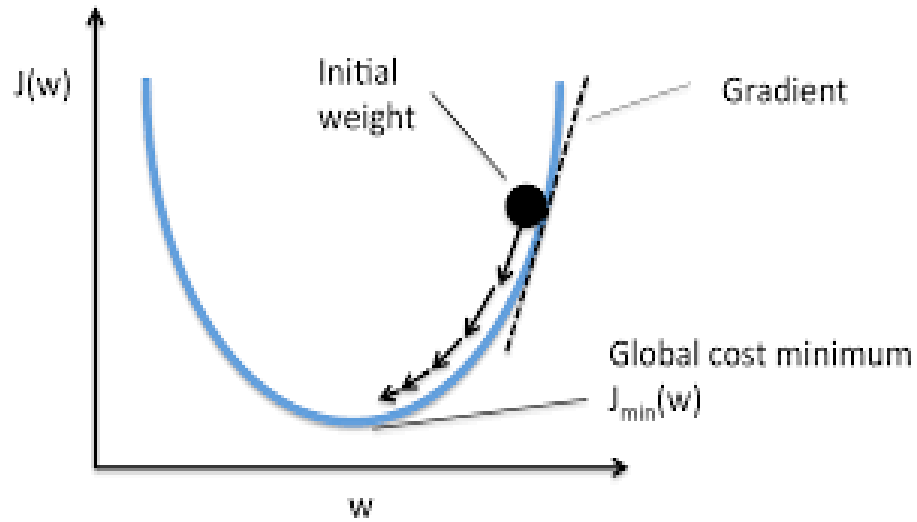
$$\hat{y}_t \approx y_t, \forall t$$

- Giải quyết vấn đề tối ưu hoá: tìm trọng số w để hàm lỗi(hàm mất mát) đạt giá trị nhỏ nhất:

$$J(w) = \frac{1}{2} \sum_{t=1}^M \sum_{k=1}^K (y_{tk} - \hat{y}_{tk})^2$$

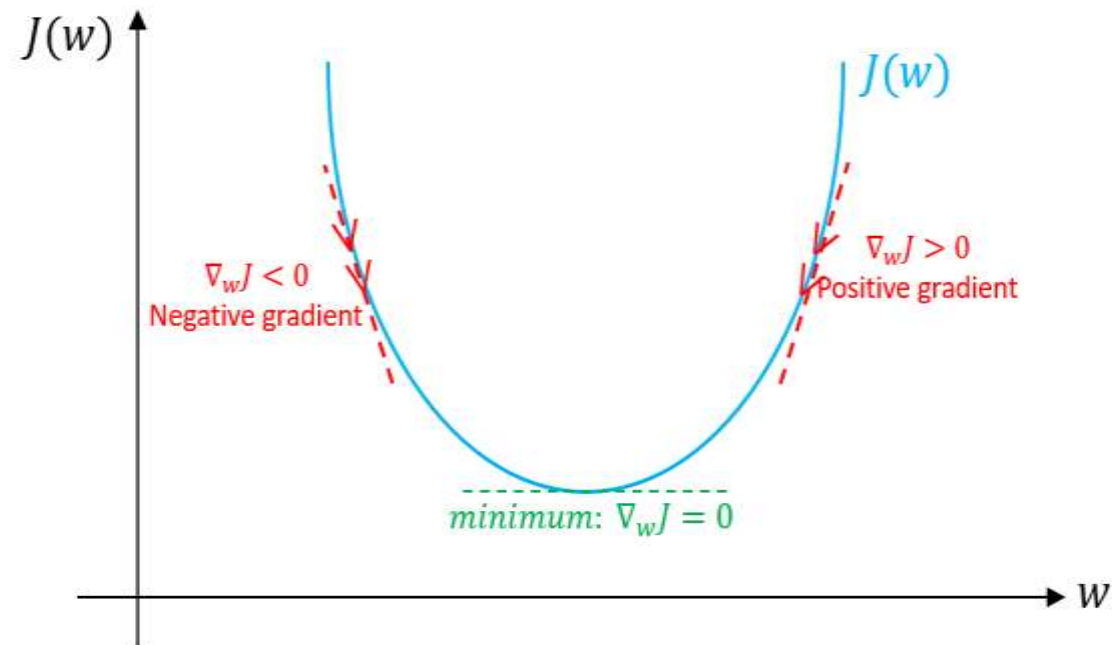
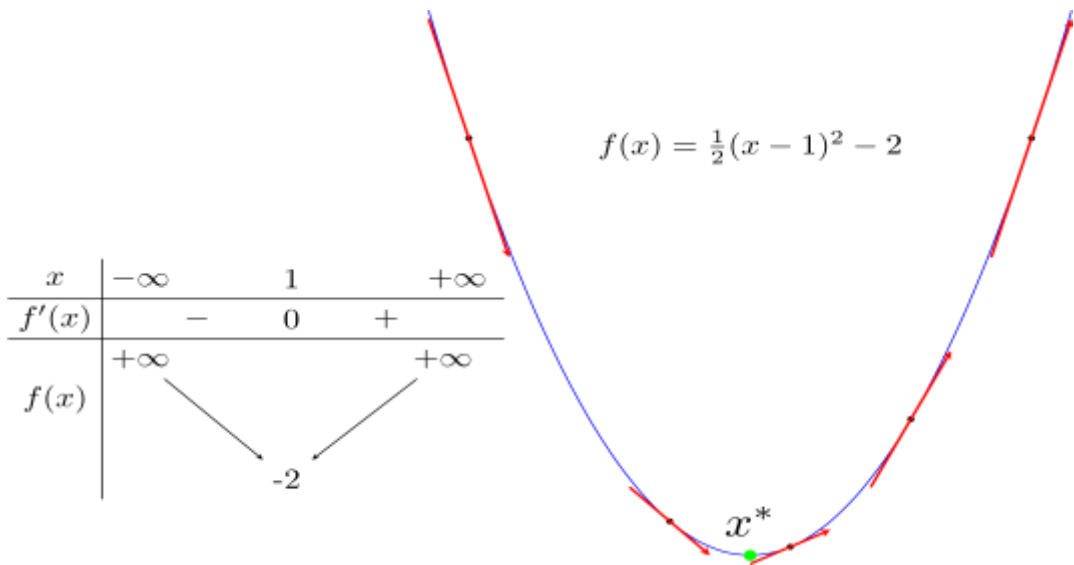


Giảm Gradient



*Global minimum????
Local minimum !!*

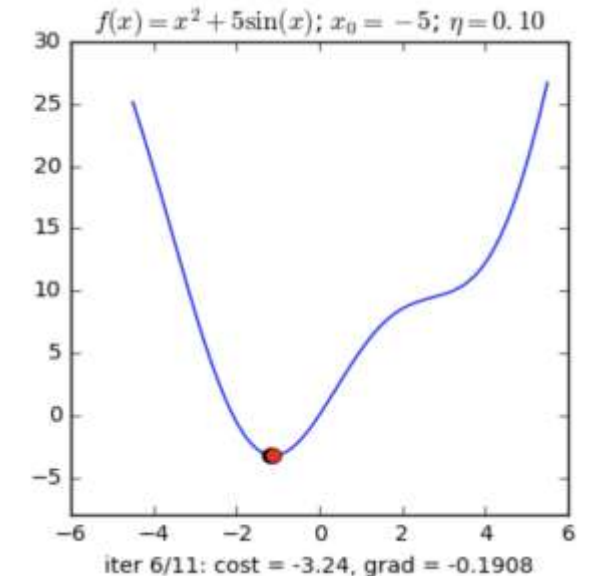
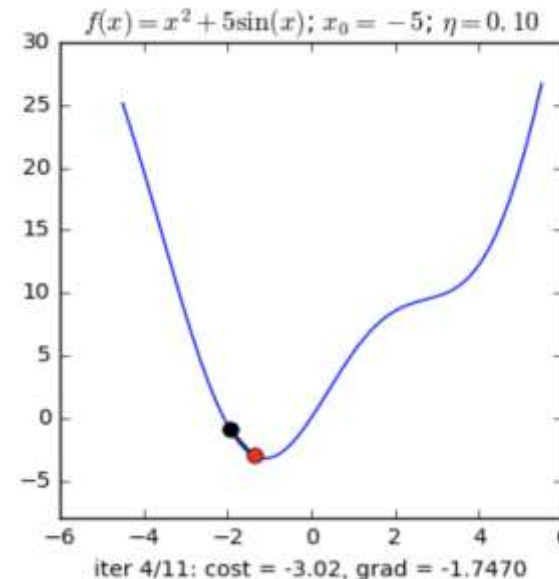
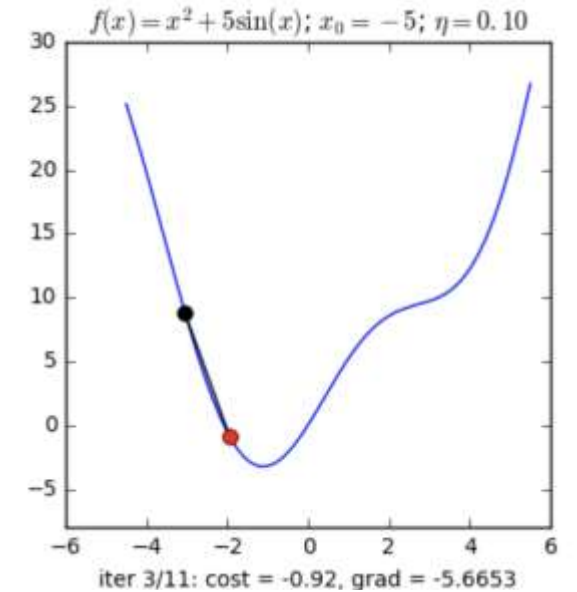
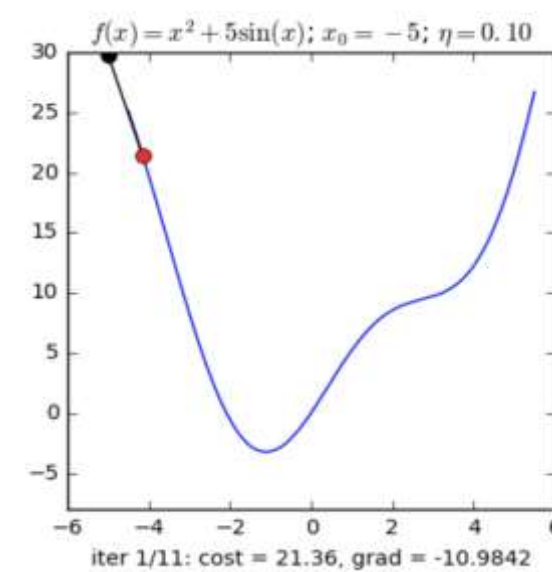
Giảm Gradient



- Nếu đạo hàm của hàm số tại x_t : $f'(x_t) > 0$ thì x_t nằm về bên phải so với x^* (và ngược lại).
- Để điểm tiếp theo x_{t+1} gần với x^* hơn, **ta cần di chuyển ngược dấu với đạo hàm.**
- x_t càng xa x^* về phía bên phải thì $f'(x_t)$ càng lớn hơn 0 (và ngược lại).
- Hai nhận xét phía trên ta rút ra: $x_{t+1} = x_t - \eta f'(x_t)$
 η - tốc độ học (*learning rate*)

Giảm gradient

- Tốc độ hội tụ của GD không những phụ thuộc vào điểm khởi tạo ban đầu và giá trị tốc độ học *learning rate*
- Việc lựa chọn *learning rate* rất quan trọng trong các bài toán thực tế.



Giảm gradient

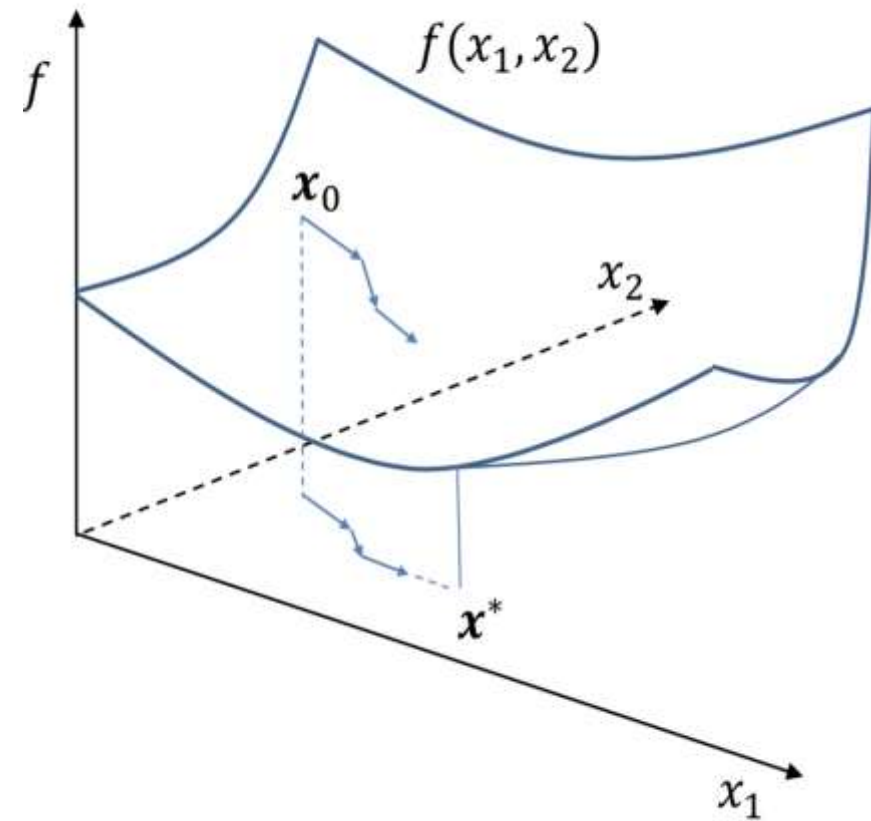
Giảm gradient cho hàm nhiều biến:

- Tại mỗi điểm $\mathbf{x} = (x_1, x_2)$, vector gradient của hàm $f(\mathbf{x})$ đối với biến \mathbf{x} là:

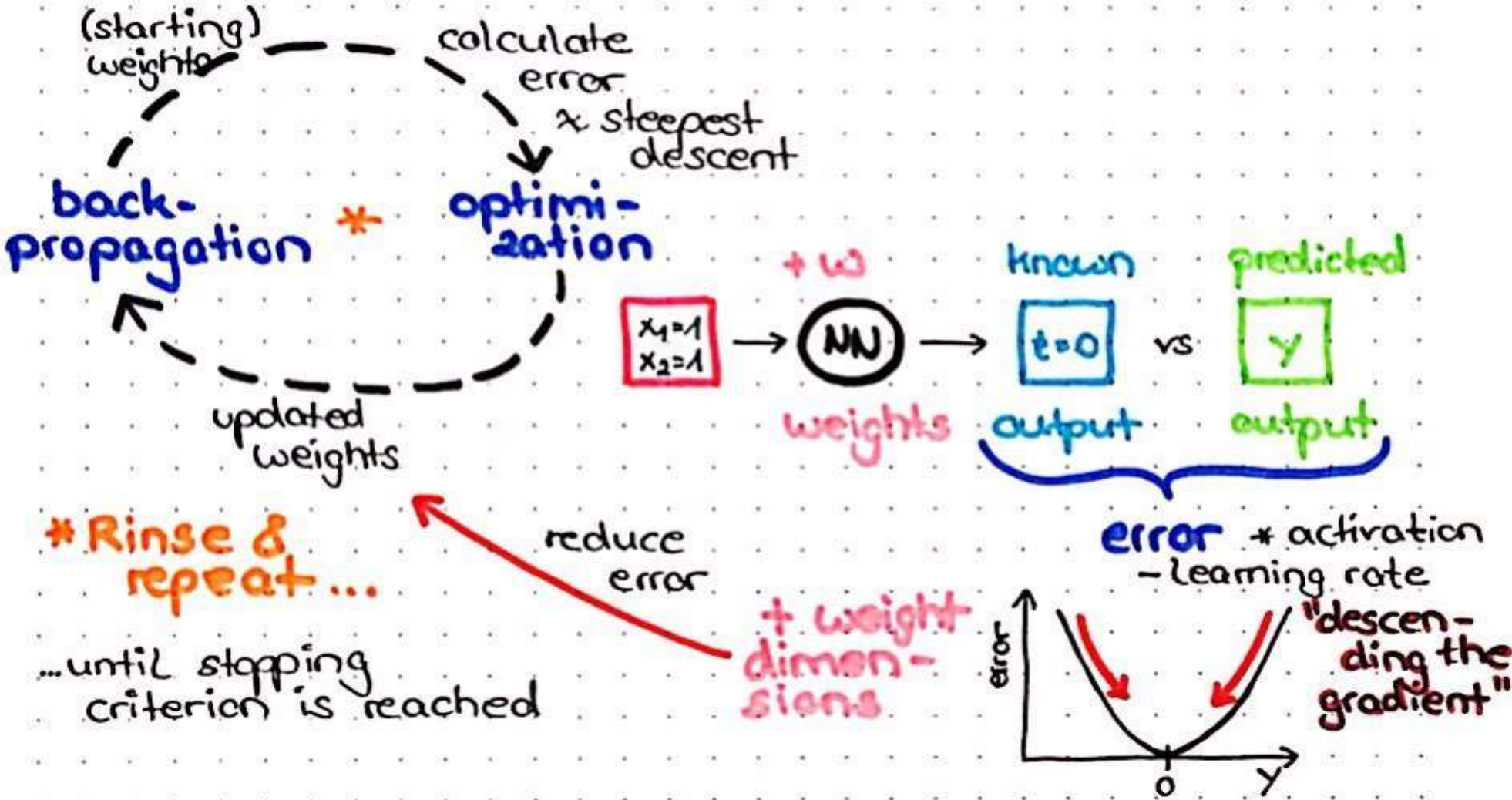
$$\nabla f(\mathbf{x}) = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2} \right)$$

- Giá trị \mathbf{x}_{t+1} được tính bằng công thức:

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \eta \times \nabla f(\mathbf{x}_t)$$

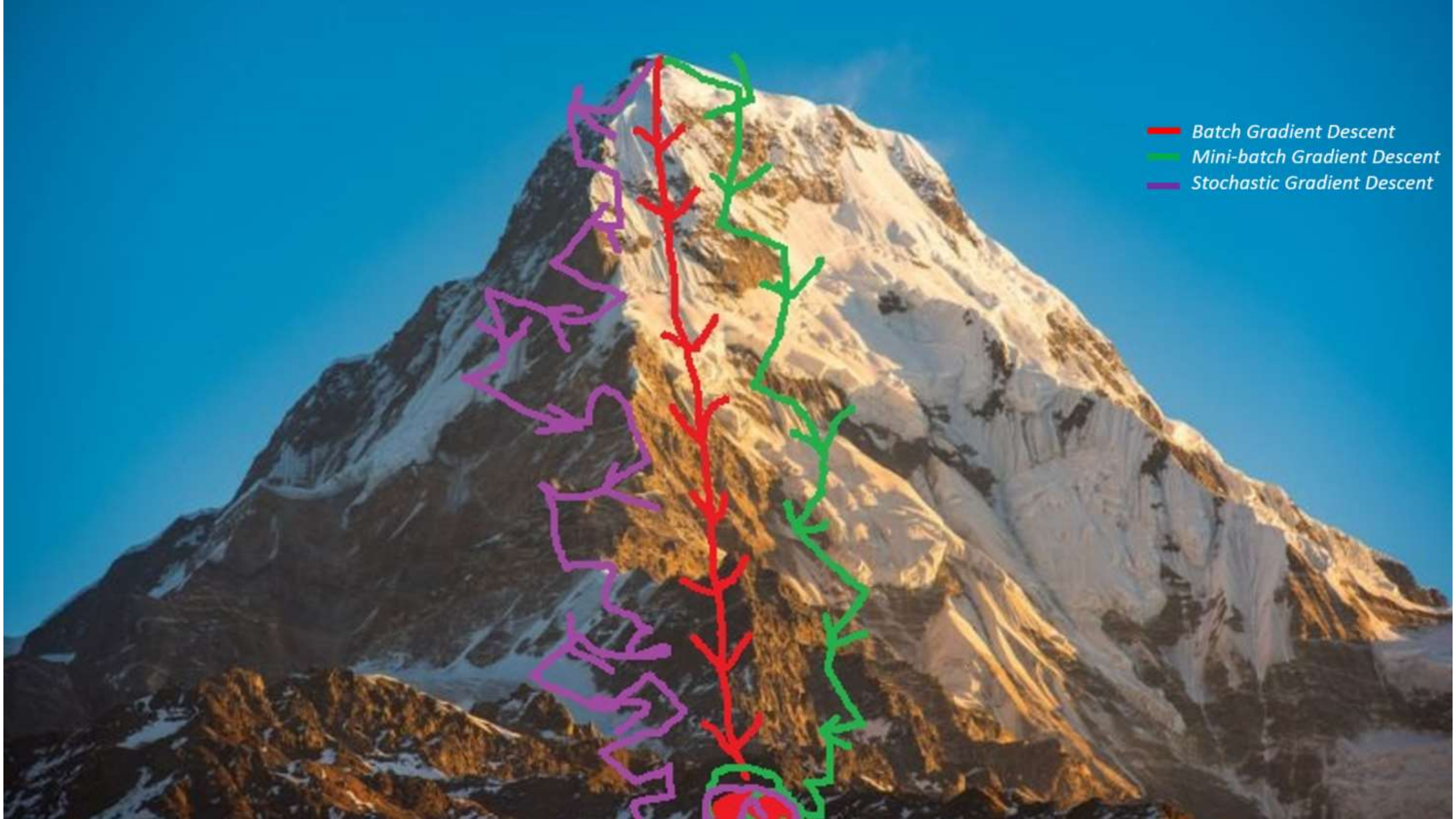


Giảm Gradient



Điều kiện dừng

- Điều kiện dừng của GD có thể là:
- Kết thúc tất cả các *epochs* đã được định sẵn.
- Giá trị của hàm mất mát đủ nhỏ và độ chính xác của model đủ lớn.
- Hàm mất mát có giá trị không thay đổi sau một số lần hữu hạn *epochs*.
- => Khó tìm được *global minimum points*, tuy nhiên có thể chấp nhận khi kết quả trả về đủ tốt



- Batch Gradient Descent
- Mini-batch Gradient Descent
- Stochastic Gradient Descent

Một số thuật toán tối ưu

- SGD
- Adam
- RMSprop

Name	Update Rule
SGD	$\Delta\theta_t = -\alpha g_t$
Momentum	$m_t = \gamma m_{t-1} + (1 - \gamma)g_t,$ $\Delta\theta_t = -\alpha m_t$
Adagrad	$G_t = G_{t-1} + g_t^2,$ $\Delta\theta_t = -\alpha g_t G_t^{-1/2}$
Adadelta	$v_t = \beta_2 v_{t-1} + (1 - \beta_2)g_t^2,$ $\Delta\theta_t = -\alpha g_t v_t^{-1/2} D_{t-1}^{1/2},$ $D_t = \beta_1 D_{t-1} + (1 - \beta_1)(\Delta\theta_t/\alpha)^2$
RMSprop	$v_t = \beta_2 v_{t-1} + (1 - \beta_2)g_t^2,$ $\Delta\theta_t = -\alpha g_t v_t^{-1/2}$
Adam	$m_t = \beta_1 m_{t-1} + (1 - \beta_1)g_t,$ $v_t = \beta_2 v_{t-1} + (1 - \beta_2)g_t^2,$ $\hat{m}_t = m_t / (1 - \beta_1^t),$ $\hat{v}_t = v_t / (1 - \beta_2^t),$ $\Delta\theta_t = -\alpha \hat{m}_t \hat{v}_t^{-1/2}$

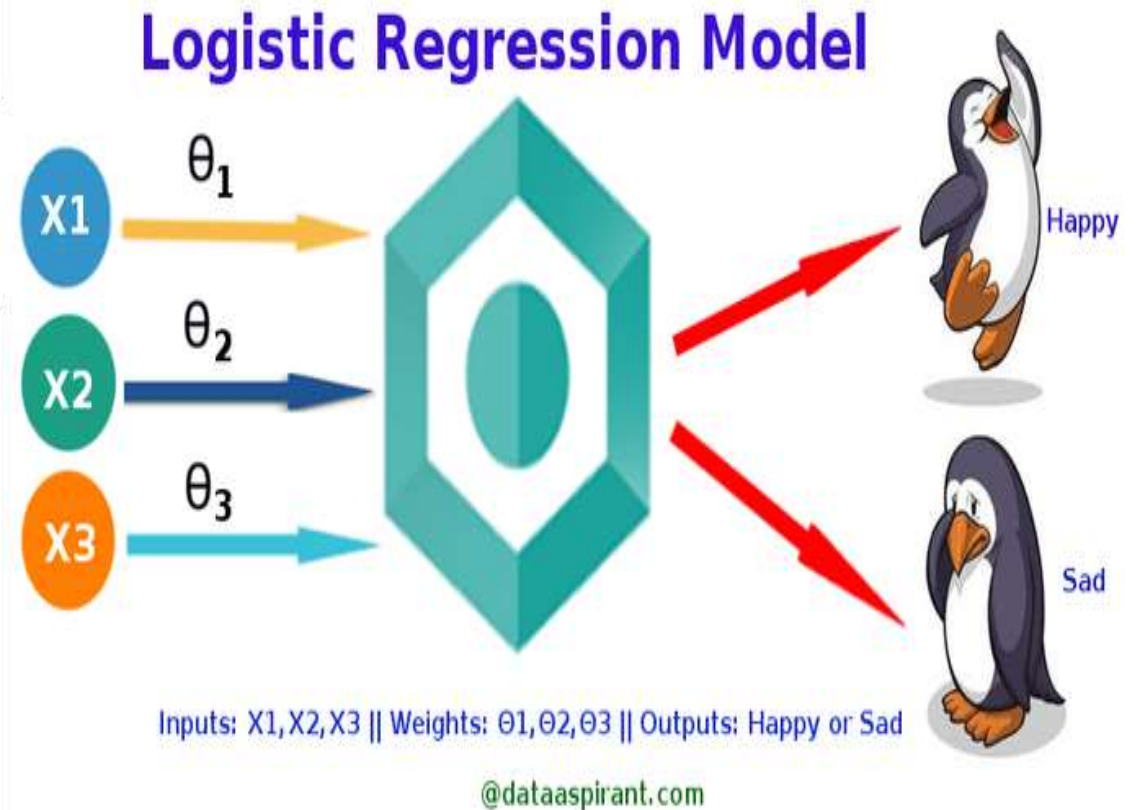
Bài toán thực tế

- Dữ liệu huấn luyện nhiều, để đảm bảo độ chính xác => so sánh tất cả các dữ liệu
- **Sample:** là một dòng dữ liệu bao gồm các inputs để đưa vào thuật toán, một output (ground-truth) để so sánh với giá trị dự đoán và tính giá trị của hàm mất mát.
- **Epochs:** số lần learning algorithm hoạt động trên model, một epoch hoàn thành là khi **tất cả dữ liệu training** được đưa vào mô hình **một lần**
- => dữ liệu quá lớn, đưa tất cả vào cùng một lúc=> không hiệu quả => chia nhỏ thành các batch.
- Một batch sẽ chứa các training samples, và số lượng các samples này được gọi là batch size
- Iteration là **số lượng batches (number of batches)** cần thiết để hoàn thành một epoch.

Linear regression and logistic regression

Linear Regression Example Data

House Price in \$1000s (Y)	Square Feet (X)
245	1400
312	1600
279	1700
308	1875
199	1100
219	1550
405	2350
324	2450
319	1425
255	1700



Linear Regression

Mean squared error

$$\text{MSE} = \frac{1}{n} \sum_{t=1}^n e_t^2$$

Root mean squared error

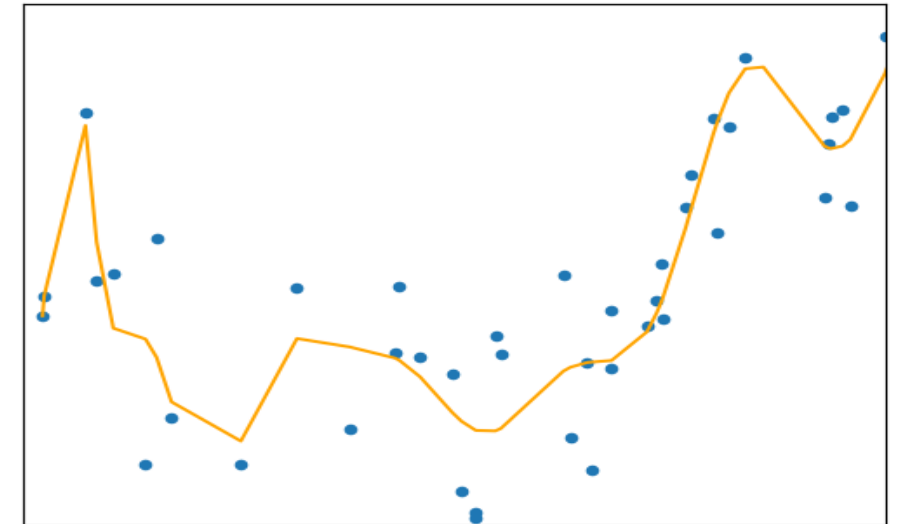
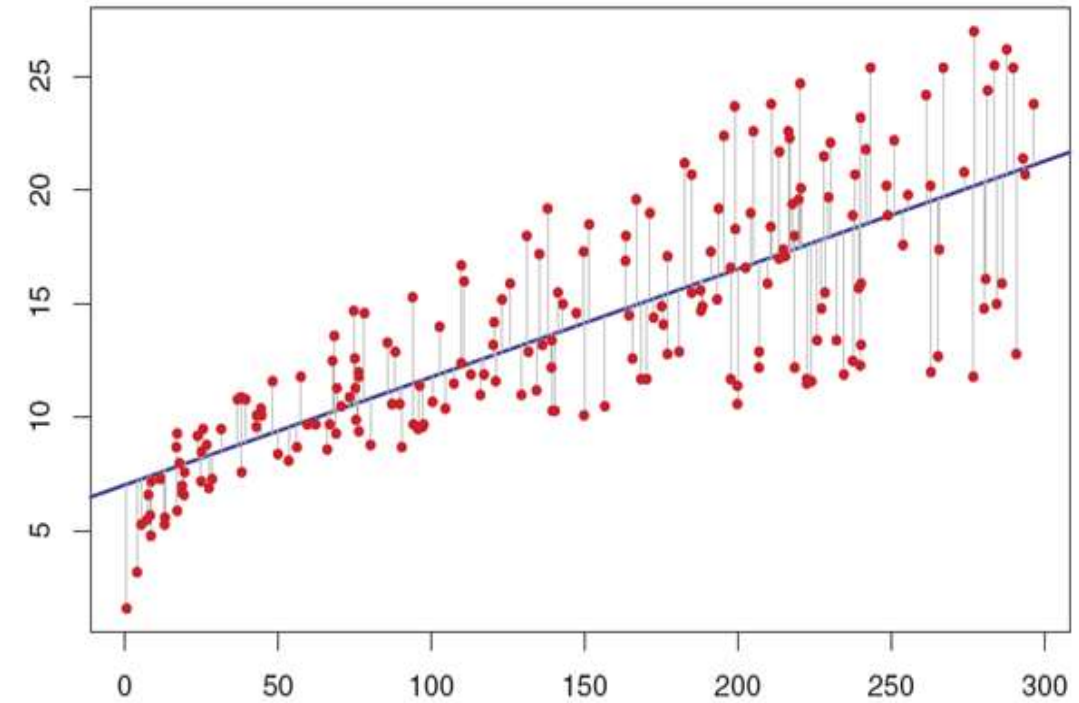
$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{t=1}^n e_t^2}$$

Mean absolute error

$$\text{MAE} = \frac{1}{n} \sum_{t=1}^n |e_t|$$

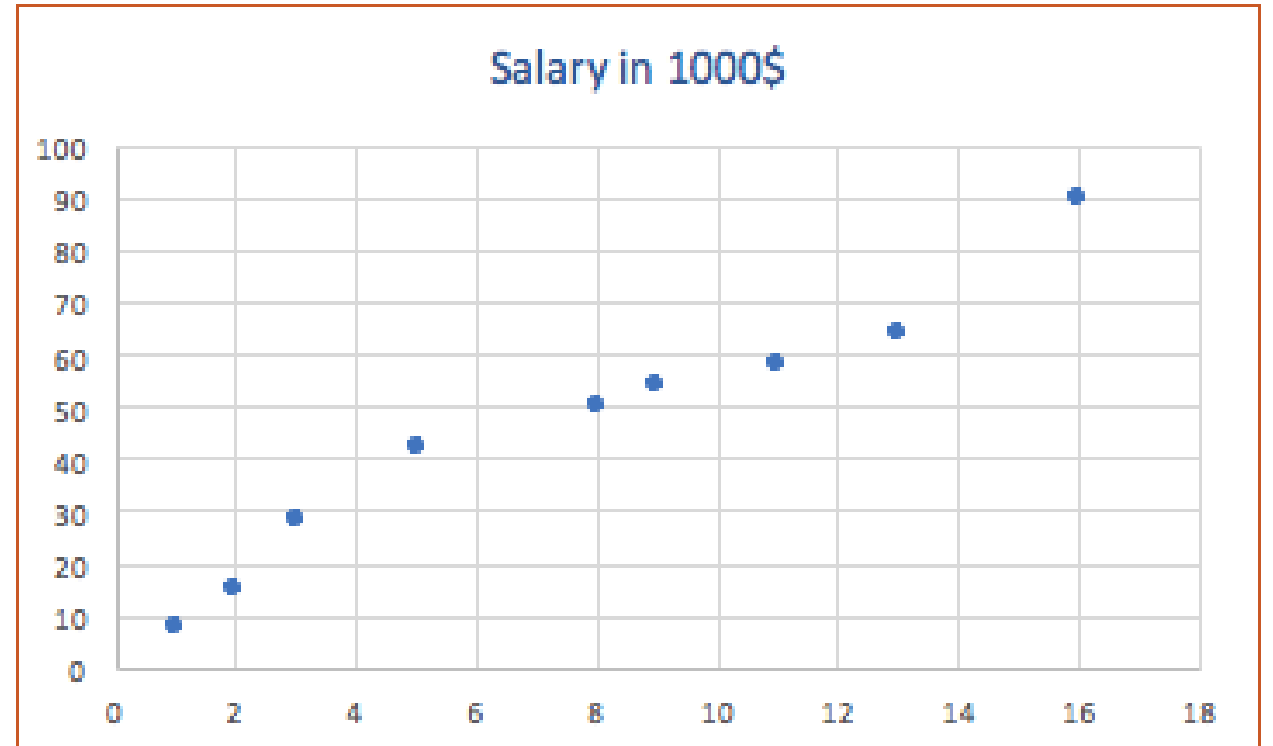
Mean absolute percentage error

$$\text{MAPE} = \frac{100\%}{n} \sum_{t=1}^n \left| \frac{e_t}{y_t} \right|$$



Linear Regression

Years of Experience	Salary in 1000\$
2	15
3	28
5	42
13	64
8	50
16	90
11	58
1	8
9	54



Linear Regression

Phương trình :

$$h_{\theta} = \theta_0 + \theta_1 x$$

Hàm mất mát:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x_i) - y_i)^2$$

Chúng ta luôn mong muốn rằng sự mất mát (sai số) là nhỏ nhất, điều đó đồng nghĩa với việc tìm vector hệ số sao cho giá trị của hàm mất mát này càng nhỏ càng tốt.

Áp dụng giảm gradient, tính đạo hàm của hàm mất mát theo vector hệ số:

$$\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{\partial}{\partial \theta_0} \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

$$\theta_0: \quad \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})$$

$$\theta_1: \quad \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x^{(i)}$$

Ứng với giá trị learning rate, tính lại vector hệ số sao trong 1 số lần lặp nhất định để tìm giá trị nhỏ nhất cho hàm mất mát.

repeat until convergence {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})$$

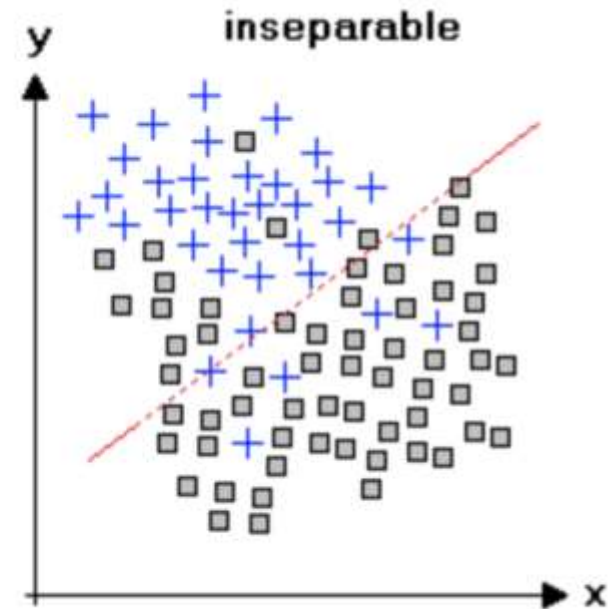
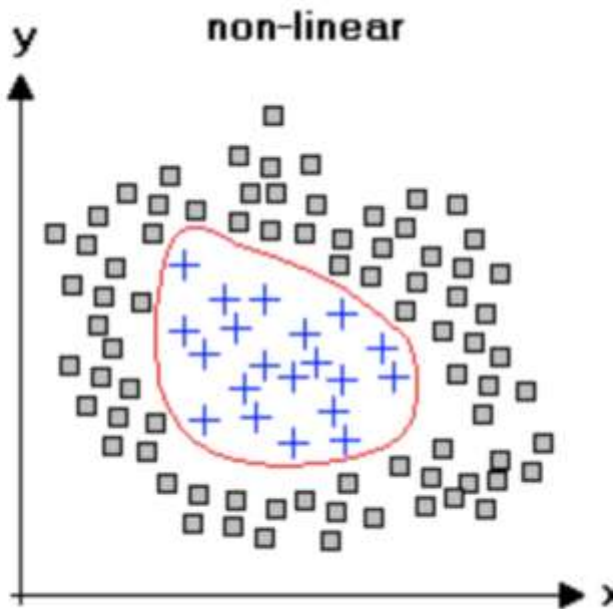
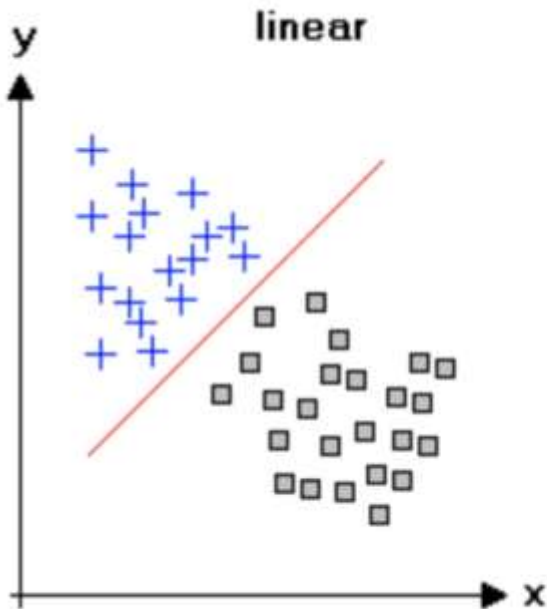
$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) \cdot x^{(i)}$$

}

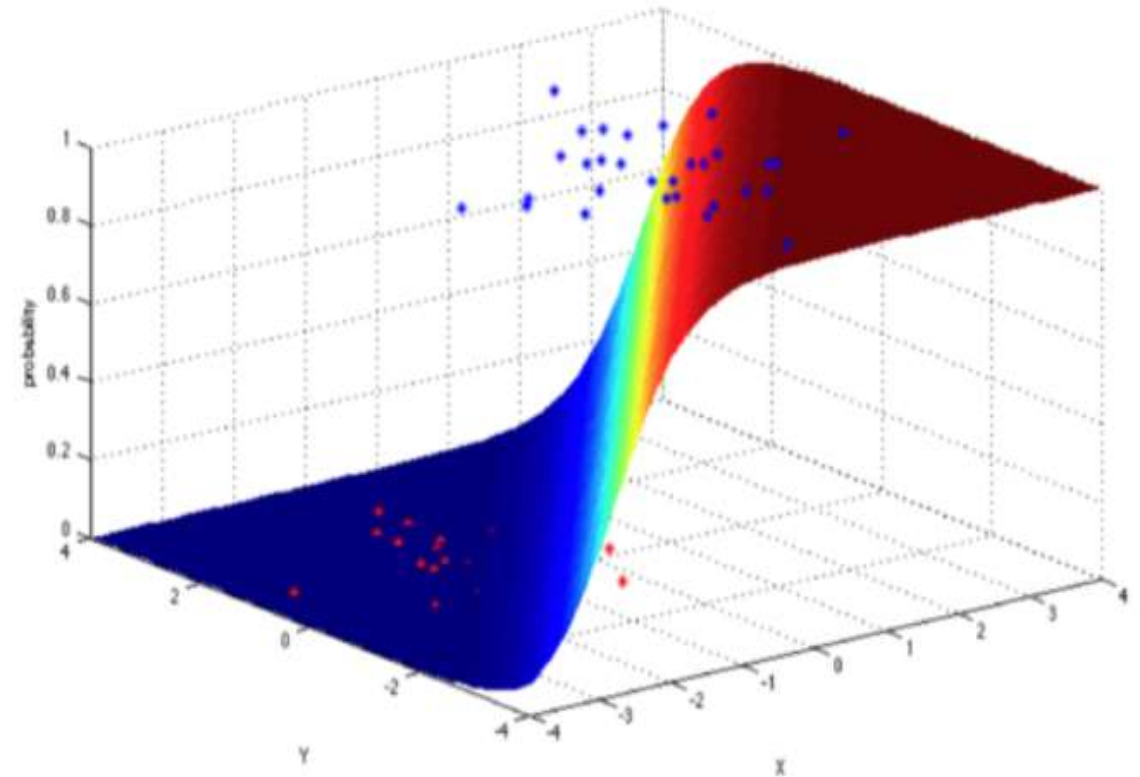
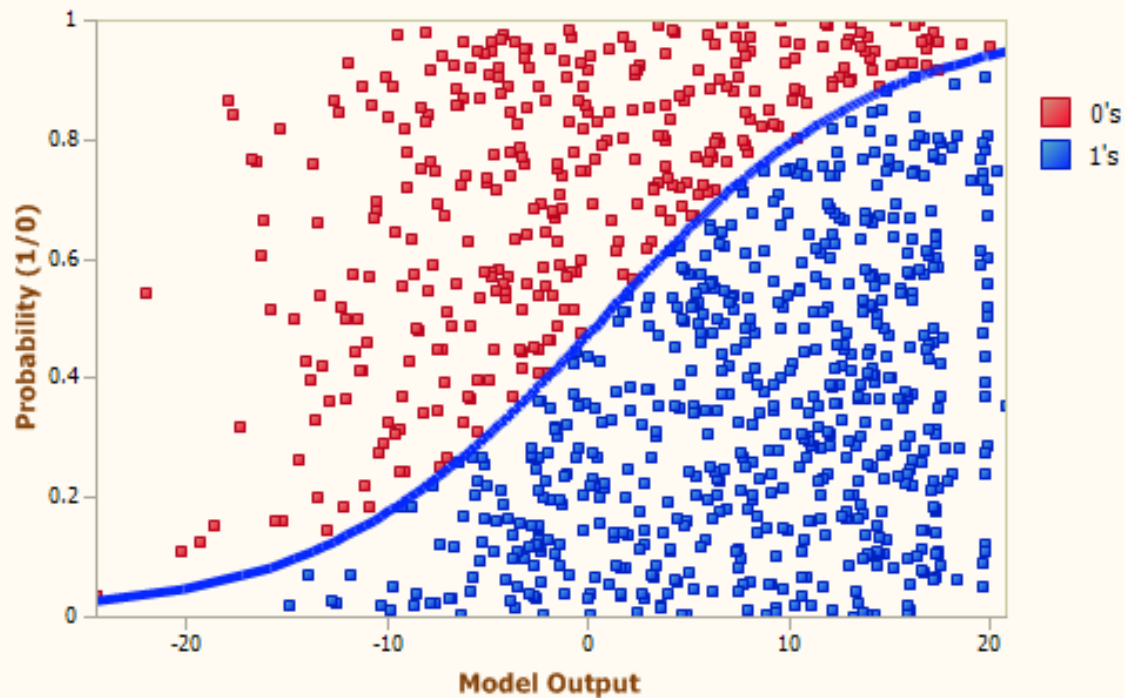
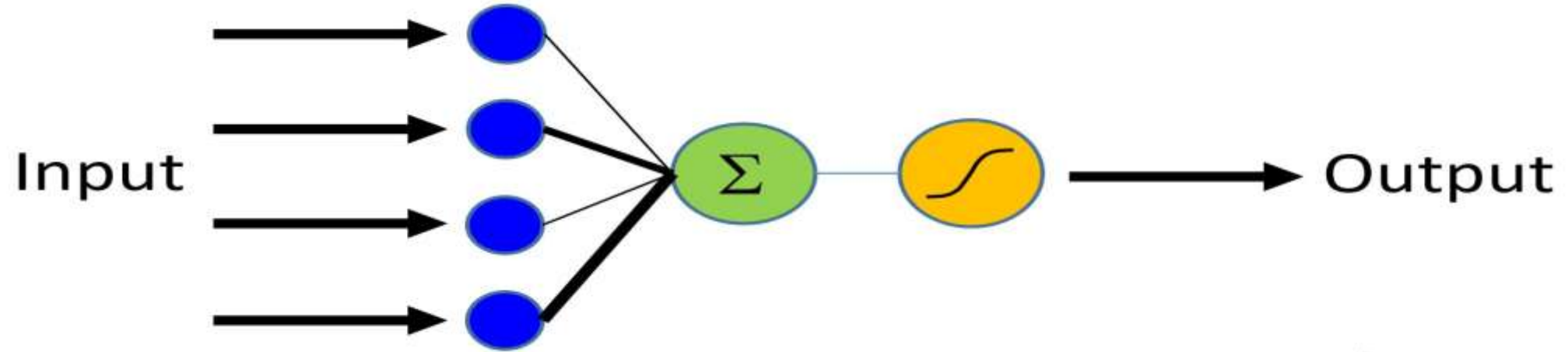
Perceptron một lớp trong bài toán phân loại dữ liệu

Phân chia tuyến tính được(linearly separable)

- Sự phân chia dữ liệu là 1 khái niệm quan trọng trong bài toán phân loại.
- Giả sử chúng ta có điểm dữ liệu 2 chiều. Các lớp dữ liệu được gọi là phân chia tuyến tính được nếu ta có thể vẽ 1 đường thẳng hoặc 1 mặt phẳng phân tách các lớp khác nhau

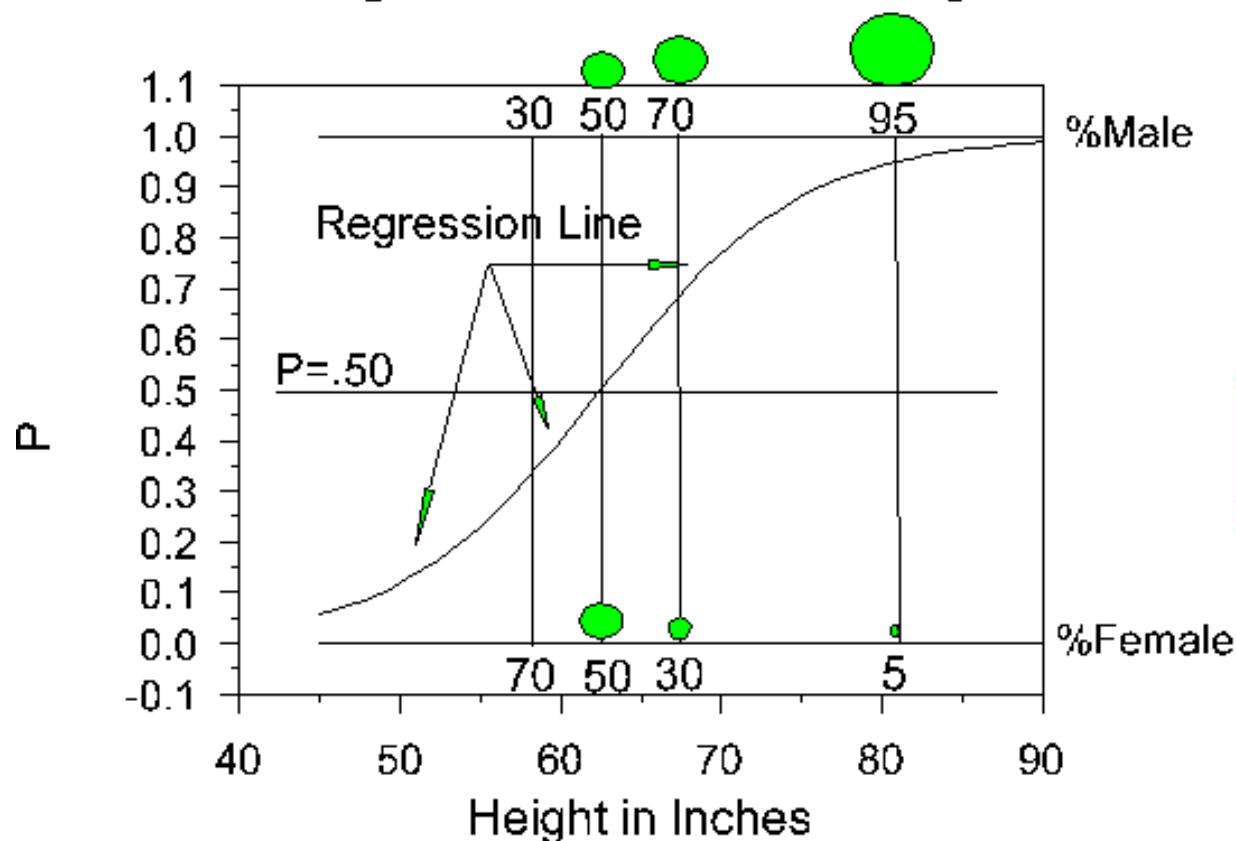


Logistic regression

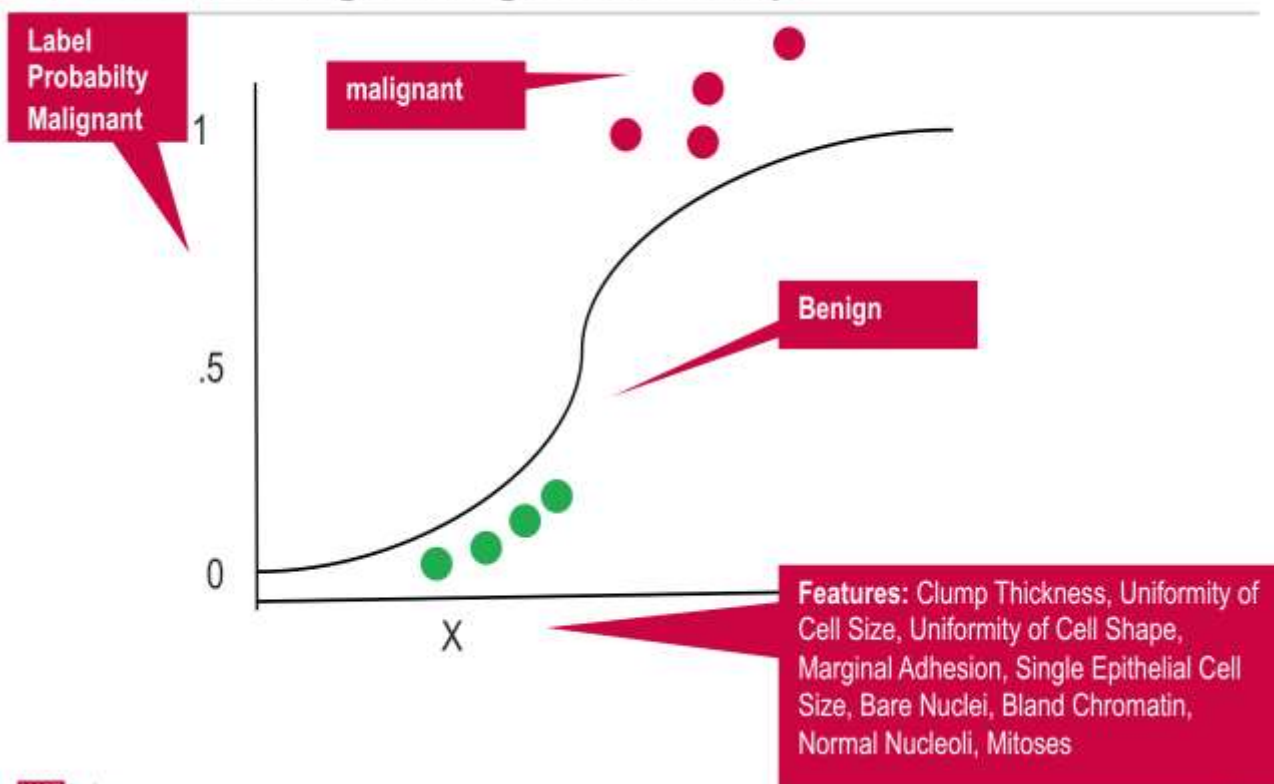


Logistic regression

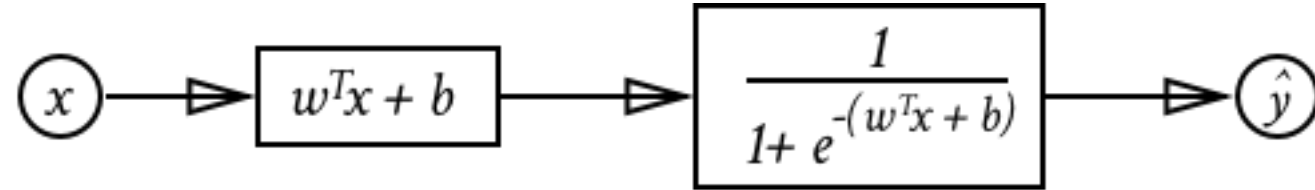
Regression of Sex on Height



Breast Cancer Logistic Regression Example



Logistic regression

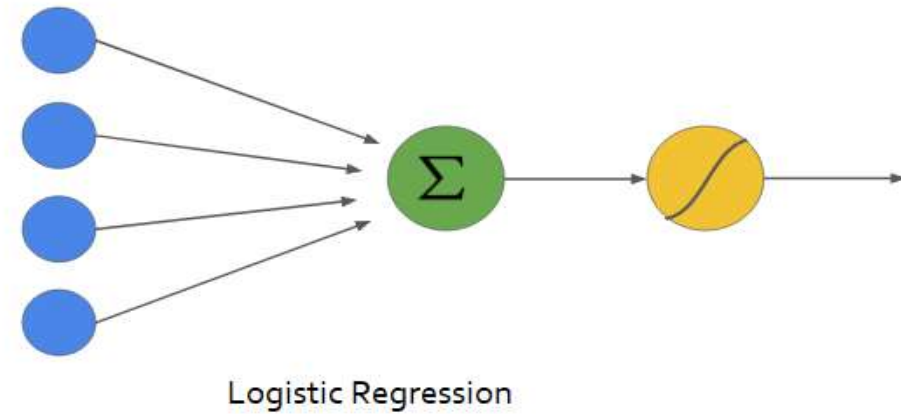
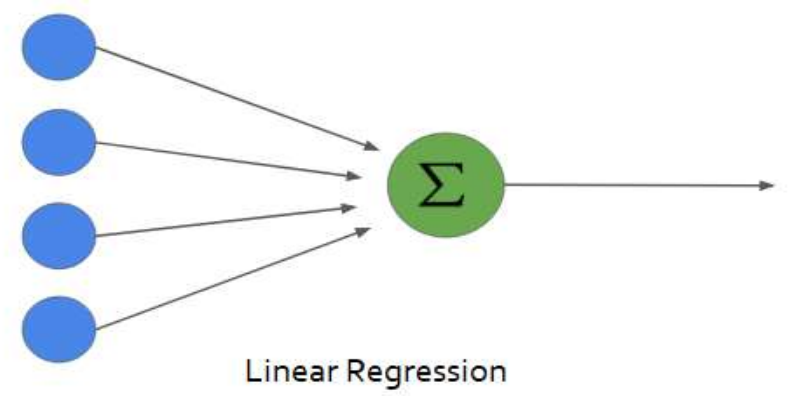


input

linear function,
 $w = \text{weight}$
 $b = \text{bias}$

sigmoid function a
 $a \in (0,1)$

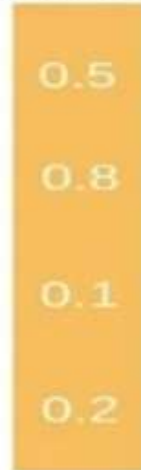
output based on a ,
1 if $a \geq 0.5$
0 if $a < 0.5$



Inputs (X)



Probabilities



Linear Model

sigmoid function

Values between
0 and 1



0
or
1

Cross entropy

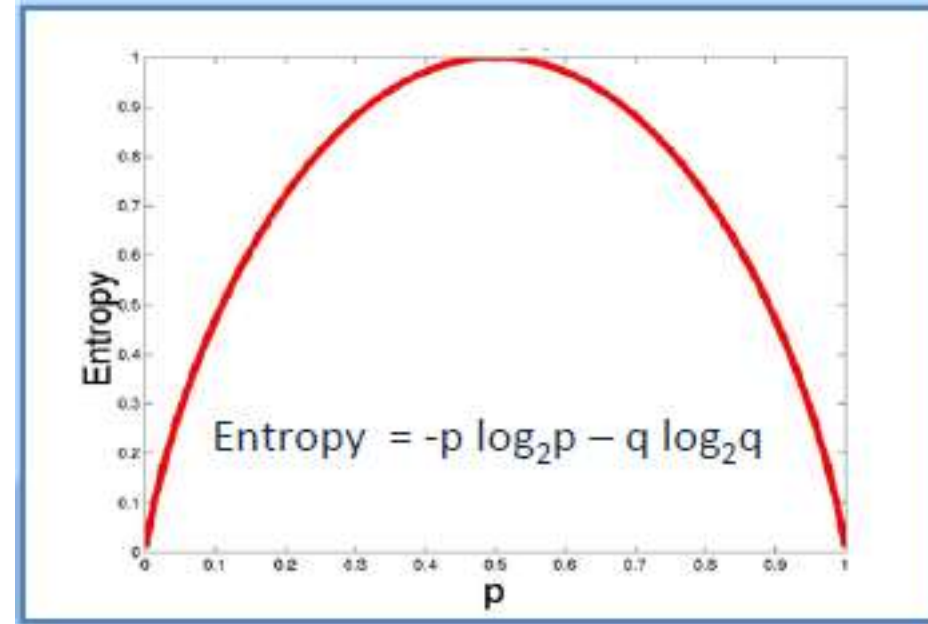
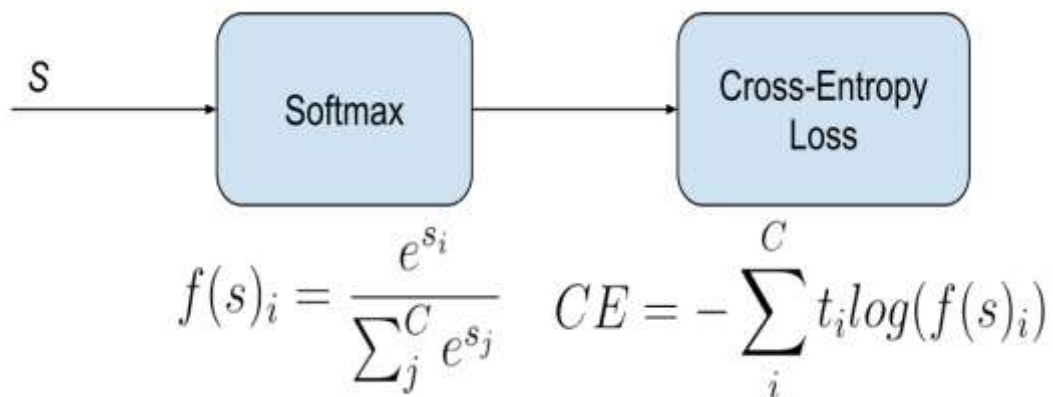
The **Cross-Entropy Loss** is defined as:

$$CE = - \sum_i^C t_i \log(s_i)$$

In a **binary classification problem**, where $C'=2$, the Cross Entropy Loss can be defined also as:

$$CE = - \sum_{i=1}^{C'=2} t_i \log(s_i) = -t_1 \log(s_1) + (1 - t_1) \log(1 - s_1)$$

Categorical Cross-Entropy loss



CROSS - E

$$\text{Entropy} = -0.5 \log_2 0.5 - 0.5 \log_2 0.5 = 1$$

$S(Y)$

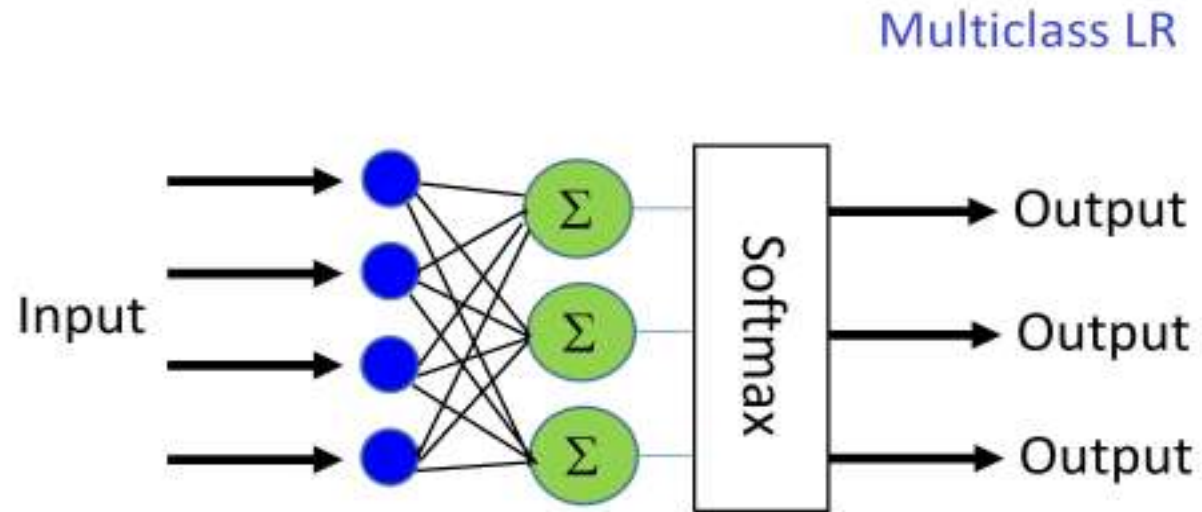
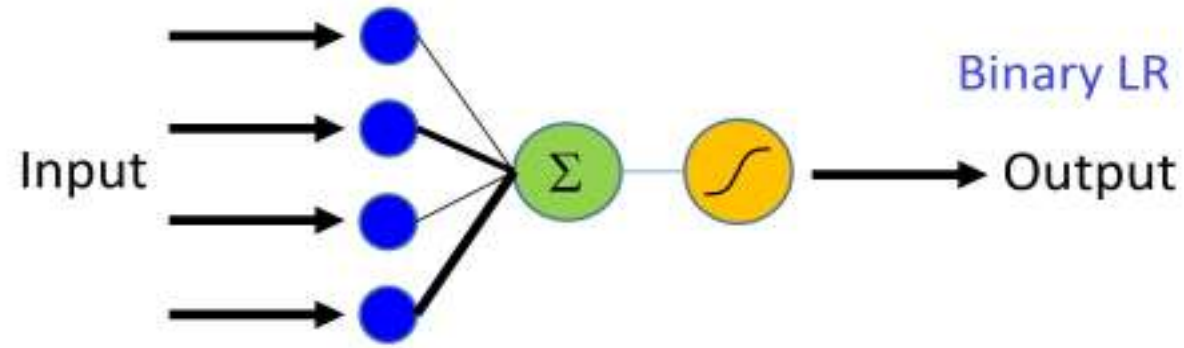
0.7
0.2
0.1

$$D(S, L) = - \sum_i L_i \log(s_i)$$

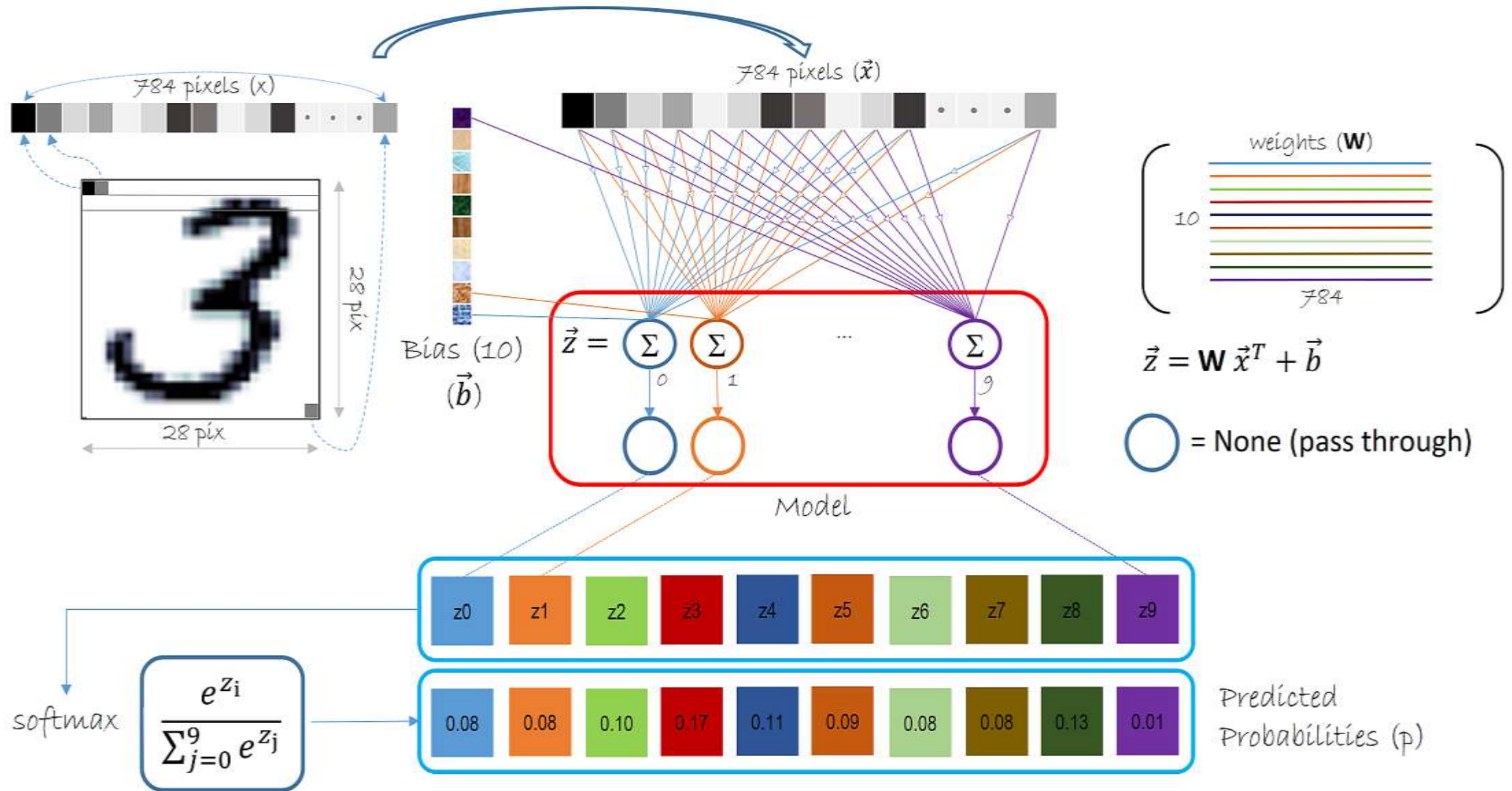
L

1.0
0.0
0.0

Softmax function

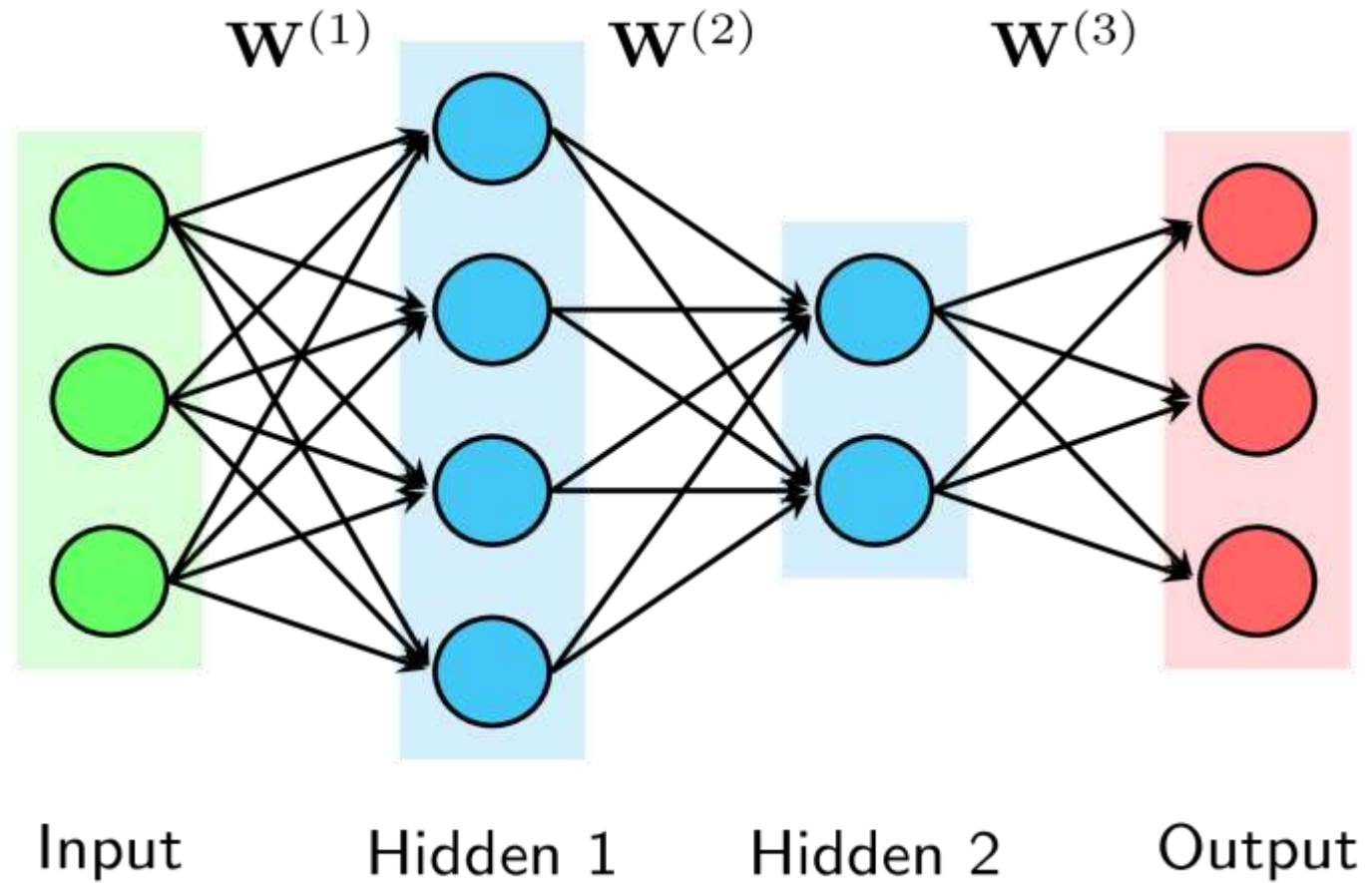


Softmax function



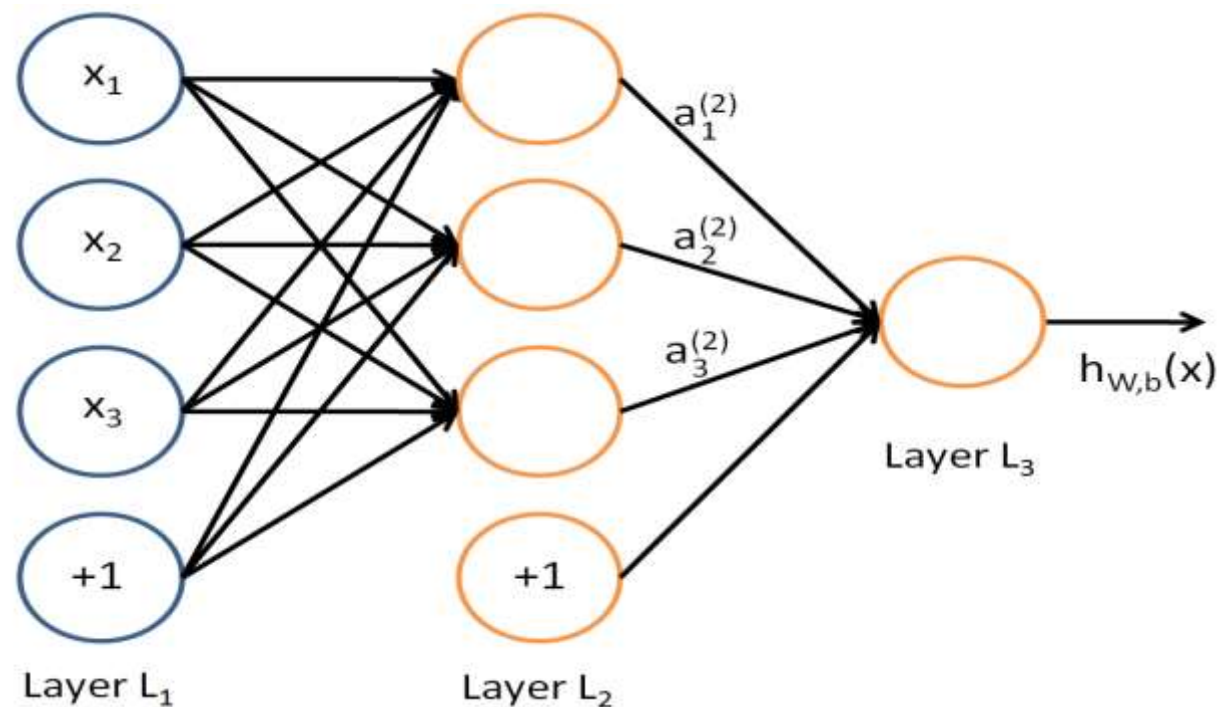
Multi-Layer Neural Network

Layer: *Input layers Output layers và các Hidden layers*
Units:



Multi-Layer Neural Network

Truyền thẳng: (forward propagation)

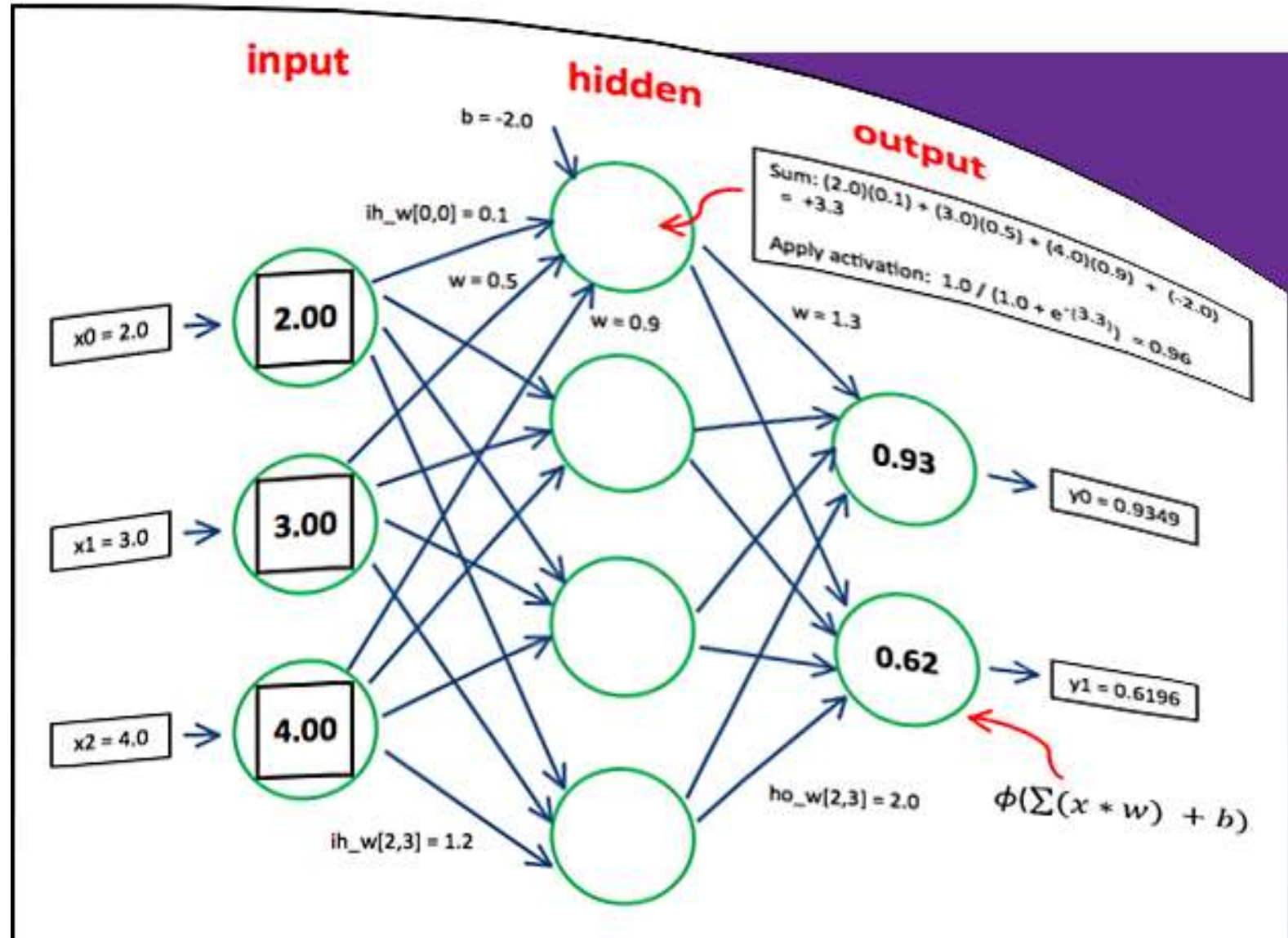
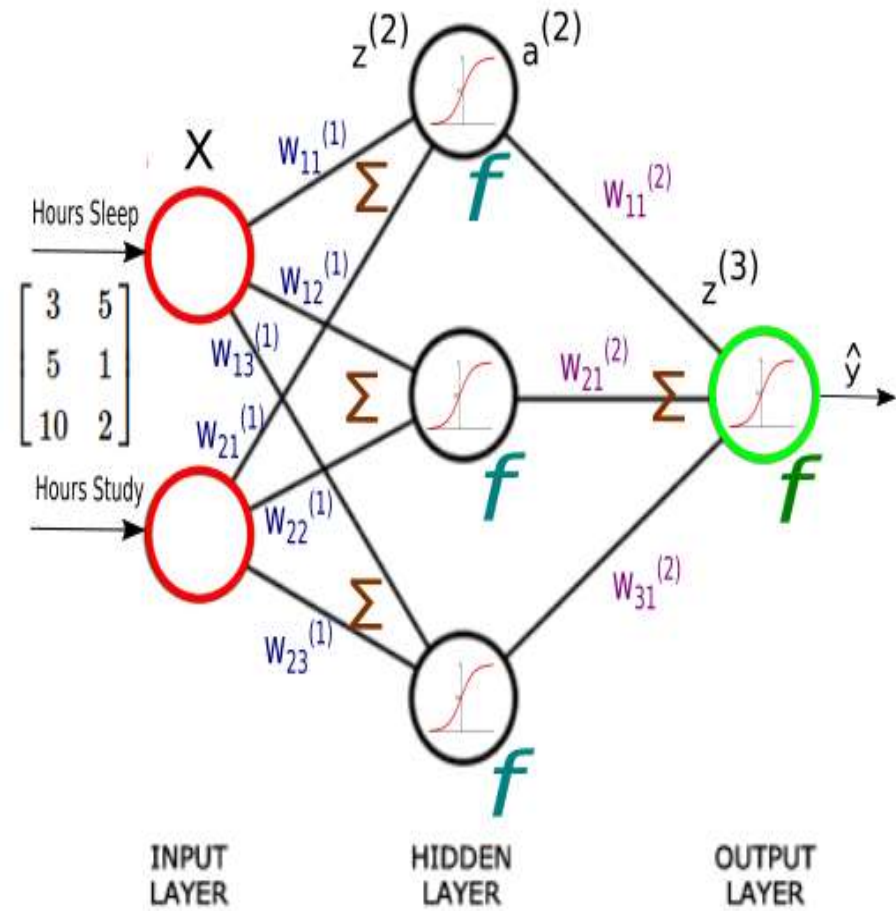


$$\begin{aligned}z^{(2)} &= W^{(1)}x + b^{(1)} \\a^{(2)} &= f(z^{(2)}) \\z^{(3)} &= W^{(2)}a^{(2)} + b^{(2)} \\h_{W,b}(x) &= a^{(3)} = f(z^{(3)})\end{aligned}$$

$$\begin{aligned}z^{(l+1)} &= W^{(l)}a^{(l)} + b^{(l)} \\a^{(l+1)} &= f(z^{(l+1)})\end{aligned}$$

$$\begin{aligned}a_1^{(2)} &= f(W_{11}^{(1)}x_1 + W_{12}^{(1)}x_2 + W_{13}^{(1)}x_3 + b_1^{(1)}) \\a_2^{(2)} &= f(W_{21}^{(1)}x_1 + W_{22}^{(1)}x_2 + W_{23}^{(1)}x_3 + b_2^{(1)}) \\a_3^{(2)} &= f(W_{31}^{(1)}x_1 + W_{32}^{(1)}x_2 + W_{33}^{(1)}x_3 + b_3^{(1)}) \\h_{W,b}(x) &= a_1^{(3)} = f(W_{11}^{(2)}a_1^{(2)} + W_{12}^{(2)}a_2^{(2)} + W_{13}^{(2)}a_3^{(2)} + b_1^{(2)})\end{aligned}$$

Neural network



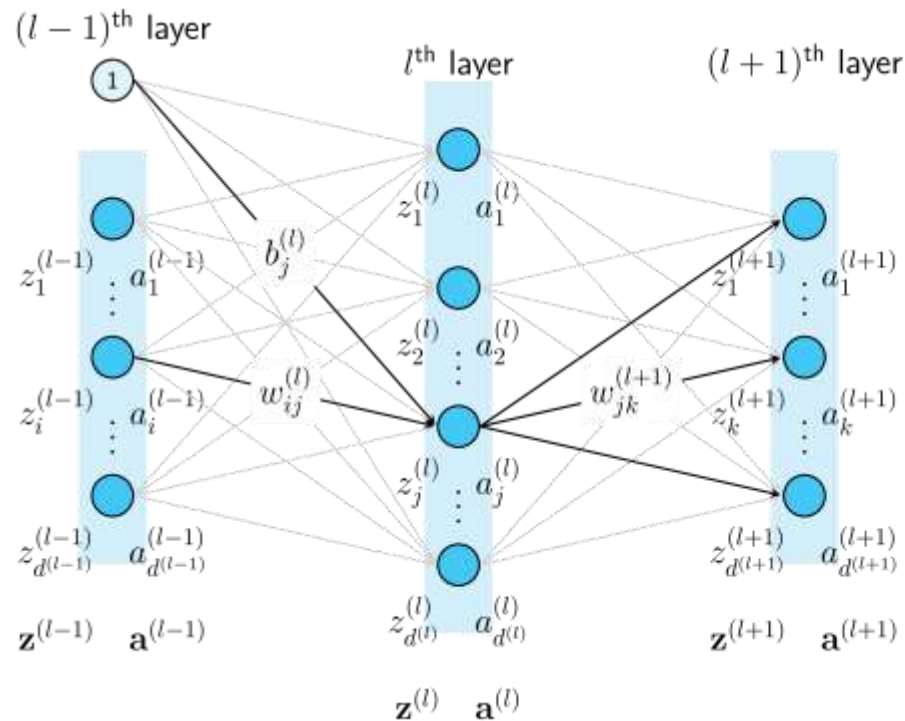
Multi-Layer Neural Network

- Giả sử $J(W,b,X,Y)$ là một hàm mất mát của bài toán, để áp dụng Gradient Descent, chúng ta cần tính được:

$$\frac{\partial J}{\partial \mathbf{W}^{(l)}}; \frac{\partial J}{\partial \mathbf{b}^{(l)}}, \quad l = 1, 2, \dots, L$$

Multi-Layer Neural Network

Lan truyền ngược (Backpropagation)



$$\mathbf{W}^{(l)} \in \mathbb{R}^{d^{(l-1)} \times d^{(l)}}$$

$$\mathbf{b}^{(l)} \in \mathbb{R}^{d^{(l)} \times 1}$$

$$z_j^{(l)} = \mathbf{w}_j^{(l)T} \mathbf{a}^{(l-1)} + b_j^{(l)}$$

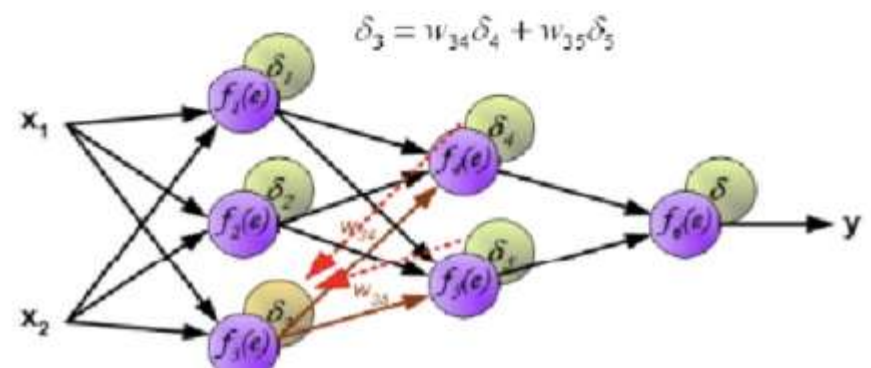
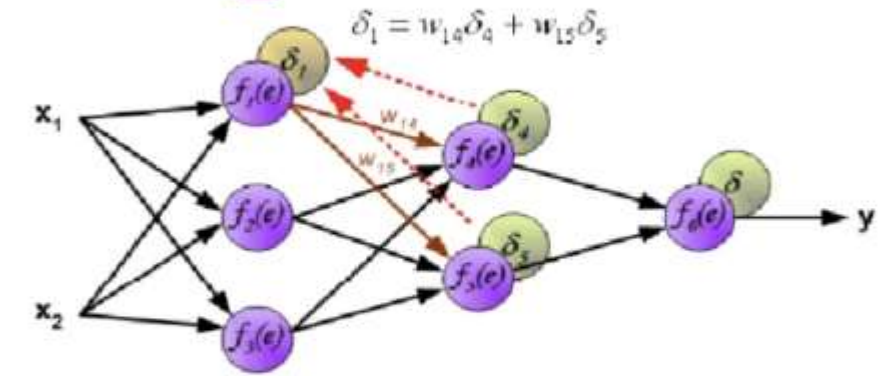
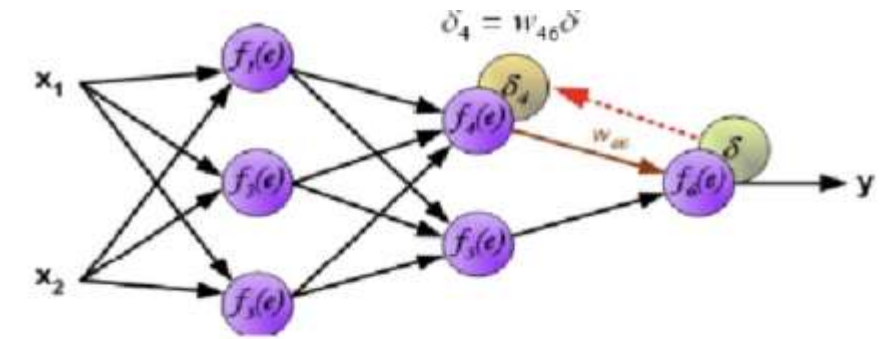
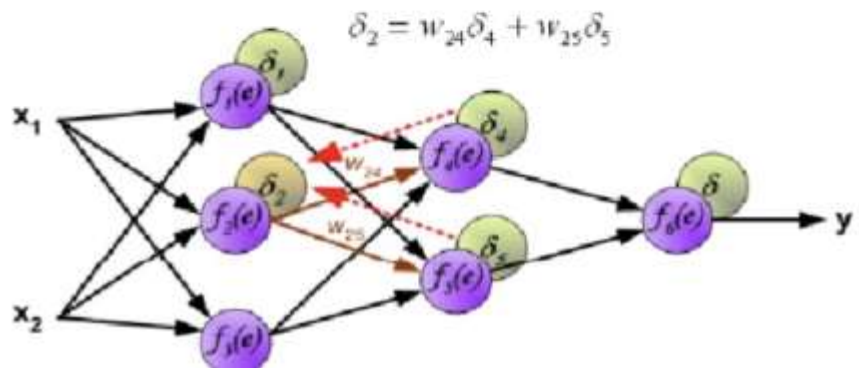
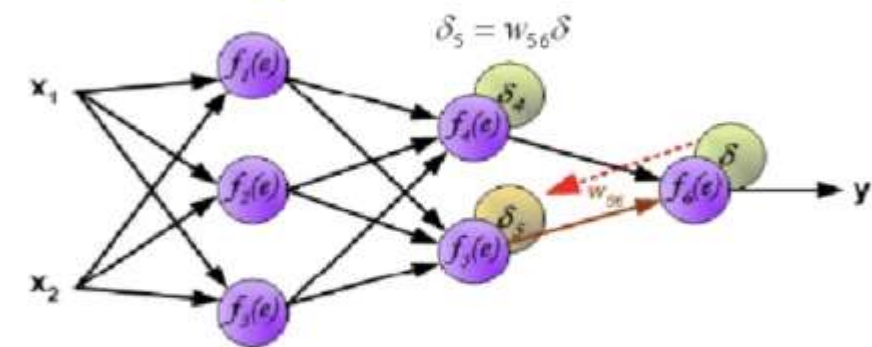
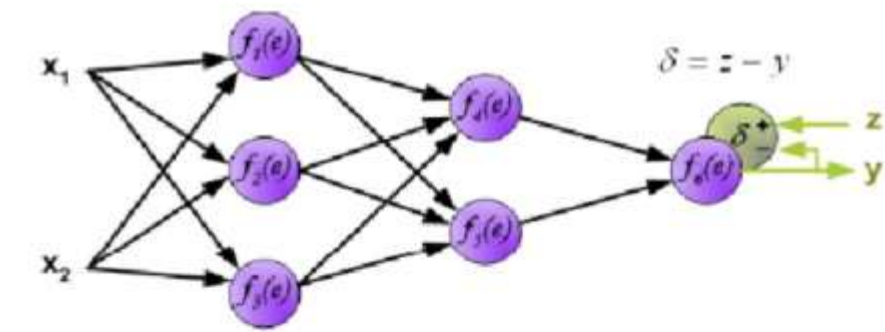
$$\mathbf{z}^{(l)} = \mathbf{W}^{(l)T} \mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}$$

$$\mathbf{a}^{(l)} = f(\mathbf{z}^{(l)})$$

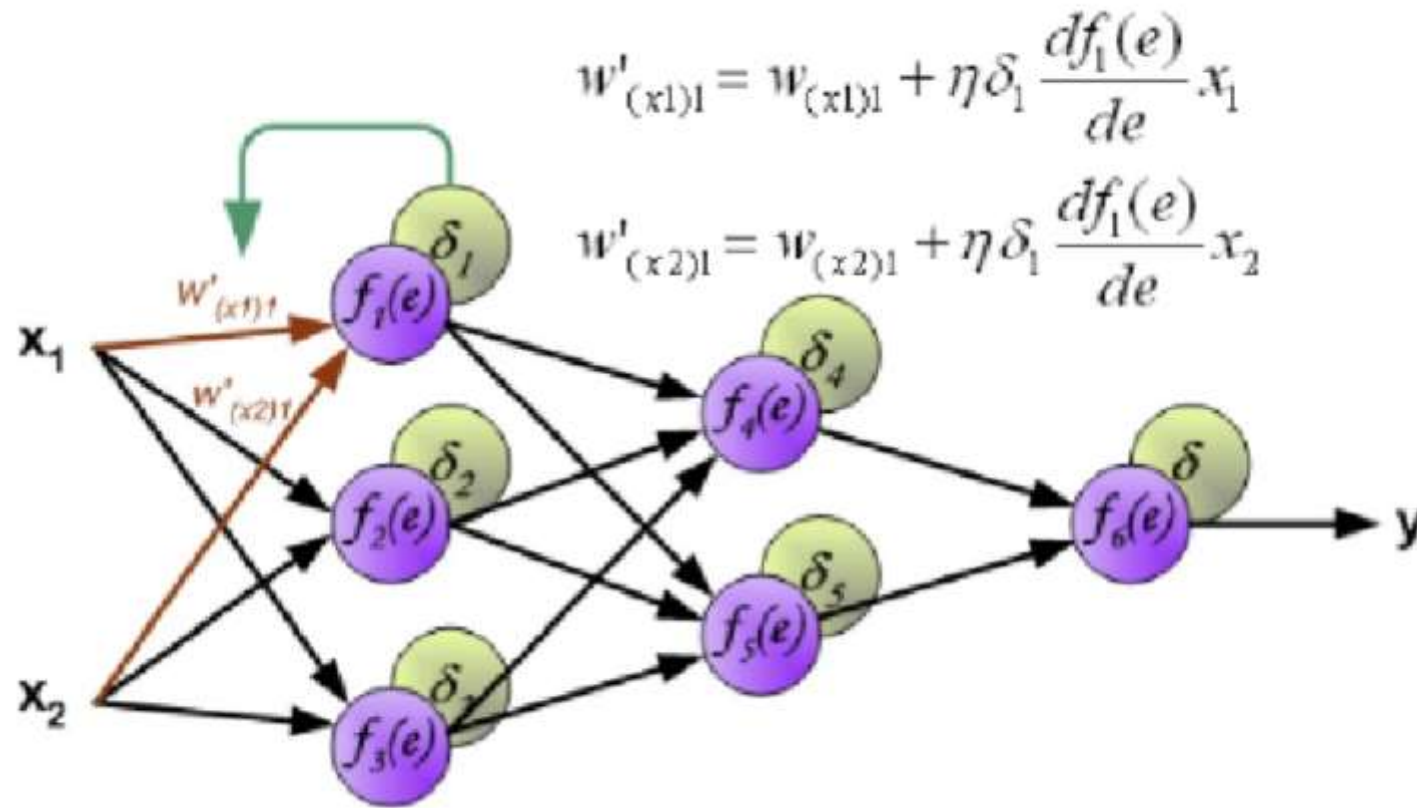
$$\begin{aligned} \frac{\partial J}{\partial w_{ij}^{(l)}} &= \frac{\partial J}{\partial z_j^{(l)}} \cdot \frac{\partial z_j^{(l)}}{\partial w_{ij}^{(l)}} \\ &= e_j^{(l)} a_i^{(l-1)} \end{aligned}$$

$$\begin{aligned} e_j^{(l)} &= \frac{\partial J}{\partial z_j^{(l)}} = \frac{\partial J}{\partial a_j^{(l)}} \cdot \frac{\partial a_j^{(l)}}{\partial z_j^{(l)}} \\ &= \left(\sum_{k=1}^{d^{(l+1)}} \frac{\partial J}{\partial z_k^{(l+1)}} \cdot \frac{\partial z_k^{(l+1)}}{\partial a_j^{(l)}} \right) f'(z_j^{(l)}) \\ &= \left(\sum_{k=1}^{d^{(l+1)}} e_k^{(l+1)} w_{jk}^{(l+1)} \right) f'(z_j^{(l)}) \\ &= \left(\mathbf{w}_{j:}^{(l+1)} \mathbf{e}^{(l+1)} \right) f'(z_j^{(l)}) \end{aligned}$$

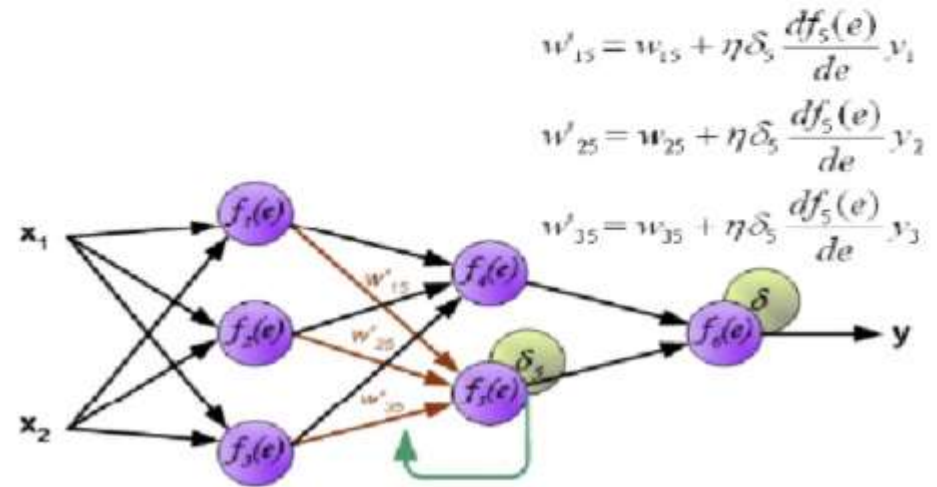
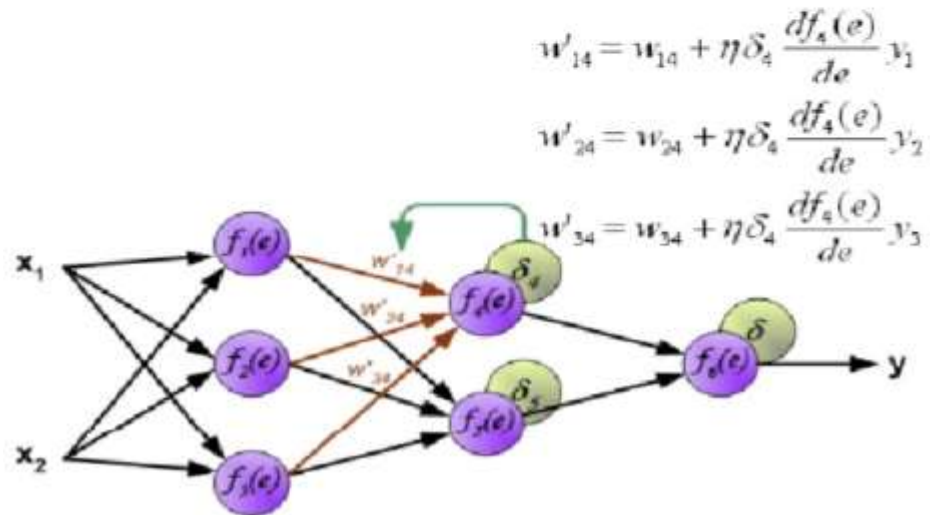
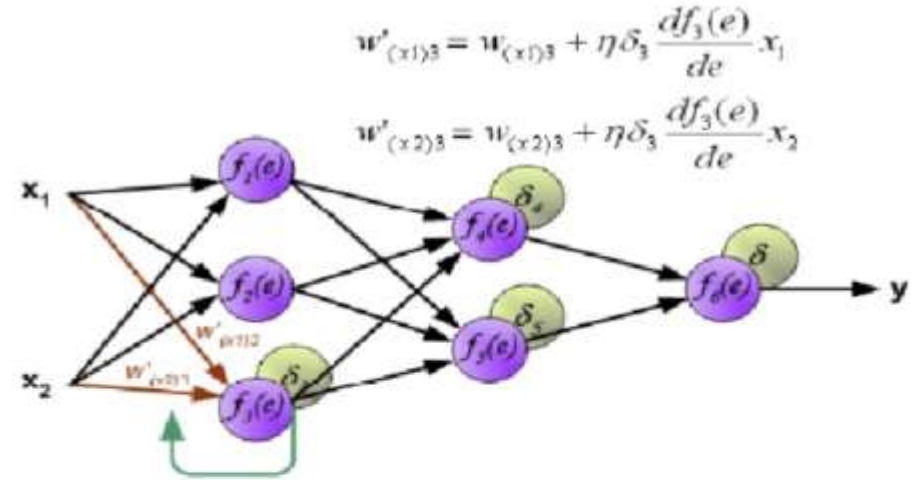
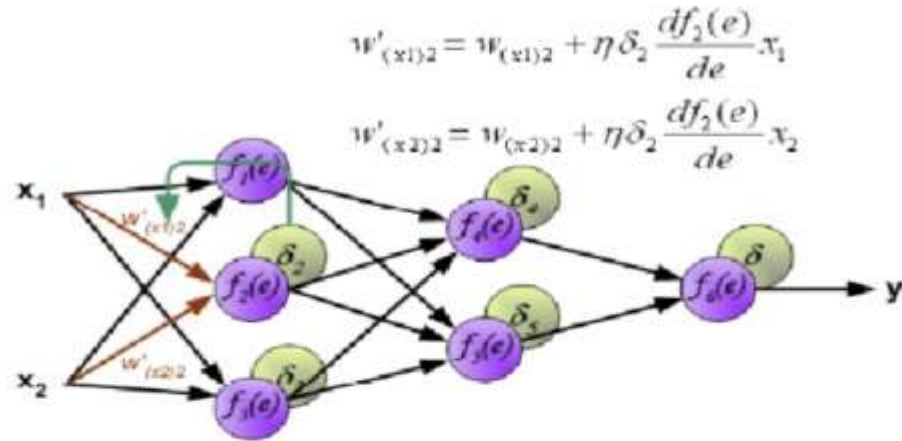
Multi-Layer Neural Network



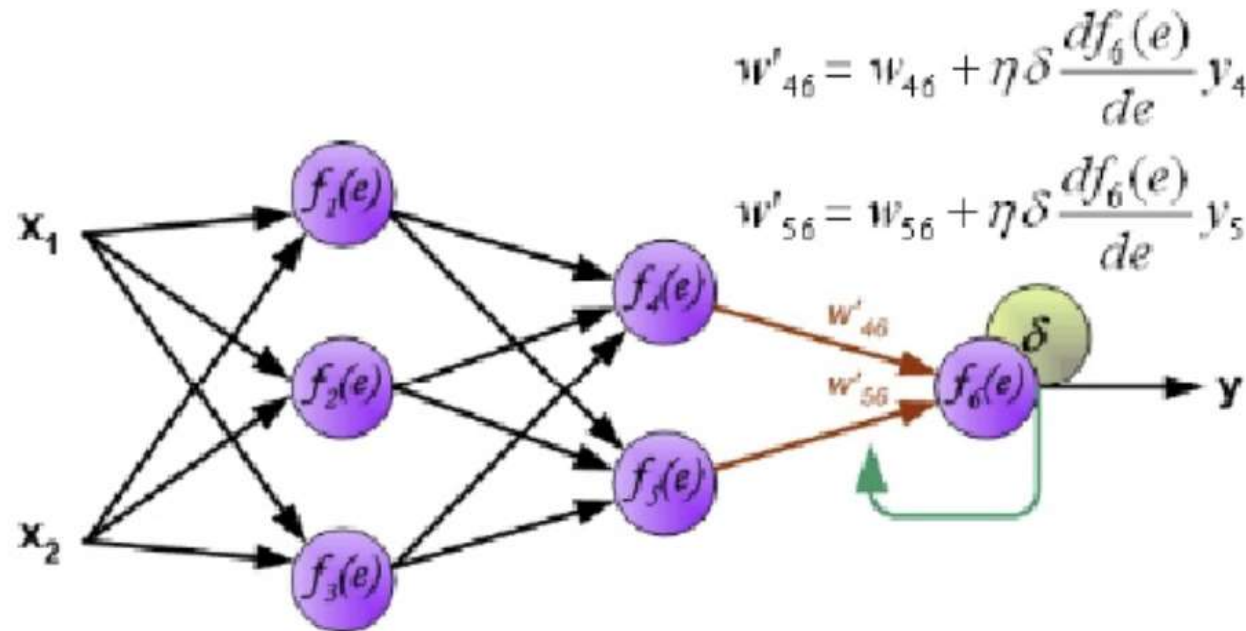
Multi-Layer Neural Network

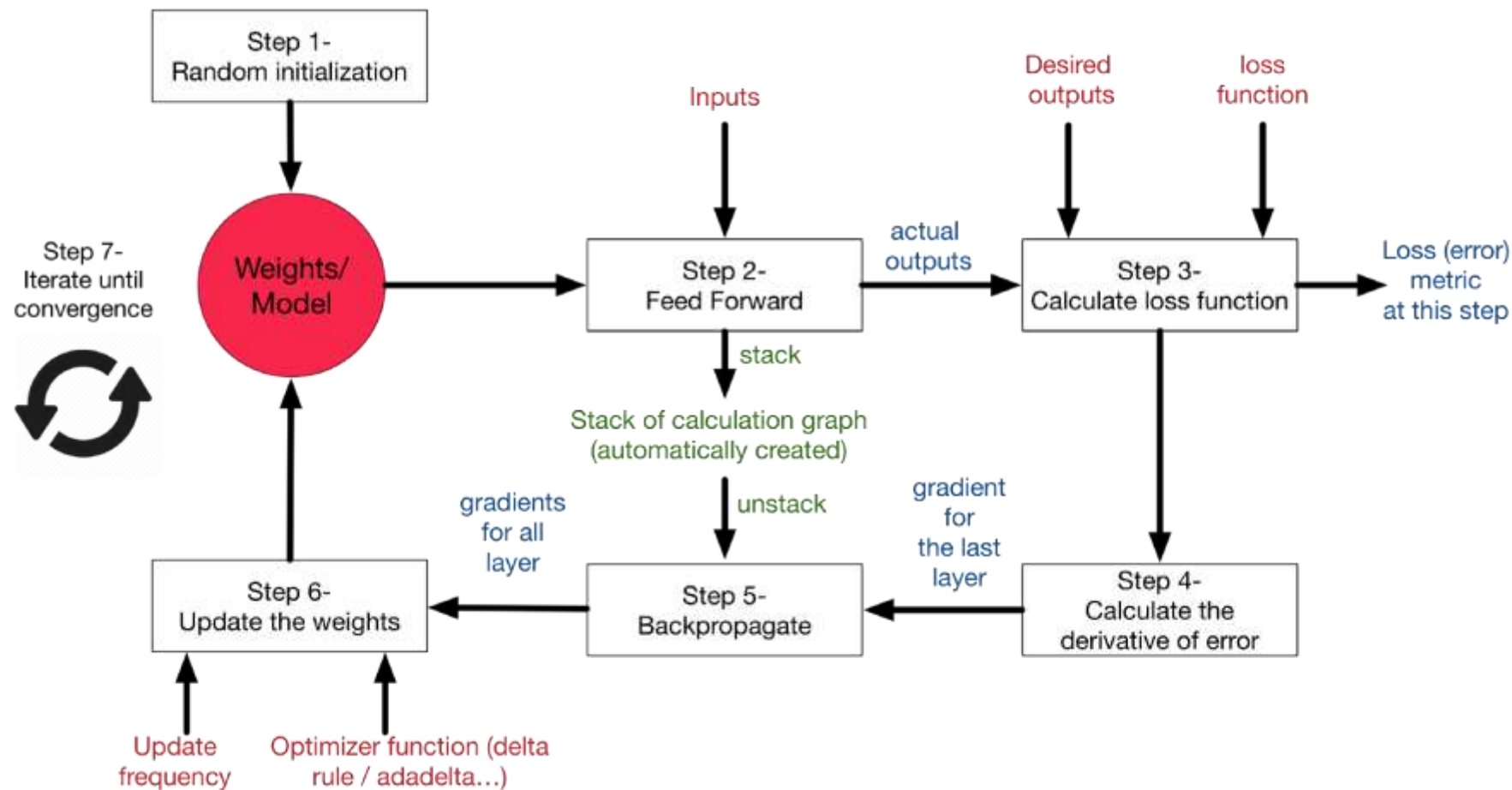


Multi-Layer Neural Network



Multi-Layer Neural Network





Bài toán phân loại chữ viết tay

