ĐẠI HỌC TÔN ĐỨC THẮNG
TON DUC THANG UNIVERSITY

Lecture 2
Basics of Neural Networks

LÊ ANH CƯỜNG
Ton Duc Thang University

Objective of this lecture

- Recap
  - Gradient Descent Algorithm for Linear Regression
  - Gradient Descent for Classification with Logistic Regression
  - Perceptron
- MultiLayer Perceptron (MLP) and Backpropagation

Gradient Descent Algorithm with Linear Regression

Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$

Parameters: θ_0, θ_1

Cost Function: $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$

Goal: $\underset{\theta_0, \theta_1}{\text{minimize}}\ J(\theta_0, \theta_1)$

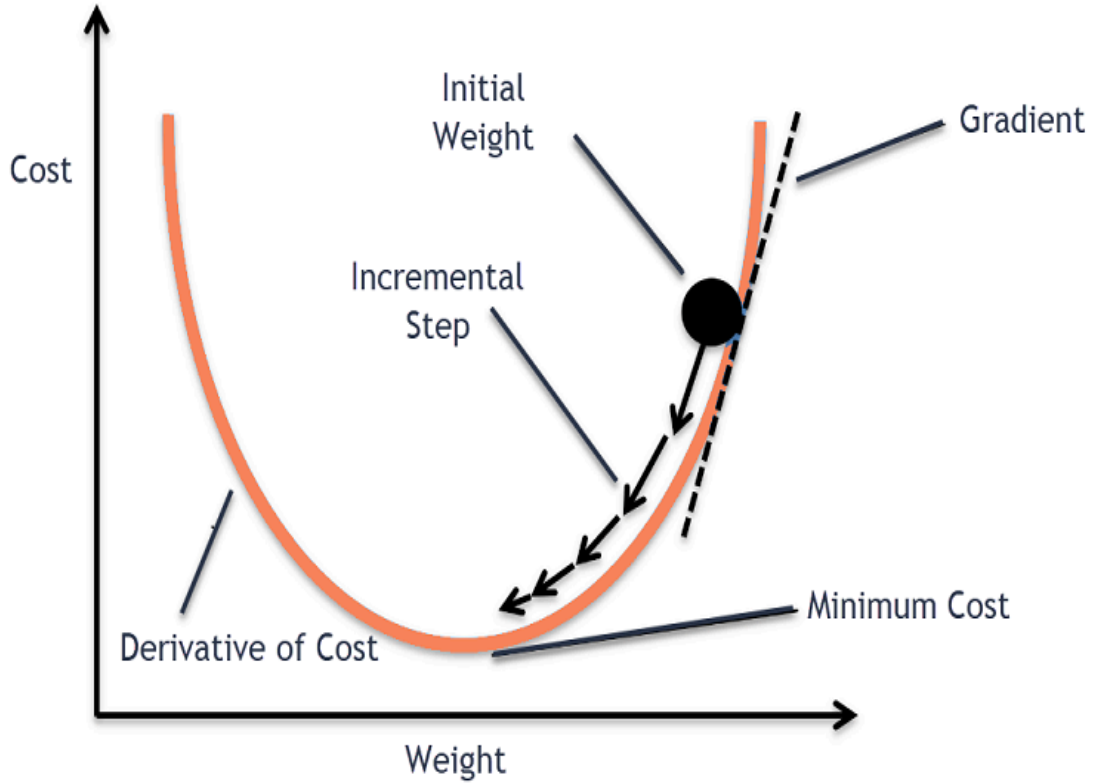
Gradient Descent Alg

Repeat until convergence

{

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

}

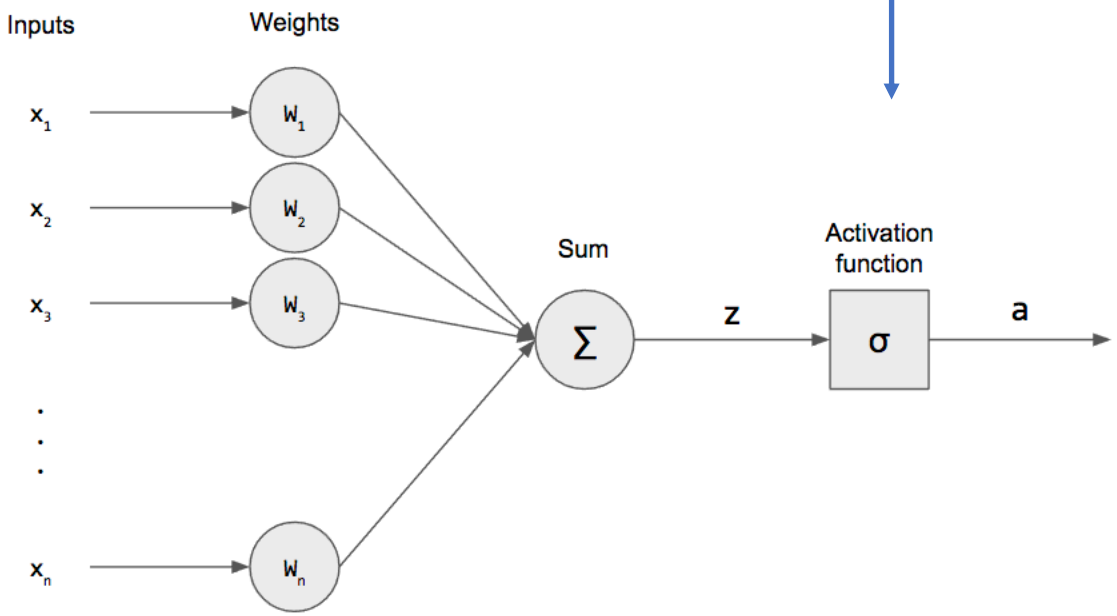


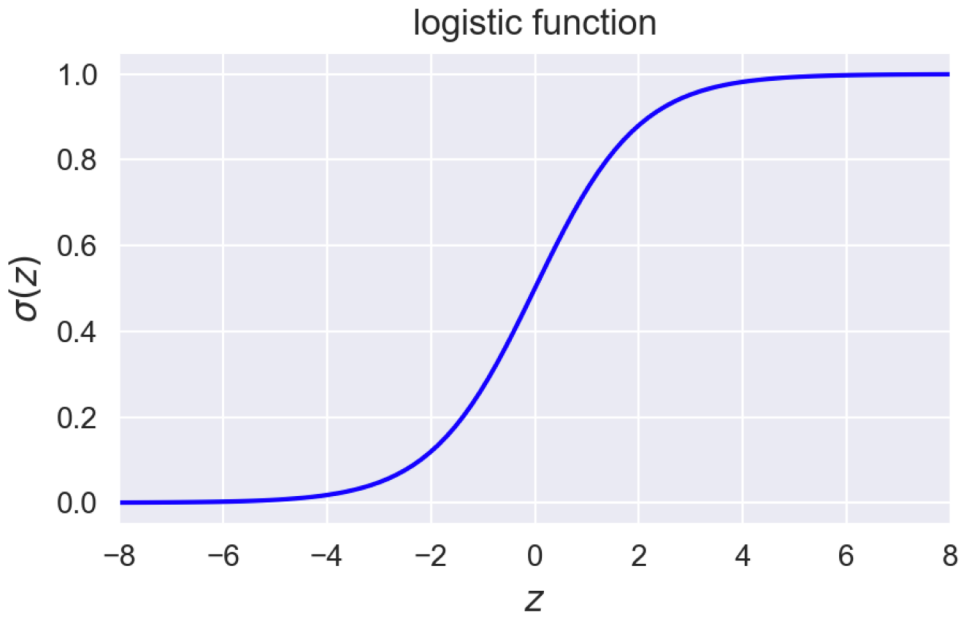
Generalize to multiple variate linear regression?

- exercise for students

Logistic Regression

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





Cross-entropy loss/cost function

The cross entropy formula takes in two distributions, $p(x)$, the true distribution, and $q(x)$, the estimated distribution, defined over the discrete variable x and is given by

$$H(p, q) = H(p) + D_{\mathrm{KL}}(p \| q)$$

$$H(p, q) = -\sum_{\forall x} p(x) \log(q(x))$$

Gradient Calculation

Having set up our notation, $p \in \{y, 1-y\}$ and $q \in \{\hat{y}, 1-\hat{y}\}$, we can use cross entropy to get a measure of dissimilarity between p and q:

$$H(p,q) = -\sum_i p_i \log q_i = -y \log \hat{y} - (1-y) \log(1-\hat{y})$$

$$J(\theta) = -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1-y^{(i)}) \log (1-h_\theta(x^{(i)}))]$$

Gradient Calculation

$$J(\theta) = -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

$$\begin{aligned}
\frac{\partial}{\partial \theta_j} J(\theta) &= -\frac{1}{m} \sum_{i=1}^{m} y^{(i)} \frac{x_j^{(i)}}{h(x^{(i)})} h(x^{(i)})(1 - h(x^{(i)})) + (1 - y^{(i)}) \frac{x_j^{(i)}}{1 - h(x^{(i)})} (-h(x^{(i)})(1 - h(x^{(i)}))) \\
&= -\frac{1}{m} \sum_{i=1}^{m} (y^{(i)} (1 - h(x^{(i)})) - (1 - y^{(i)}) h(x^{(i)})) x_j^{(i)} \\
&= -\frac{1}{m} \sum_{i=1}^{m} \left(y^{(i)} - y^{(i)} h(x^{(i)}) - h(x^{(i)}) + y^{(i)} h(x^{(i)}) \right) x_j^{(i)} \\
&= \frac{1}{m} \sum_{i=1}^{m} \left(h(x^{(i)}) - y^{(i)} \right) x_j^{(i)}
\end{aligned}$$

Gradient Descent Alg for Logistic Regression

$$J(\theta) = -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

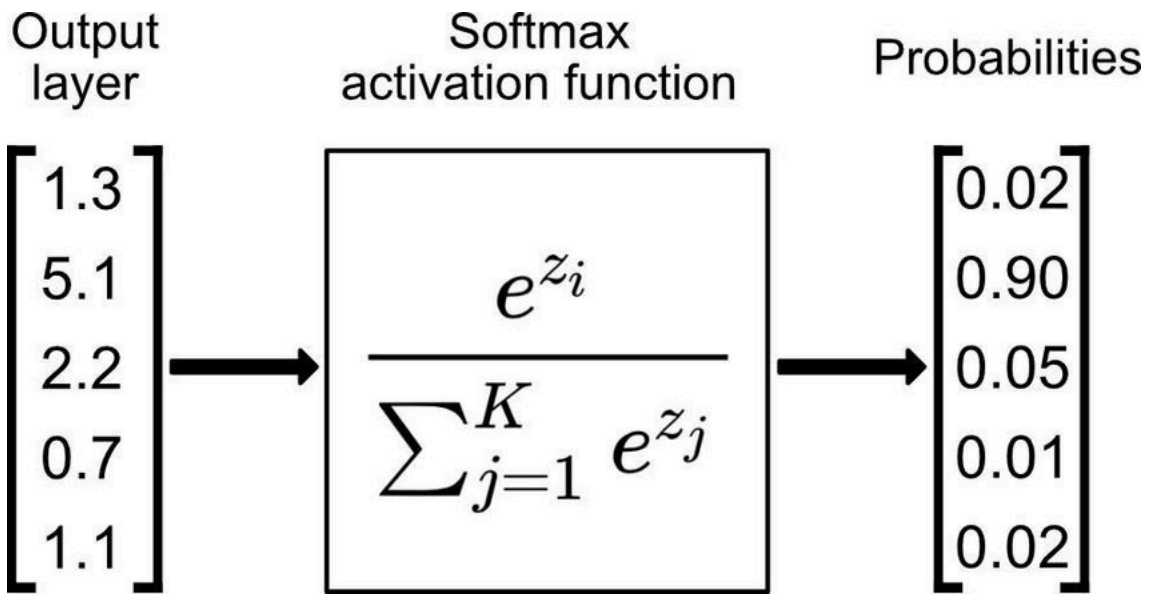
Want $\min_\theta J(\theta)$:

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

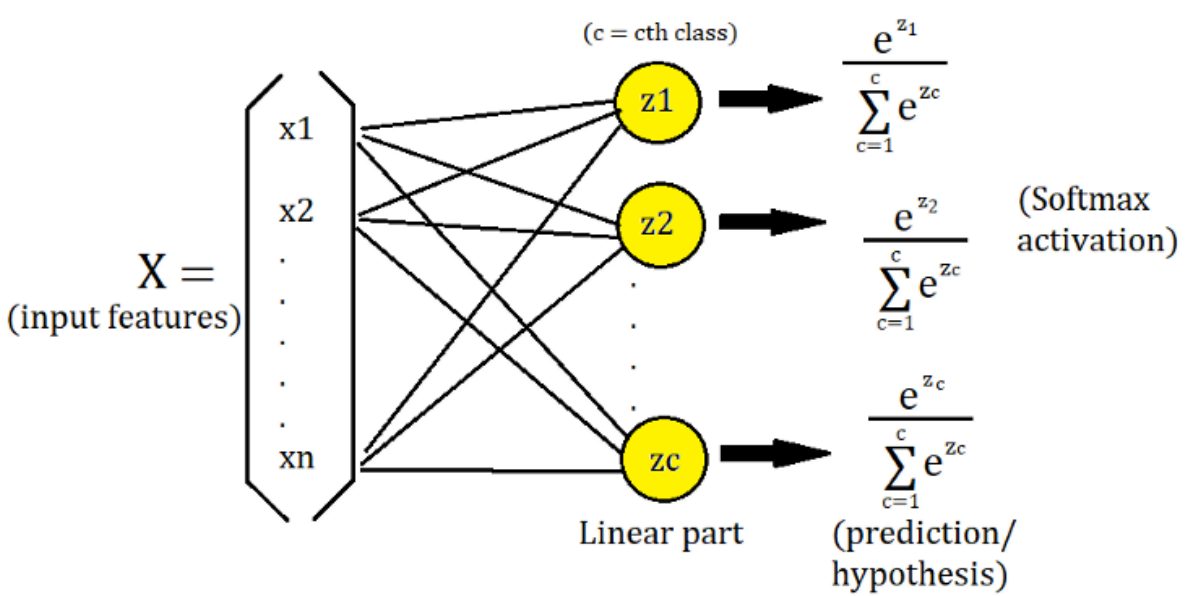
Softmax Functions

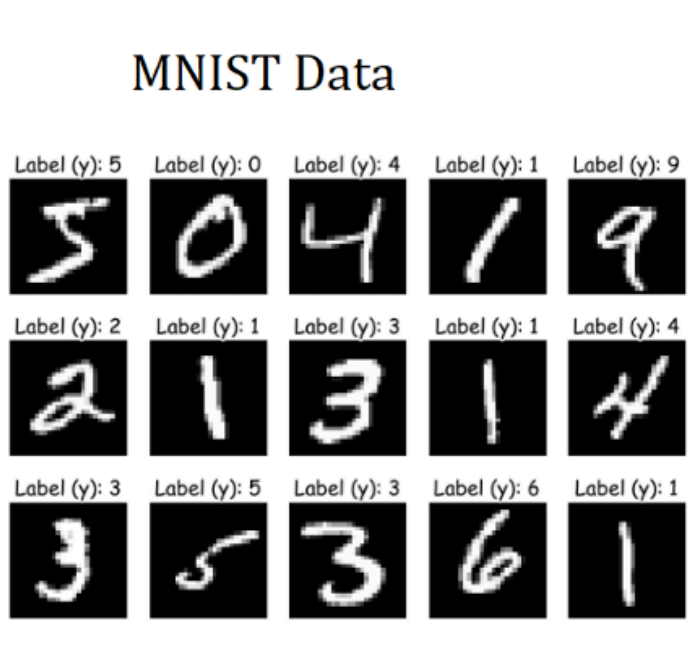
$$\text{Softmax } \sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

$$\text{Sigmoid } S(x) = \frac{1}{1+e^{-x}}$$

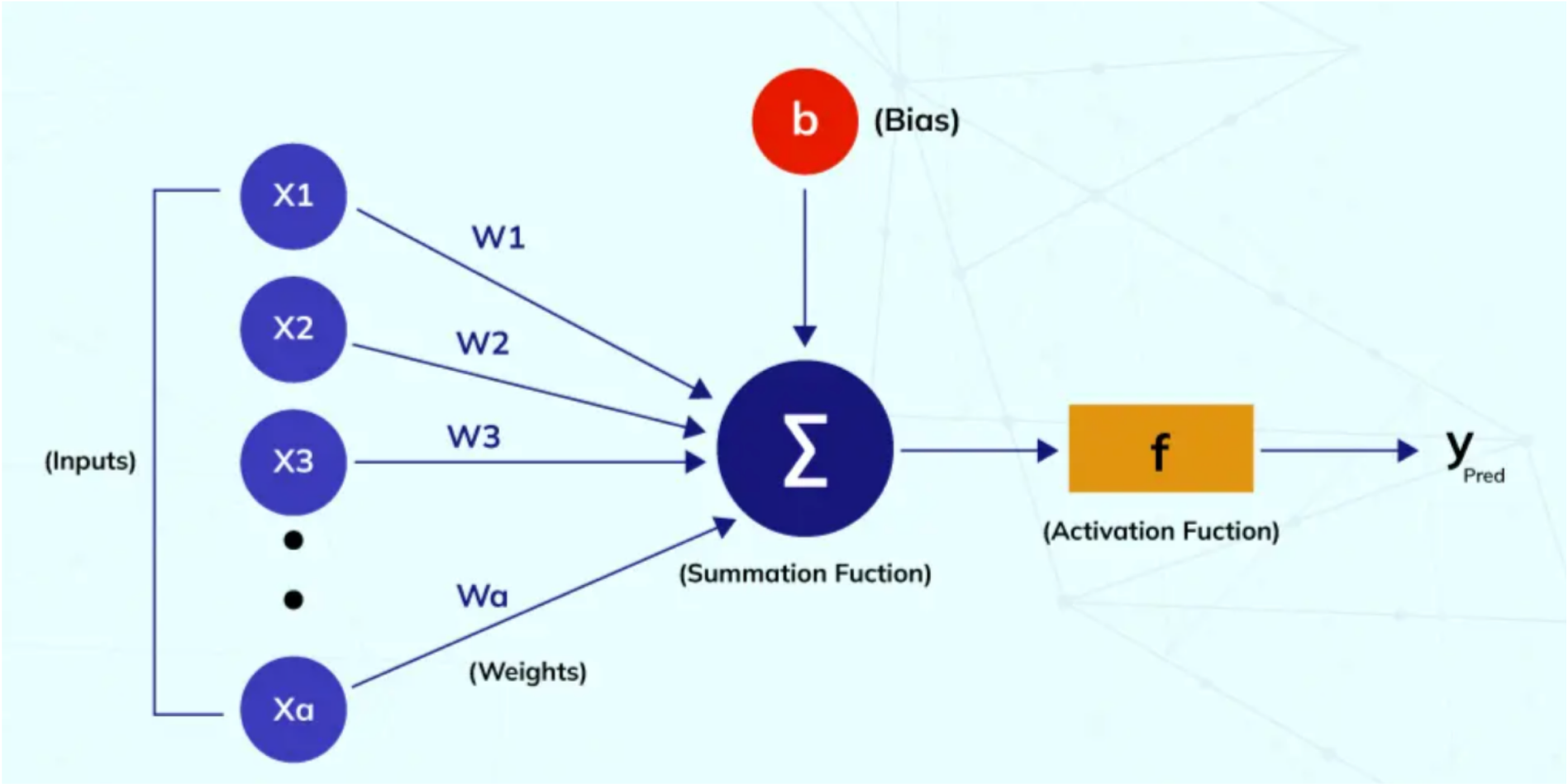


Softmax Function for Multiclass Classification





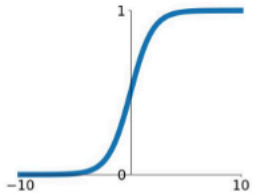
Perceptron Architecture



Common Activation Functions

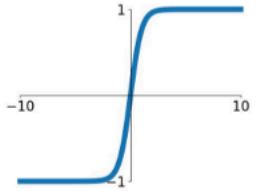
Sigmoid

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



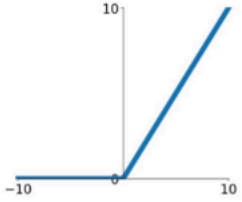
tanh

$$\tanh(x)$$



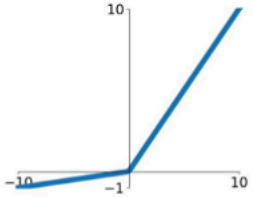
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

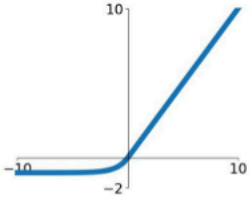


Maxout

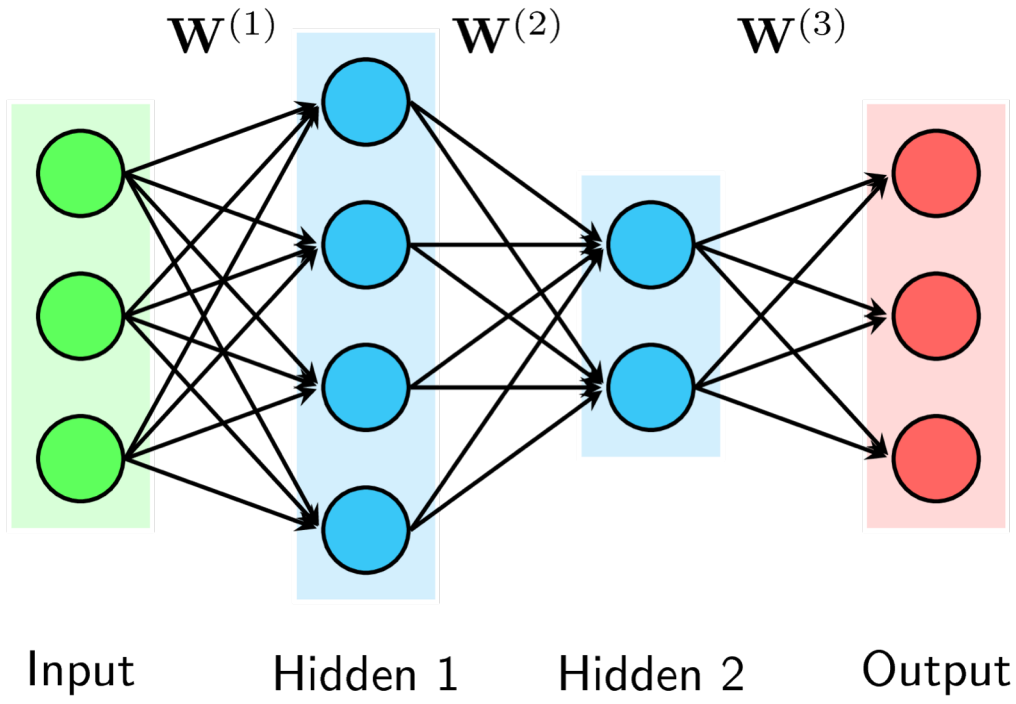
$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



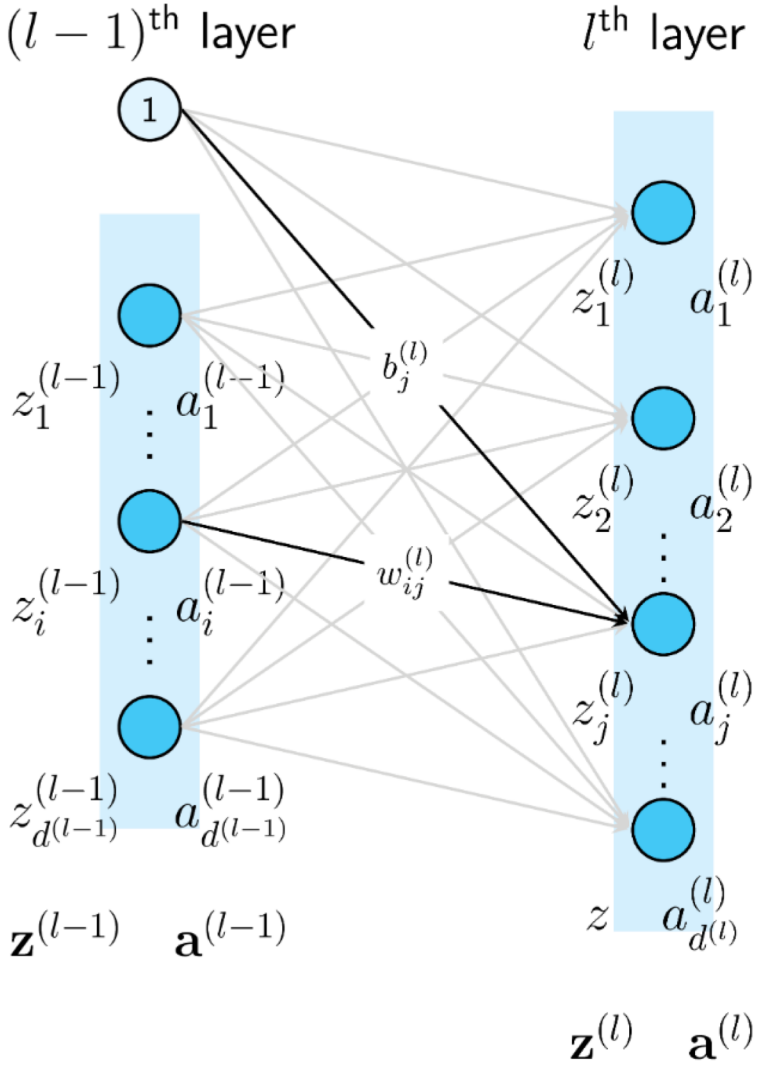
Architecture of MultiLayer Perceptron



Hình 3: MLP với hai hidden layers (các biases đã bị ẩn).
(*From Machinelearningcoban.com*)

Backpropagation

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



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

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

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

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

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