Deep Neural Networks (DNN)

Dr.Varodom Toochinda

Dept. of Mechanical Engineering

Kasetsart University

Outline

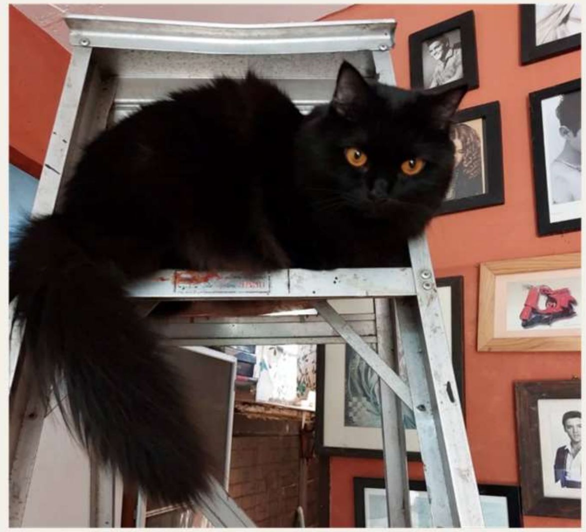
- Logistic regression

Logistic regression
(การถดถอยลอจิสติก)

$$X = \begin{bmatrix} \vdots & \vdots & \vdots & \vdots \\ X^{(1)} & X^{(2)} & \dots & X^{(m)} \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

$$Y = \begin{bmatrix} y^{(1)} & y^{(2)} & \dots & y^{(m)} \end{bmatrix}$$

64



64

binary classification

Outcome

$$Y = \begin{cases} 1 : \text{cat} \\ 0 : \text{no cat} \end{cases}$$

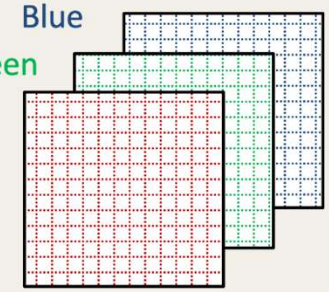
Input vector

Blue

Green

Red

64



64

$$X = \left.\begin{bmatrix} x_1 \\ x_2 \\ : \\ : \\ x_n \end{bmatrix} \begin{matrix} \} \text{Red} \\ \\ \} \text{Green} \\ \\ \} \text{Blue} \end{matrix} \right\} n_x$$

$n = n_x = 64 \times 64 \times 3 = 12{,}288$ elements

Sigmoid activation function

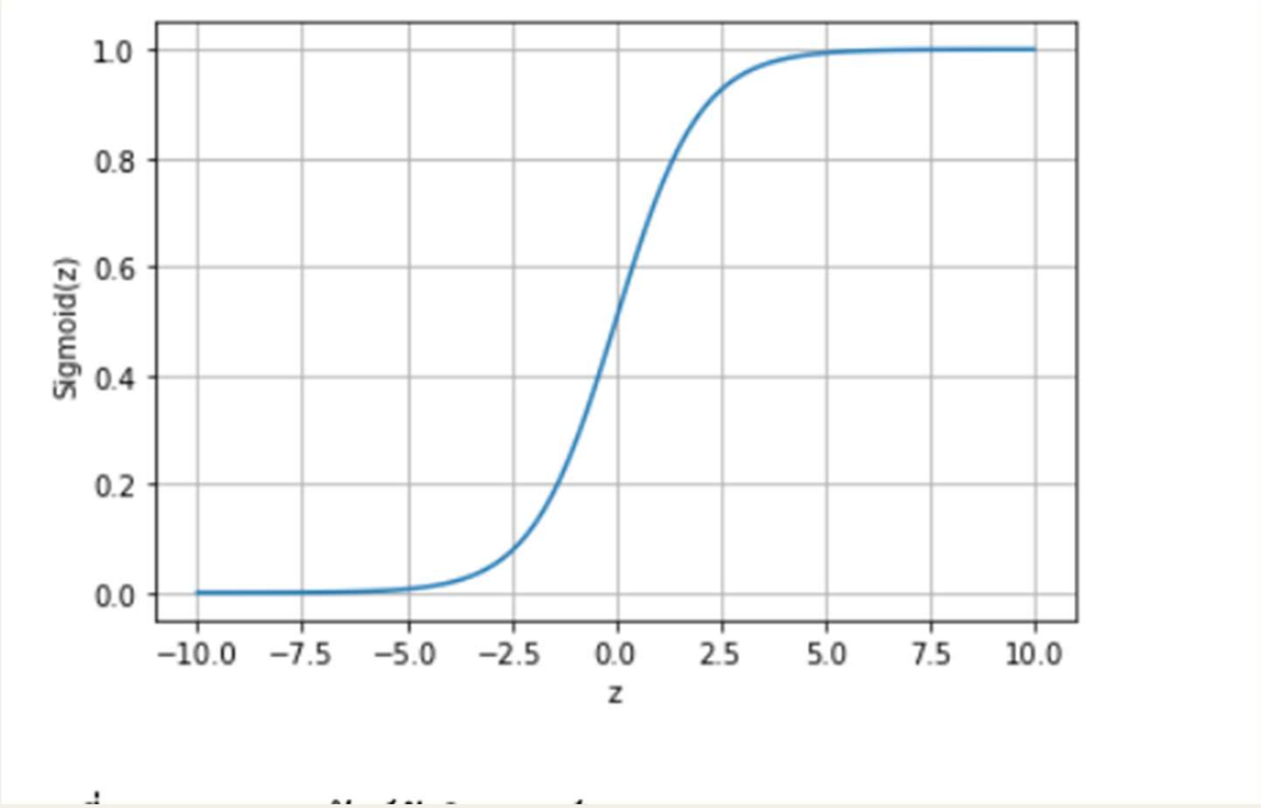
in range [0,1]

$$\hat{y} = \sigma(w^T x + b)$$

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

```
z = np.linspace(-10, 10, 100)
y = 1/(1 + np.exp(-z))

plt.plot(z,y)
plt.xlabel("z")
plt.ylabel("Sigmoid(z)")
plt.grid()
plt.show()
```



Loss function

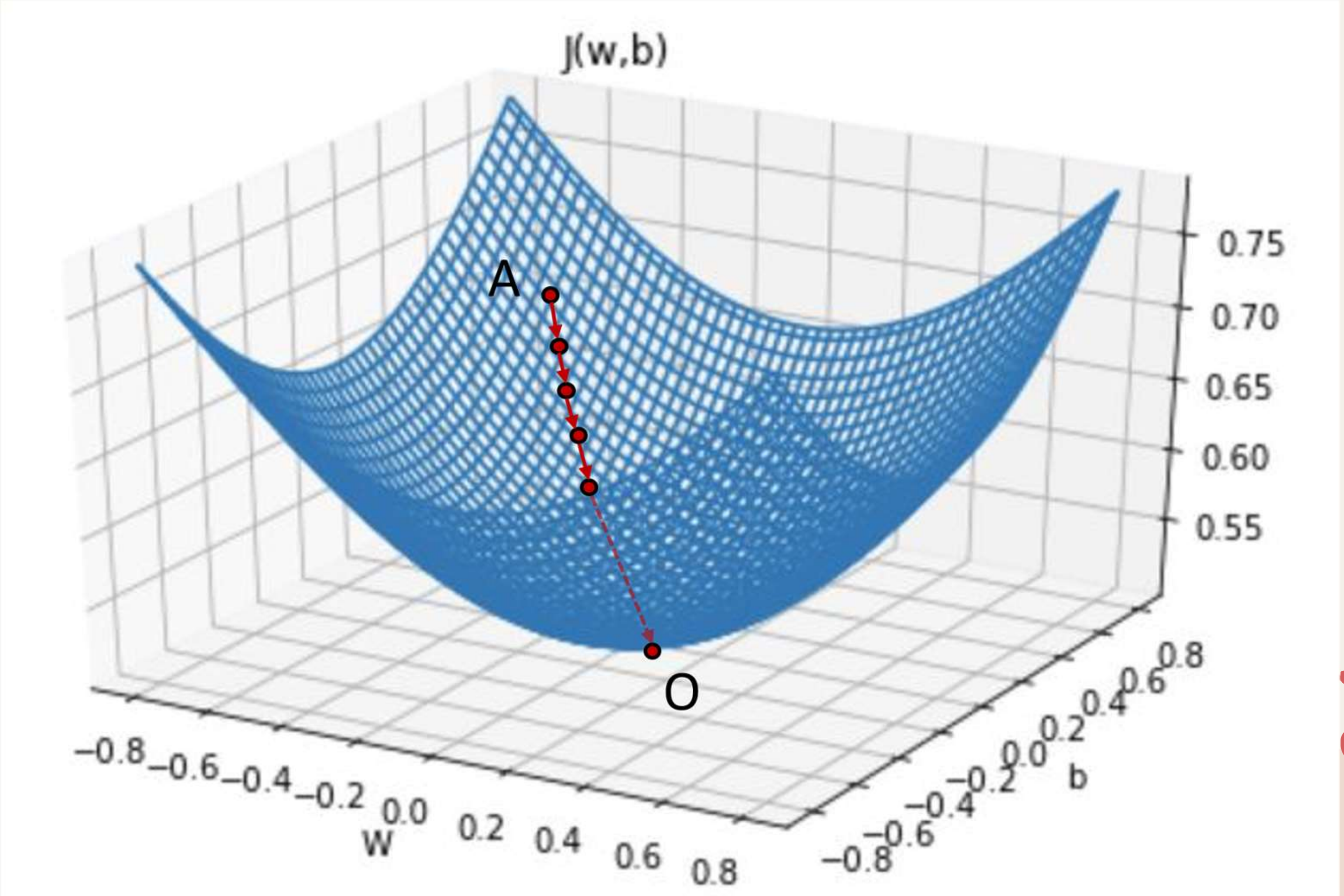
convex function

$$\mathcal{L}(\hat{y}, y) = -(y\ log\hat{y} + (1 - y)log(1 - \hat{y}))$$

$$J(w, b) = \frac{1}{m}\sum_{i=1}^{m}\mathcal{L}(\hat{y}^{(i)}, y^{(i)}) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}log\hat{y}^{(i)} + (1 - y^{(i)})log(1 - \hat{y}^{(i)})]$$

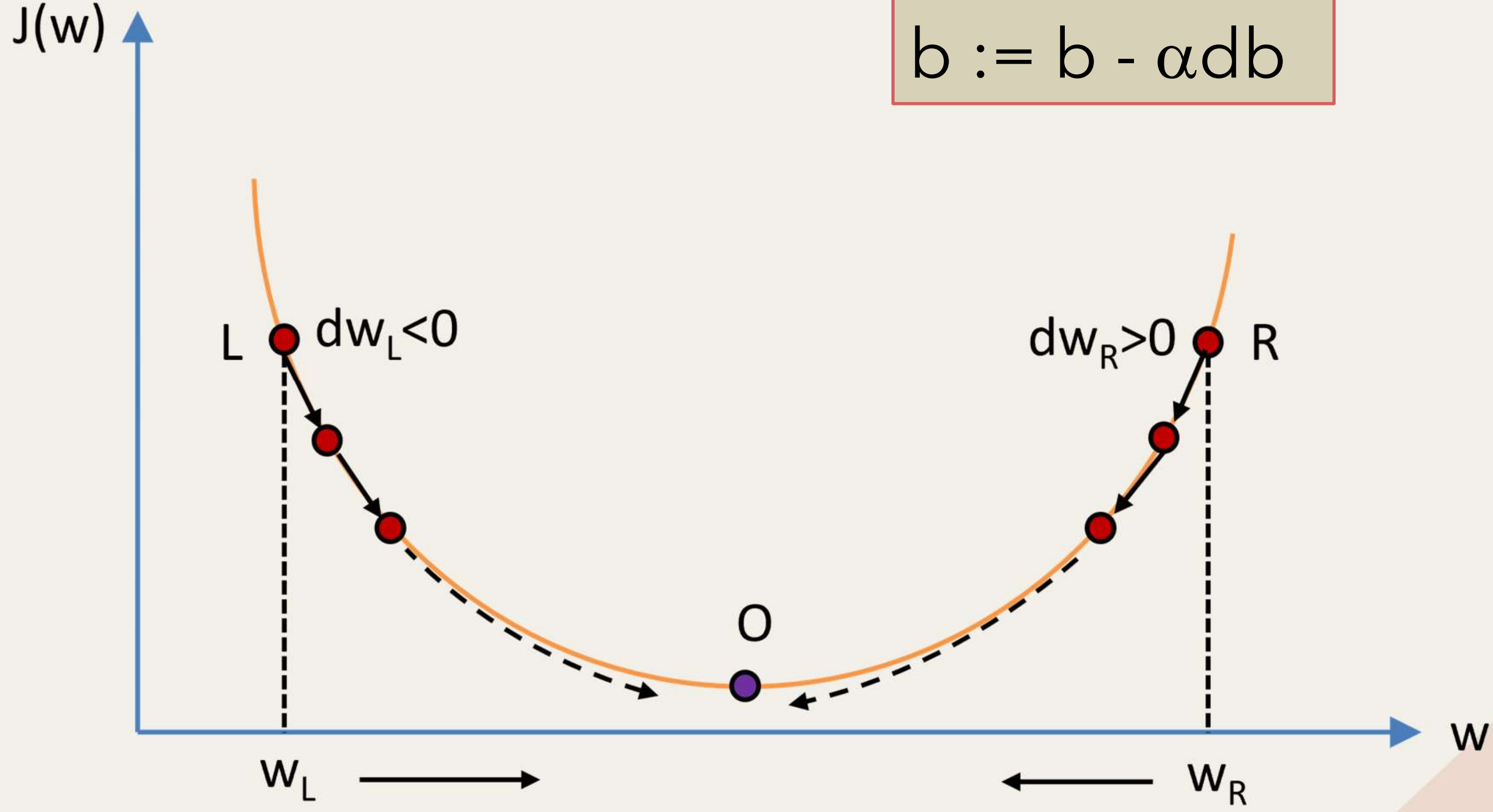
sometimes called ”cost funciton”

Gradient descent

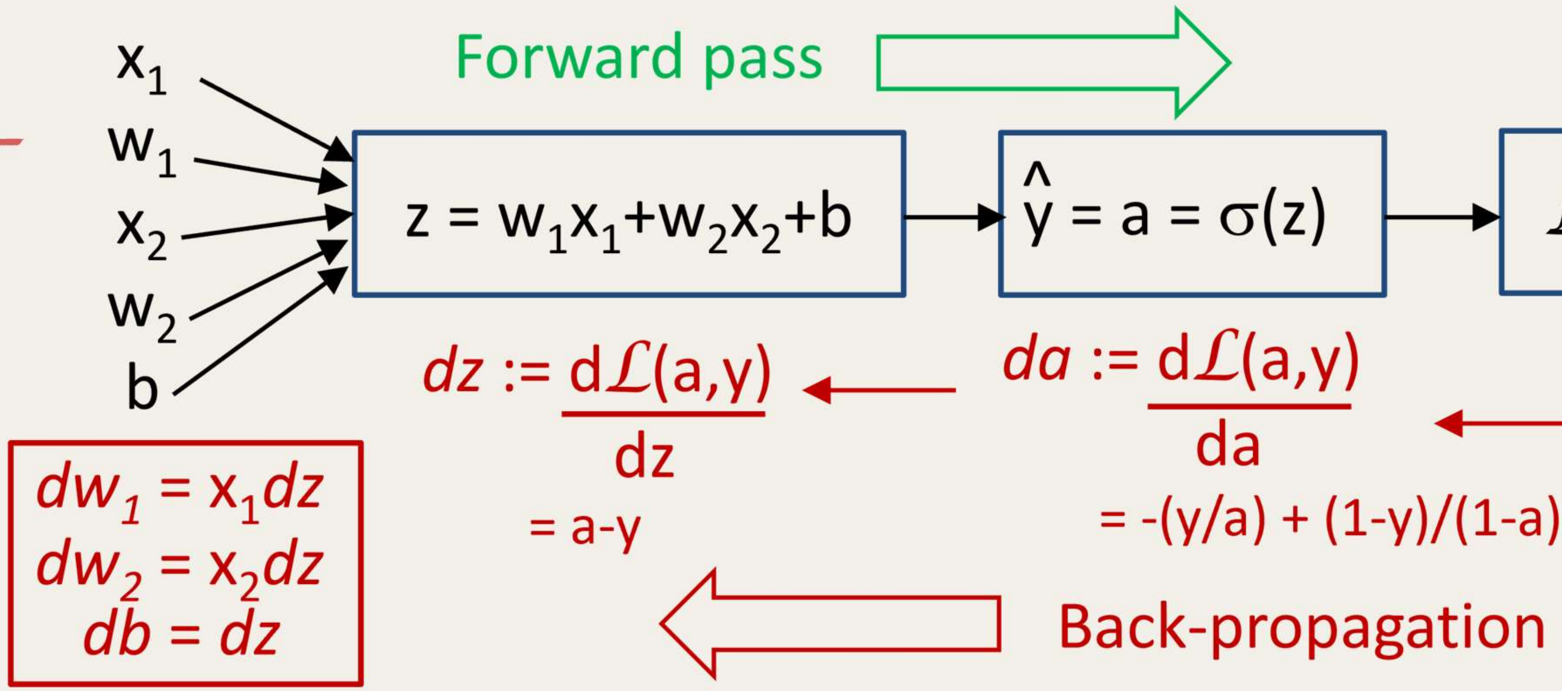


Gradient descent in 2D

$w := w - \alpha dw$
$b := b - \alpha db$

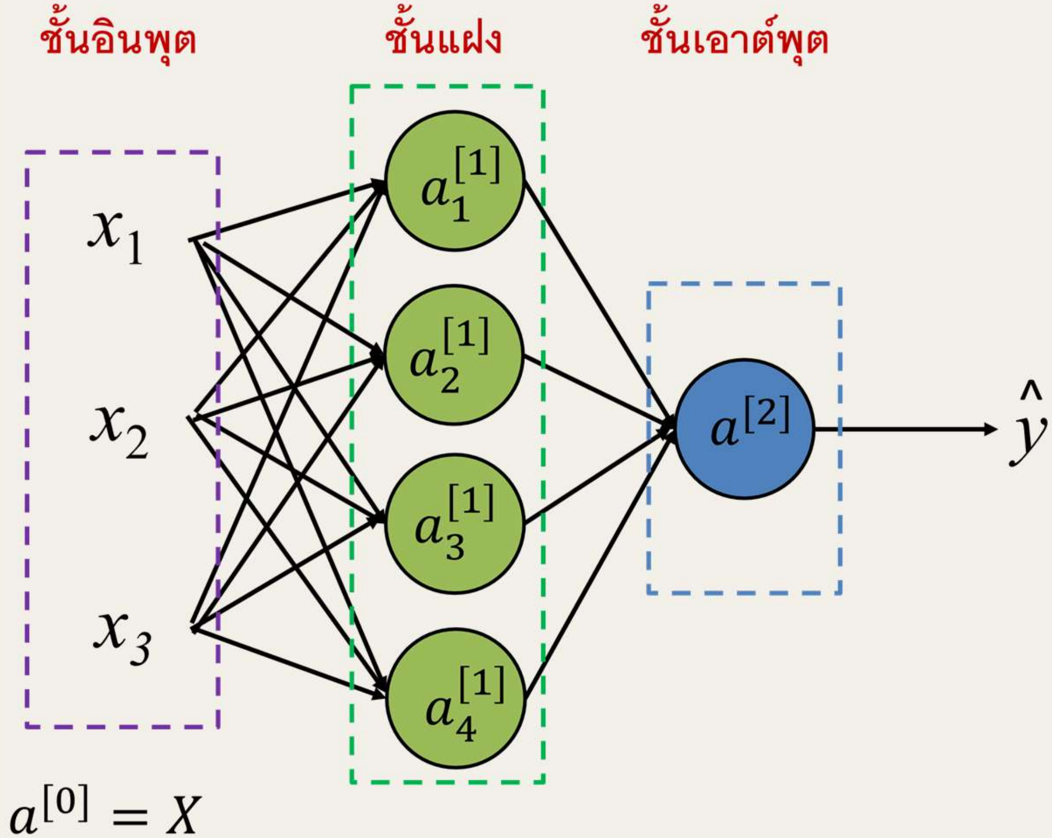


Computational graph of logistic regression

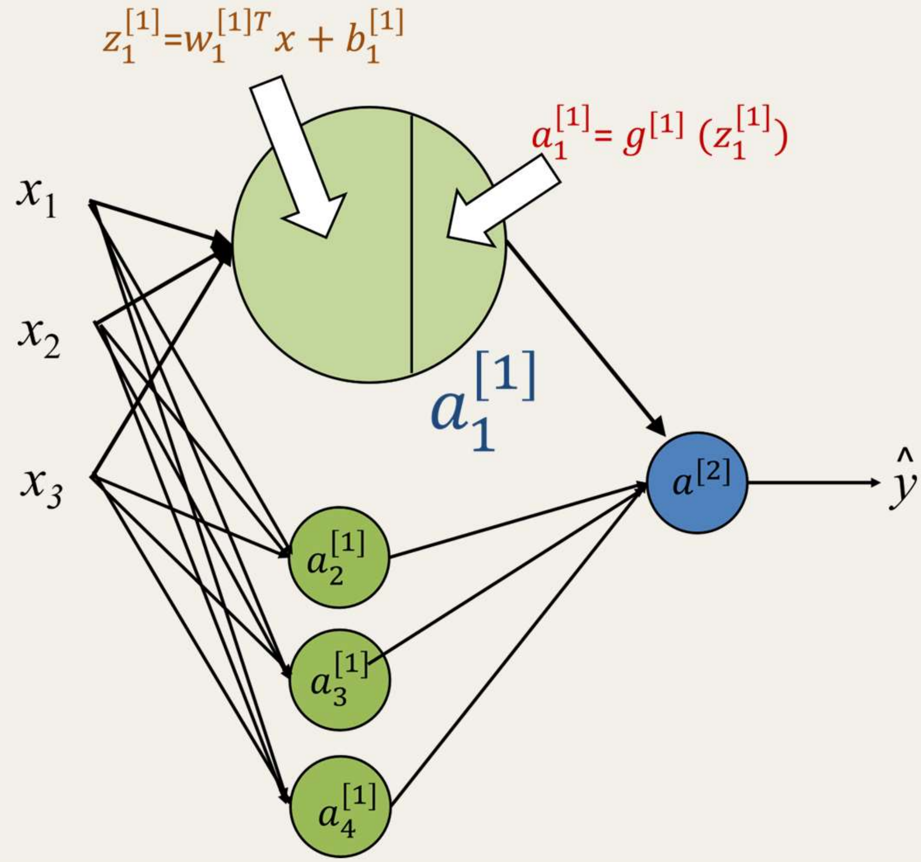


see details in logistic_regression.ipynb

Neural network with one hidden layer



Neural network with one hidden layer



Neural network with one hidden layer (vectorized)

$$\hat{Y} = \left[\, \hat{y}^{(1)} \quad \hat{y}^{(2)} \quad \ldots \quad \hat{y}^{(m)} \,\right]$$

$$Z^{[l]} = \begin{bmatrix} \vdots & \vdots & \vdots & \vdots \\ z^{[l](1)} & z^{[l](2)} & \ldots & z^{[l](m)} \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

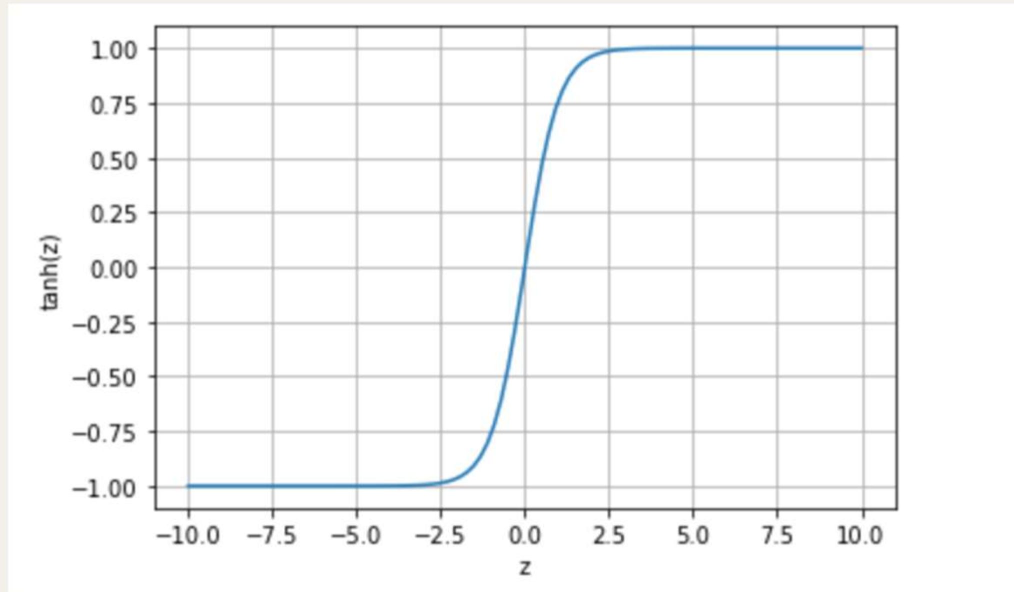
$$A^{[1]} = \begin{bmatrix} \vdots & \vdots & \vdots & \vdots \\ a^{[l](1)} & a^{[l](2)} & \ldots & a^{[l](m)} \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

forward pass

$$\boxed{\begin{aligned} Z^{[1]} &= W^{[1]}X + b^{[1]} \\ A^{[1]} &= g^{[1]}(Z^{[1]}) \\ Z^{[2]} &= W^{[2]}A^{[1]} + b^{[2]} \\ \hat{Y} = A^{[2]} &= g^{[2]}(Z^{[2]}) \end{aligned}}$$

tanh() activation function

```
z = np.linspace(-10, 10, 100)
y = np.tanh(z)
plt.plot(z,y)
plt.xlabel("z")
plt.ylabel("tanh(z)")
plt.grid()
plt.show()
```

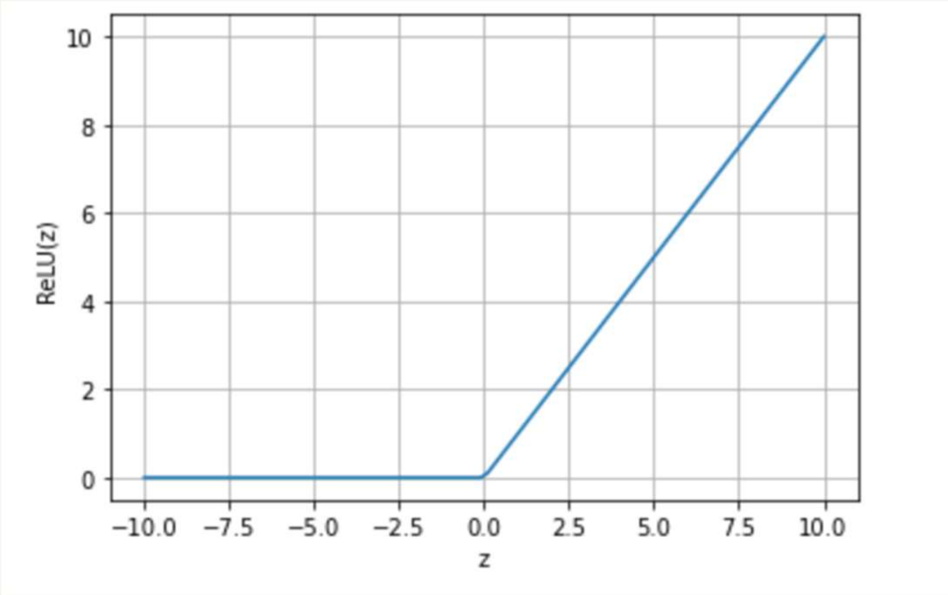


$$tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

ReLU activation function

```
def ReLU(z):
   return z*(z>0)

z = np.linspace(-10, 10, 100)
y = ReLU(z)
plt.plot(z,y)
plt.xlabel("z")
plt.ylabel("ReLU(z)")
plt.grid()
plt.show()
```

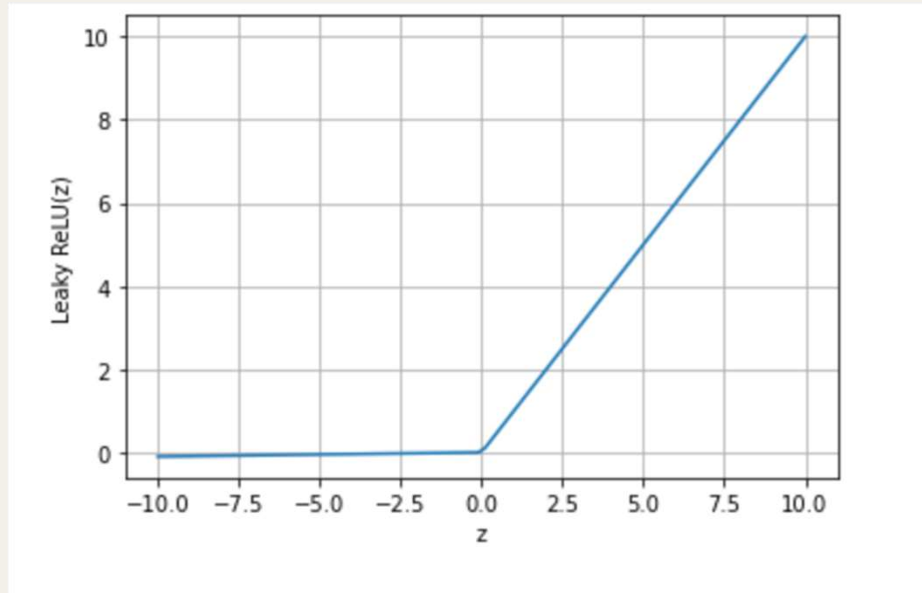


$$f(z) = max(0, z)$$

leaky ReLU activation function

```
def LeakyReLU(z):
   return np.where(z>0, z, z*0.01)

z = np.linspace(-10, 10, 100)
y = LeakyReLU(z)
plt.plot(z,y)
plt.xlabel("z")
plt.ylabel("Leaky ReLU(z)")
plt.grid()
plt.show()
```



$$f(z) = max(\alpha z, z)$$

2.3.4 อนุพันธ์ของฟังก์ชันกระตุ้น

เริ่มจากฟังก์ชันซิกมอยด์ $\sigma()$ (2.4) สามารถแสดงโดยแคลคูลัสได้อนุพันธ์ดังนี้

$$g'(z) = \frac{1}{1+e^{-z}}\left(1 - \frac{1}{1+e^{-z}}\right) = g(z)(1-g(z)) = a(1-a)$$

โดยด้านขวาสุดของ (2.36) มาจากการใช้สัญกรณ์ $a = g(z)$

กรณีฟังก์ชัน $tanh()$ (2.33) ได้อนุพันธ์เท่ากับ

$$g'(z) = 1 - (tanh(z))^2 = 1 - g^2(z) = 1 - a^2$$

สำหรับฟังก์ชัน ReLU (2.34)

$$g'(z) = \begin{cases} 0 & \text{if} \quad z < 0 \\ 1 & \text{if} \quad z > 0 \end{cases}$$

และสำหรับ Leaky ReLU (2.35)

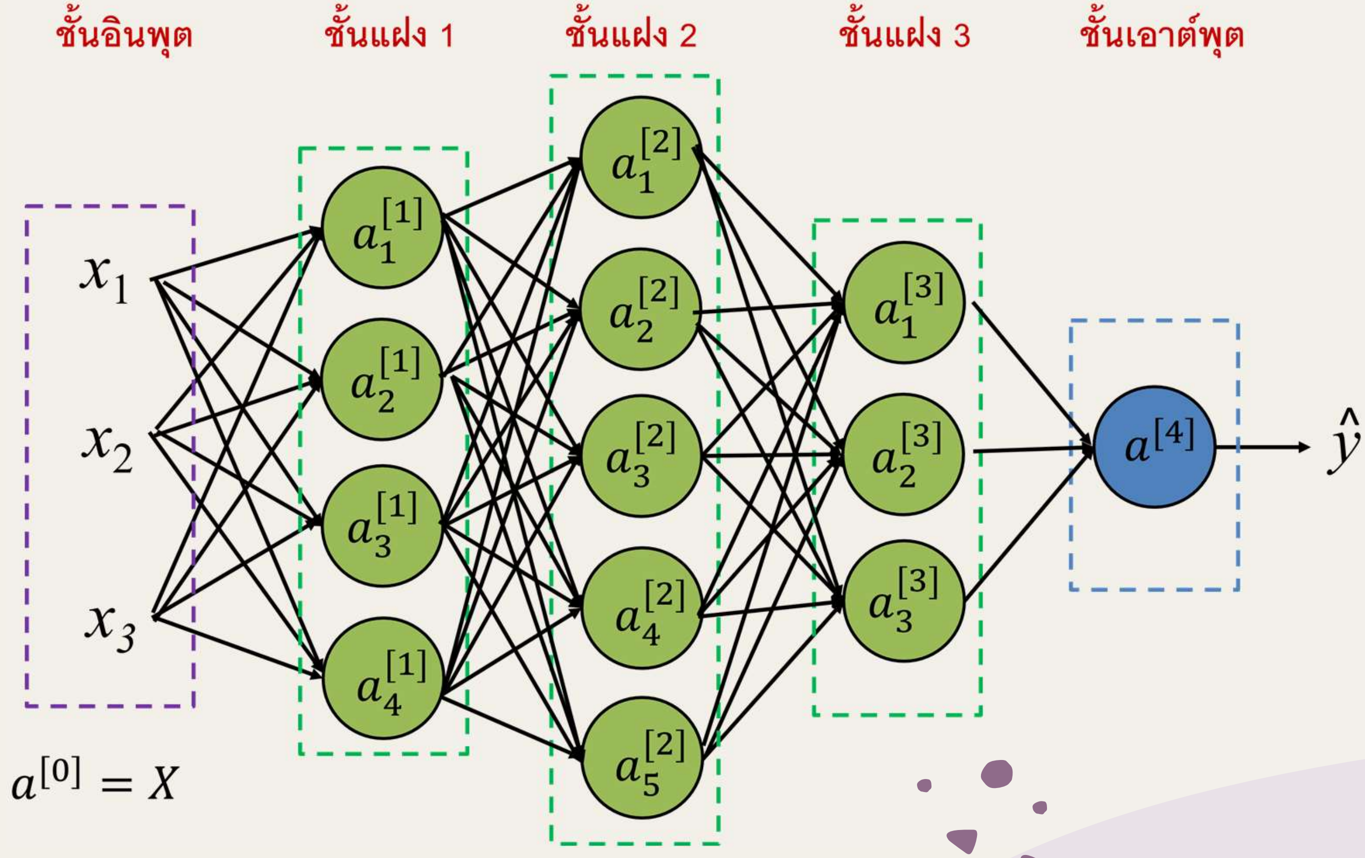
$$g'(z) = \begin{cases} \alpha & \text{if} \quad z < 0 \\ 1 & \text{if} \quad z > 0 \end{cases}$$

ตัวอย่าง 2.3

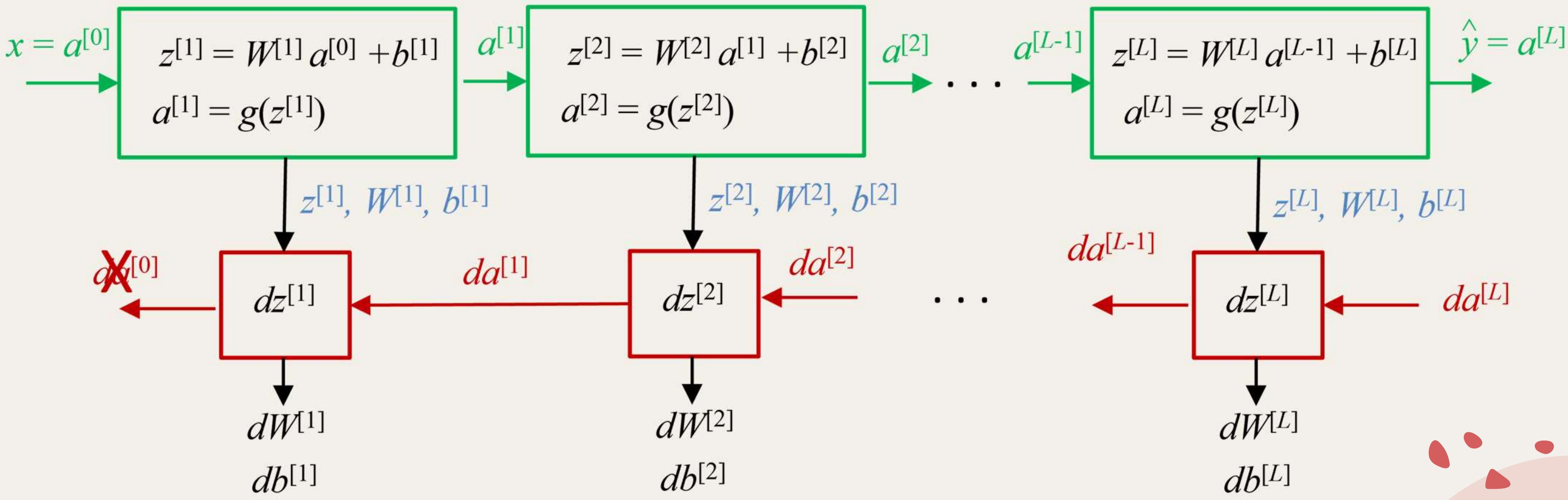
see single_layer_dnn.ipynb

DNN model

also called
MLP (Multi-Layer
Perceptron)



Computational graph of L-layer DNN



see multi_layer_dnn.ipynb

Supplementary notebooks

- model_tf.ipynb
- multiclass.ipynb

Exercises

- Problem 2-4