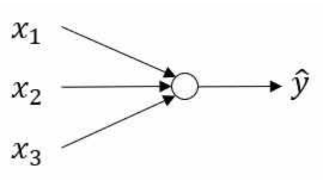
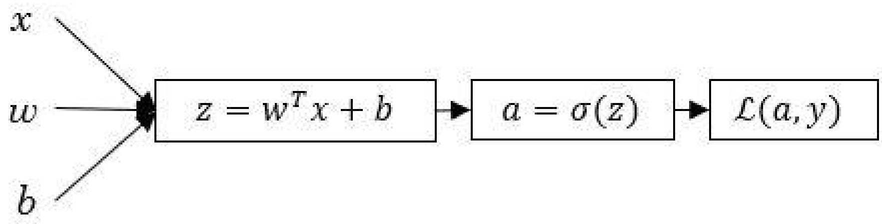
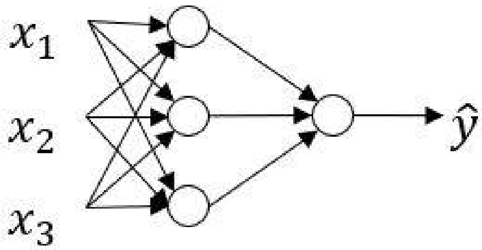
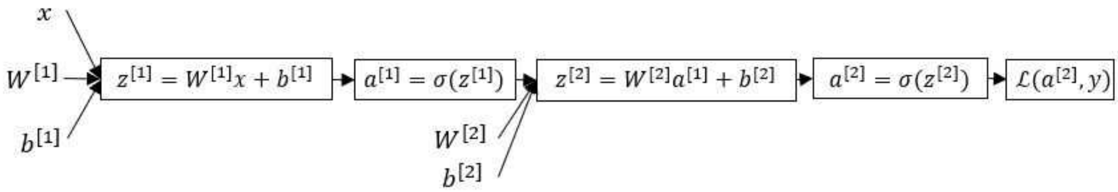
Shallow Neural Networks

Nueral Networks Overview



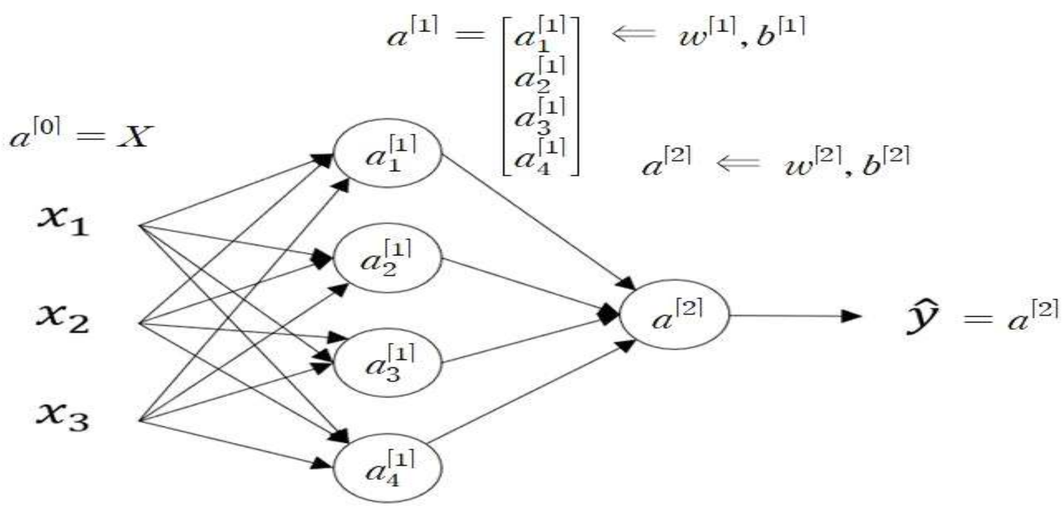






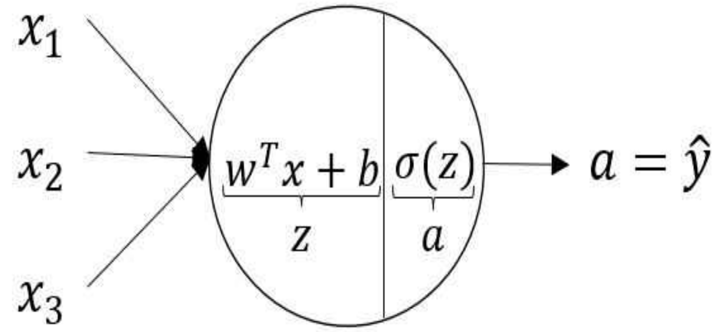
Neural Network Representation

2 layer NN(입력층은 세지 않기 때문에 실제로는 3개 층이지만 2개 층이라고 한다)



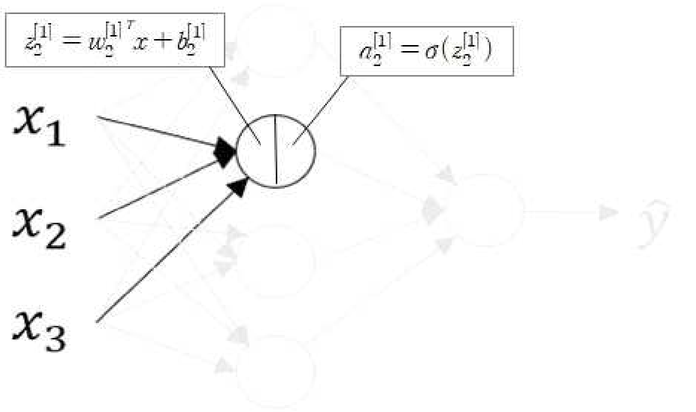
Input layer	Hiddent layer	Output layer

Computing a Neural Network's Output



$z = w^T x + b$

$a = \sigma(z)$



$$a_i^{[l]}$$

l : layer

i : node in layer l

$z_1^{[1]} = w_1^{[1]T} x + b_1^{[1]}, \quad a_1^{[1]} = \sigma(z_1^{[1]})$

$z_2^{[1]} = w_2^{[1]T} x + b_2^{[1]}, \quad a_2^{[1]} = \sigma(z_2^{[1]})$

$z_3^{[1]} = w_3^{[1]T} x + b_3^{[1]}, \quad a_3^{[1]} = \sigma(z_3^{[1]})$

$z_4^{[1]} = w_4^{[1]T} x + b_4^{[1]}, \quad a_4^{[1]} = \sigma(z_4^{[1]})$

$$z^{[1]} = w^{[1]} x + b^{[1]} = \begin{bmatrix} - w_1^{[1]T} - \\ - w_2^{[1]T} - \\ - w_3^{[1]T} - \\ - w_4^{[1]T} - \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \\ b_3^{[1]} \\ b_4^{[1]} \end{bmatrix} = \begin{bmatrix} w_1^{[1]T} x + b_1^{[1]} \\ w_2^{[1]T} x + b_2^{[1]} \\ w_3^{[1]T} x + b_3^{[1]} \\ w_4^{[1]T} x + b_4^{[1]} \end{bmatrix} = \begin{bmatrix} z_1^{[1]} \\ z_2^{[1]} \\ z_3^{[1]} \\ z_4^{[1]} \end{bmatrix}$$

$$a^{[1]} = \begin{bmatrix} a_1^{[1]} \\ a_2^{[1]} \\ a_3^{[1]} \\ a_4^{[1]} \end{bmatrix} = \sigma(z^{[1]})$$

$x = a^{[0]}$, $w_i^{[l]} = W^{[l]}$, $\hat{y} = a^{[L]}$

$$\underset{(4,1)}{z^{[1]}} = \underset{(4,3)}{W^{[1]}} \underset{(3,1)}{x} + \underset{(4,1)}{b^{[1]}} = \underset{(4,3)}{W^{[1]}} \underset{(3,1)}{a^{[1]}} + \underset{(4,1)}{b^{[1]}}$$

$$\underset{(4,1)}{a^{[1]}} = \sigma(\underset{(4,1)}{z^{[1]}})$$

$$\underset{(1,1)}{z^{[2]}} = \underset{(1,4)}{W^{[2]}} \underset{(4,1)}{a^{[2]}} + \underset{(1,1)}{b^{[2]}}$$

$$\underset{(1,1)}{a^{[2]}} = \sigma(\underset{(1,1)}{z^{[2]}})$$

Vectorizing across multiple examples

$$a^{[l](i)}$$

l : layer

i : example i

$$X_{n_x \times m} = \overset{\#trainingexamples(m)}{\begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix}} \updownarrow \#features(n_x),\ X = A^{[0]},$$

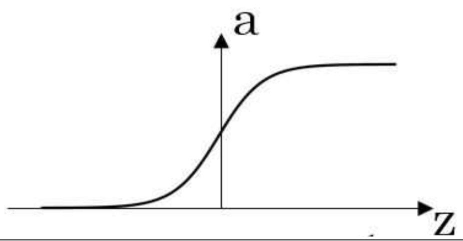
$$Z^{[1]} = W^{[1]}X + b^{[1]} = W^{[1]}A^{[0]} + b^{[1]}$$
$$A^{[1]} = \sigma(Z^{[1]})$$
$$Z^{[2]} = W^{[2]}A^{[1]} + b^{[2]}$$
$$A^{[2]} = \sigma(Z^{[2]})$$

$$Z^{[1]} = \overset{\#trainingexamples}{\begin{bmatrix} | & | & & | \\ z^{[1](1)} & z^{[1](2)} & \cdots & z^{[1](m)} \\ | & | & & | \end{bmatrix}} \updownarrow hiddenunits$$

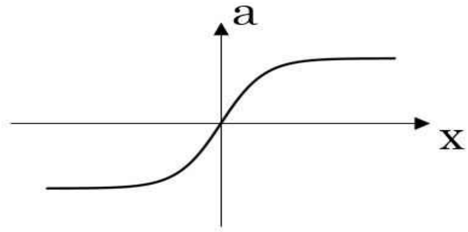
$$A^{[1]} = \overset{\#trainingexamples(m)}{\begin{bmatrix} | & | & & | \\ a^{[1](1)} & a^{[1](2)} & \cdots & a^{[1](m)} \\ | & | & & | \end{bmatrix}} \updownarrow hiddenunits(n_x)$$

$$Z^{[1]} = W^{[1]}X + b^{[1]} = \left[\ W^{[1]}x^{(1)} + b^{[1]}\ \ W^{[1]}x^{(2)} + b^{[1]} \cdots\ W^{[1]}x^{(m)} + b^{[1]} \right] = \begin{bmatrix} | & | & & | \\ z^{[1](1)} & z^{[1](2)} & \cdots & z^{[1](m)} \\ | & | & & | \end{bmatrix}$$

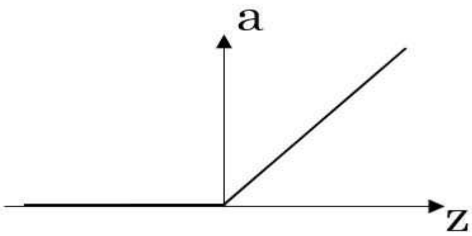
Activation functions



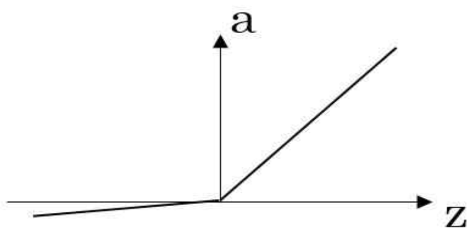
sigmoid: $a = \dfrac{1}{1+e^{-z}}$



tanh: $a = \dfrac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$



ReLU: $a = max(0, z)$



Leaky ReLU: $a = max(0.01z, z)$

Why need non-linear activation functions?

activation function이 없거나 linear activation function을 사용하면 결과값은 단순한 입력값의 선형 결합이다. 따라서 hidden layer가 의미가 없어진다. 각 층은 결국 선형 결합으로 치환할 수 있기 때문이다. 단, 마지막 결과값을 계산하는 경우(회귀)에는 linear activation function을 사용할 수 있다.(ex. 집값 예측)

Derivates of activation functions

$g(z): activation\, function$, $g'(z) = \frac{d}{dz}g(z) = slope\, of\, g(x)\, at\, z$	
sigmoid: $a = \frac{1}{1+e^{-z}}$	$g'(z) = a(1-a)$
tanh: $a = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	$g'(z) = 1-(\tanh(z))^2$
ReLU: $a = max(0, z)$	$g'(z) = \begin{cases} 0 & \text{if } z < 0 \\ 1 & \text{if } z > 0 \\ undefined & \text{if } z = 0 \end{cases} = \begin{cases} 0 & \text{if } z < 0 \\ 1 & \text{if } z \geq 0 \end{cases}$
Leaky ReLU: $a = max(0.01z, z)$	$g'(z) = \begin{cases} 0.01 & \text{if } z < 0 \\ 1 & \text{if } z > 0 \\ undefined & \text{if } z = 0 \end{cases} = \begin{cases} 0.01 & \text{if } z < 0 \\ 1 & \text{if } z \geq 0 \end{cases}$

Sigmoid의 경우 0과 1 사의 값으로 나오므로 binary classification에 좋다

Random Initialization

1. 모든 W를 0으로 초기화하면 모든 neuron과 layer가 같은 값으로 처리하고, 많은 iteration이 지나도 같은 값으로 처리하기 때문에 $np.random.randn((n_l, n_{l-1}))$
2. W를 그냥 혹은 큰 값을 곱해주면 activation function의 기울기가 너무 작아 학습 속도가 느리기 때문에 $W^{[l]} = np.random.randn((n_l, n_{l-1})) * 0.01$

$$W^{[l]} = np.random.randn((n_l, n_{l-1})) * 0.01$$

$$b^{[l]} = np.zeros((n_l, 1))$$

Gradient descent for neural networks

<table>
<tr><td colspan="2">Parameters: $W^{[1]}$, $b^{[1]}$, $W^{[2]}$, $b^{[2]}$
$n_x = n^{[0]}$, $n^{[1]}$, $n^{[2]} = 1$</td></tr>
<tr><td>Forward propagation

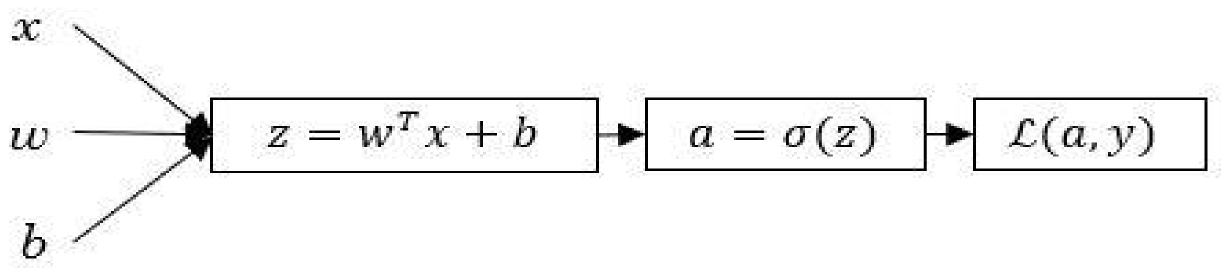
$Z^{[1]} = W^{[1]}X + b^{[1]} = W^{[1]}A^{[0]} + b^{[1]}$
$A^{[1]} = g^{[1]}(Z^{[1]})$

$Z^{[2]} = W^{[2]}A^{[1]} + b^{[2]}$
$A^{[2]} = g(Z^{[2]}) = \sigma(Z^{[2]})$</td><td>Backward propagation

$dZ^{[2]} = A^{[2]} - Y,\ \ Y = \left[y^{(1)}\, y^{(2)} \cdots y^{(m)} \right]$
$dW^{[2]} = \frac{1}{m} dZ^{[2]} A^{[1]^T}$
$db^{[2]} = \frac{1}{m}\, np.sum(dZ^{[2]}, axis = 1, keepdims = True)$

$dZ^{[1]} = W^{[2]^T} dZ^{[2]} * g'^{[1]}(Z^{[1]})$
$dW^{[1]} = \frac{1}{m} dZ^{[1]} A^{[0]^T} = \frac{1}{m} dZ^{[1]} X^T$
$db^{[1]} = \frac{1}{m}\, np.sum(dZ^{[1]}, axis = 1, keepdims = True)$</td></tr>
<tr><td colspan="2">Cost function: $J(W^{[1]}, b^{[1]}, W^{[2]}, b^{[2]}) = \frac{1}{m}\sum_{i=1}^{n} L(\hat{y}, y)$, $\hat{y} = A^{[2]}$</td></tr>
<tr><td colspan="2">Gradient descent
repeat {
Compute predict ($\hat{y}^{(i)}$, $i = 1, \cdots, m$)
$dW^{[1]} = \frac{\partial J}{\partial W^{[1]}}$, $db^{[1]} = \frac{\partial J}{\partial b^{[1]}}$, $dW^{[2]} = \frac{\partial J}{\partial W^{[2]}}$, $db^{[2]} = \frac{\partial J}{\partial b^{[2]}}$
$W^{[1]} := W^{[1]} - \alpha\, dW^{[1]}$
$b^{[1]} := b^{[1]} - \alpha\, db^{[1]}$
$W^{[2]} := W^{[2]} - \alpha\, dW^{[2]}$
$b^{[2]} := b^{[2]} - \alpha\, db^{[2]}$
}</td></tr>
</table>

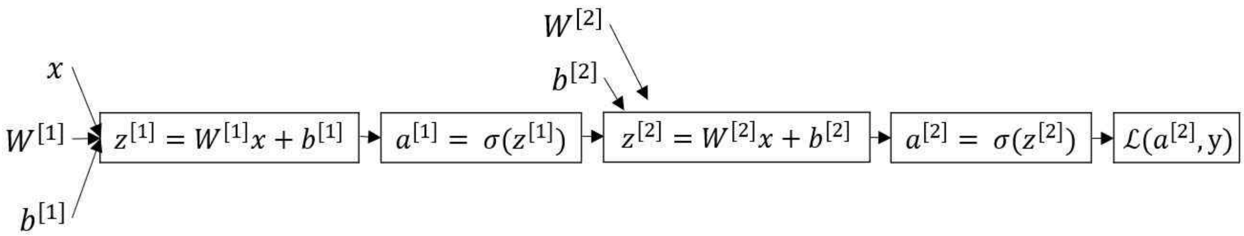
Backpropation intuition



$$dA = \frac{\partial}{\partial A} L(A,\, Y) = -\frac{Y}{A} + \frac{1-Y}{1-A}, \quad g(Z) = \sigma(Z)$$

$$dZ = \frac{\partial L}{\partial Z} = \frac{\partial L}{\partial a}\frac{\partial a}{\partial Z} = da\,\frac{d}{aZ} g(Z) = da\, g'(Z)$$

if $g(Z) = \sigma(Z)$, $dZ = A - Y$



$$dA^{[2]} = \frac{\partial}{\partial A^{[2]}} L(A^{[2]},\, Y) = -\frac{Y}{A^{[2]}} + \frac{1-Y}{1-A^{[2]}}, \quad g^{[2]}(Z) = \sigma(Z)$$

$$dZ^{[2]} = A^{[2]} - Y$$

$$dW^{[2]} = \frac{1}{m} dZ^{[2]} A^{[1]^T} \;\rightarrow\; W^{[2]} := W^{[2]} - \alpha\, dW^{[2]}$$

$$db^{[2]} = \frac{1}{m}\sum_{i=1}^{n^{[2]}} dZ^{[2](i)} \longrightarrow b^{[2]} := b^{[2]} - \alpha\, db^{[2]}$$

$$dZ^{[1]} = W^{[2]^T} dZ^{[2]} * g'^{[1]}(Z^{[1]})$$

$$dW^{[1]} = \frac{1}{m} dZ^{[1]} A^{[0]^T} = \frac{1}{m} dZ^{[1]} X^T \;\rightarrow\; W^{[1]} := W^{[1]} - \alpha\, dW^{[1]}$$

$$db^{[1]} = \frac{1}{m}\sum_{i=1}^{n^{[1]}} dZ^{[1](i)} \longrightarrow b^{[1]} := b^{[1]} - \alpha\, db^{[1]}$$