Neural Networks and Deep Learning

1. What is a Neural Network?

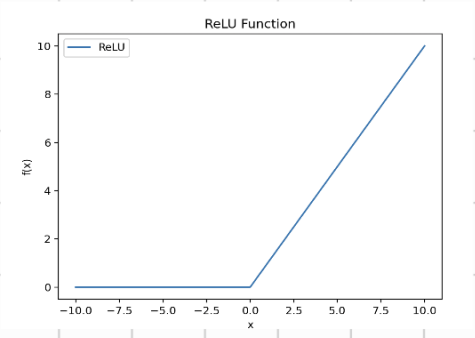
Example) 집값 예측 문제

- x : 집 크기 (입력값)
- y : 가격 (출력값)

⇒ size (x) → ○ (neuron) → Price (y)

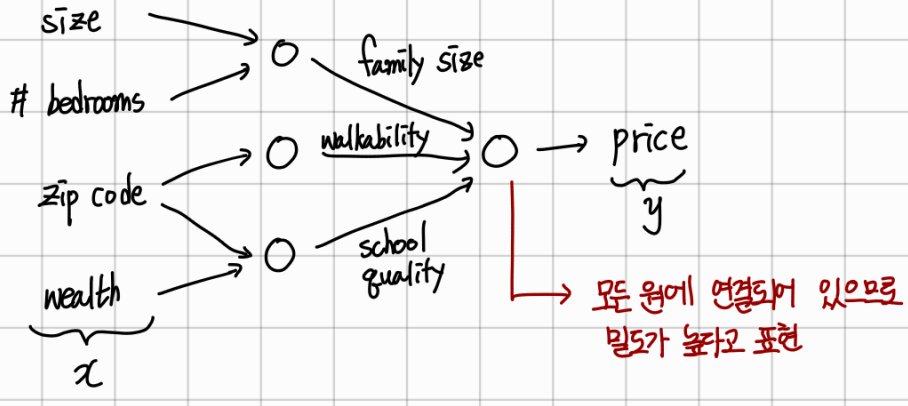
- 뉴런 : 크기 입력, 선형함수계산, 예상 가격 출력

① ReLu 함수 (Rectified Linear Unit)



- $ReLU(x) = max(0, x)$
- 양수일 시 입력값(x) 출력
- 양수가 아닐 시 0 출력
- 기울기 [1 , 입력값이 양수일 때 / 0 , 입력값이 음수일 때

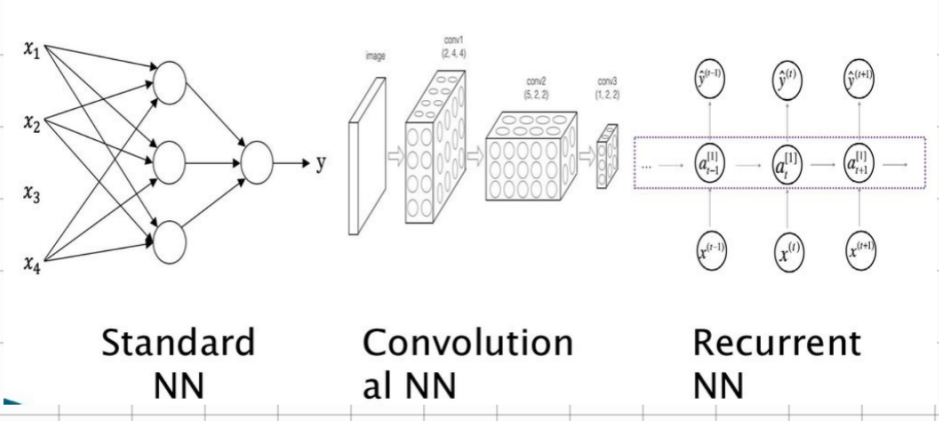
② 신경망 예제



- x, y에 대한 충분한 훈련 예제가 주어질 시 신경망은 그에서 y까지 매핑하는 함수를 알아냄.

2. Supervised Learning with Neural Networks

Example)	Input (X)	Output (y)	Application	NN
	Home features	Price	Real Estate	Standard NN
	Ad, user info	Click on ad? (0/1)	Online Advertising	Standard NN
	Image	Object (1, ..., 1000)	Photo tagging	CNN
	Audio	Text transcript	Speech recognition	RNN
	English	Chinese	Machine translation	RNN
	Image, Radar info	Position of other cars	Autonomous driving	Custom/ Hybrid



- CNN : 이미지 데이터에 자주 쓰임
- RNN : 시간적 요소가 있을 수 있는 1차원 시퀀스 데이터에 매우 적합

- Unstructured Data

  ex) 오디오, 이미지, 텍스트

  feature : 이미지의 픽셀 값, 텍스트의 개별 단어

Basic of Neural Network Programming

① Binary Classification

- 로지스틱 회귀 : 이진 분류 알고리즘
- Notation

1) (x, y) : 하나의 훈련 표본
2) $x \in \mathbb{R}^{n_x}$: x 차원을 가진 특징 벡터
3) $y \in \{0, 1\}$: y는 0 or 1 중에 하나의 값을 가지는 레이블
4) m training example : $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \cdots, (x^{(m)}, y^{(m)})$
5) 학습 데이터임을 강조하기 위해 $m = m_train$으로 작성
6) M_{test} = # test example
7) $X = \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix}_{n_x}$ (m 방향), $X \in \mathbb{R}^{n_x \times m}$, $X.shape = (n_x, m)$

② Logistic Regression

- Given x, want $\hat{y} = P(y=1|x)$, $x \in \mathbb{R}^{n_x}$
- parameters : $w \in \mathbb{R}^{n_x}$, $b \in \mathbb{R}$ ⇒ b는 스펙트럼 간 대응
- Output : $\hat{y} = \sigma(w^T x + b)$, σ : sigmoid
- $\sigma(z) = \frac{1}{1+e^{-z}}$, if z large $\sigma(z) \approx \frac{1}{1+0} = 1$

  if z large negative number $\sigma(z) \approx \frac{1}{1+\infty} \approx 0$

③ Logistic Regression Cost Function

- $z^{(i)} = w^T x^{(i)} + b$
- Loss (error) function : $\mathcal{L}(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$

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

  if $y=1$: $\mathcal{L}(\hat{y}, y) = -\log \hat{y}$ ← want $\log \hat{y}$ large, want $\hat{y}$ large

  if $y=0$: $\mathcal{L}(\hat{y}, y) = -\log(1-\hat{y})$ ← want $\log 1-\hat{y}$ large, want $\hat{y}$ small
- Cost function : $J(w, b) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)})$

  $= -\frac{1}{m} \sum_{i=1}^{m} [y^{(i)} \log \hat{y}^{(i)} + (1-y^{(i)}) \log(1-\hat{y}^{(i)})]$

④ Gradient Descent

- Algorithm

  Repeat {

  $w := w - \alpha \frac{dJ(w)}{dw}$

  }

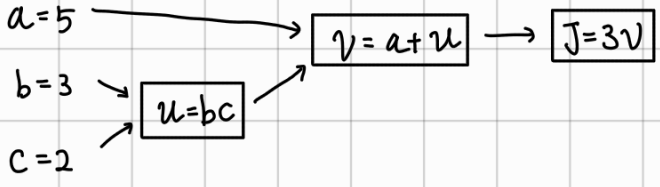
- α : learning rate

  $w := w - \alpha \frac{\partial J(w,b)}{\partial w}$ → J가 두개 이상의 변수의 함수라면 d 대신 ∂ 사용

  $b := b - \alpha \frac{\partial J(w,b)}{\partial b}$ → "

⑤ Derivatives with Computation Graph

- Computation Graph



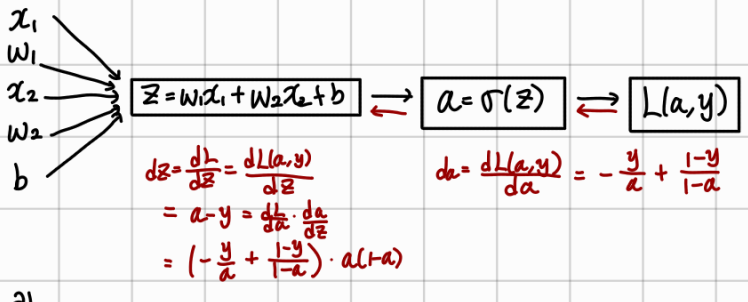
- $\frac{dJ}{dv} = 3 = \frac{dJ}{dv} \cdot \frac{dv}{da}$, $\frac{df(a)}{da} = \frac{df}{da} = 3$
- $\frac{d\text{FinalOutputVar}}{d\text{Var}}$ 코딩→ "dvar" : J와 같이 완성 있는 최종 출력 변수의 도함수
- 역전파 방식 때문에 du 또한 3이라는 값을 가짐 ($\frac{dJ}{du}$)

  $\frac{dJ}{db} = \frac{dJ}{du} \cdot \frac{du}{db}$, $\frac{du}{db} = 2$ $\therefore \frac{dJ}{db} = 6$

  $\frac{dJ}{dc} = \frac{dJ}{du} \cdot \frac{du}{dc} = 9$

⑥ Logistic Regression Gradient Descent

- $\hat{y} = a = \sigma(z)$
- Diagram



$dz = \frac{dL}{dz} = \frac{dL(a,y)}{dz}$

$= a - y = \frac{dL}{da} \cdot \frac{da}{dz}$

$= \left(-\frac{y}{a} + \frac{1-y}{1-a}\right) \cdot a(1-a)$

$da = \frac{dL(a,y)}{da} = -\frac{y}{a} + \frac{1-y}{1-a}$

- $\frac{\partial L}{\partial w_1} = dw_1 = x_1 \cdot dz$

⑦ Gradient descent on m examples

- $J(w,b) = \frac{1}{m}\sum_{i=1}^{m} L(a^{(i)}, y)$

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

- $\frac{\partial}{\partial w_1} J(w,b) = \frac{1}{m}\sum_{i=1}^{m} \frac{\partial}{\partial w_1} L(a^{(i)}, y^{(i)})$ → 전체 기울기

- Algorithm

$J=0$; $dw_1=0$; $dw_2=0$; $db=0$

For $i=1$ to m :

$z^{(i)} = w^T x^{(i)} + b$

$a^{(i)} = \sigma(z^{(i)})$

$J \mathrel{+}= -[y^{(i)} \log a^{(i)} + (1-y^{(i)}) \log(1-a^{(i)})]$

$dz^{(i)} = a^{(i)} - y^{(i)}$

$dw_1 \mathrel{+}= x_1^{(i)} dz^{(i)}$

$dw_2 \mathrel{+}= x_2^{(i)} dz^{(i)}$] $n=2$

$db \mathrel{+}= dz^{(i)}$

$J /= m$

$dw_1 /= m$; $dw_2 /= m$; $db /= m$

$dw_1 = \frac{\partial J}{\partial w_1}$

$w_1 := w_1 - \alpha dw_1$

$w_2 := w_2 - \alpha dw_2$

$b := b - \alpha db$

⇒ 함수 n개에 대한 for 루프가 필요

↳ 벡터화기법 (Vectorization) 사용

※ 딥러닝에서 for loop 제거가 중요

⑧ Vectorization

- for 루프 제거 기술
- Non-vectorized vs vectorized

```
z = 0
for i in range(n-x):
    z += w[i] * x[i]
z += b
```

```
z = np.dot(w, x) + b
```

GPU, CPU] SIMD (단일 명령 다중 데이터)

- Logistic regression

$dz = [dz^{(1)}\ dz^{(2)}\ \cdots\ dz^{(m)}]$

$= A - Y = [a^{(1)} - y^{(1)}\ \ a^{(2)} - y^{(2)}\ \ \cdots\ \ a^{(m)} - y^{(m)}]$

$db = \frac{1}{m}\sum_{i=1}^{m} dz^{(i)} = \frac{1}{m}$ np.sum(dz)

$dw = \frac{1}{m} X dz^T = \frac{1}{m}\begin{bmatrix} x_1^{(1)} & \cdots & x_1^{(m)} \end{bmatrix}\begin{bmatrix} dz^{(1)} \\ \vdots \\ dz^{(m)} \end{bmatrix} = \frac{1}{m}[x^{(1)} dz^{(1)} + \cdots + x^{(m)} dz^{(m)}]$

- Logistic Regression Gradient descent 단일 반복 알고리즘

$z = w^T x + b$ = np.dot(w.T, x) + b

$A = \sigma(z)$

$dz = A - Y$

$dw = \frac{1}{m} X dz^T$

$db = \frac{1}{m}$ np.sum(dz)

$w := w - \alpha dw$

$b := b - \alpha db$

⑨ Broadcasting in Python

- Example (1)

	Apples	Beef	Eggs	Potatoes
Carb	56.0	0.0	4.4	68.0
Protein	1.2	104.0	52.0	8.0
Fat	1.8	135.0	99.0	0.9

```
cal = A.sum(axis = 0)
percentage = 100 * A / (cal.reshape(1, 4))
```

- Example (2)

$\begin{bmatrix}1\\2\\3\\4\end{bmatrix}$ + 100의 경우 100 → $\begin{bmatrix}100\\100\\100\\100\end{bmatrix}$ 으로 전환하여 계산 (100을 매개변수 boc)

⑩ A note on python / numpy vectors

- 행렬 설정 시 np.random.randn(n, 1)로 설정
- a = np.random.randn(5)
  a.shape(5,) → rank-1 array] Don't use
- a = np.random.randn(5, 1) → column vector
- a = np.random.randn(1, 5) → row vector

⑪ Logistic Regression cost function

- If $y=1$: $p(y|x)=\hat{y}$
  If $y=0$: $p(y|x)=1-\hat{y}$

  $\Rightarrow p(y|x)=\hat{y}^y(1-\hat{y})^{(1-y)}$

  $\log p(y|x)=\log \hat{y}^y(1-\hat{y})^{(1-y)}=y\log\hat{y}+(1-y)\log(1-\hat{y})=-L(\hat{y},y)$

- Cost on m examples

  $\log p(\text{labels in training set})=\log\prod_{i=1}^{m}p(y^{(i)}|x^{(i)})$

  $\log p(\cdots)=\sum_{i=1}^{m}\log p(y^{(i)}|x^{(i)})=-\sum_{i=1}^{m}L(\hat{y}^{(i)},y^{(i)})$] MLE

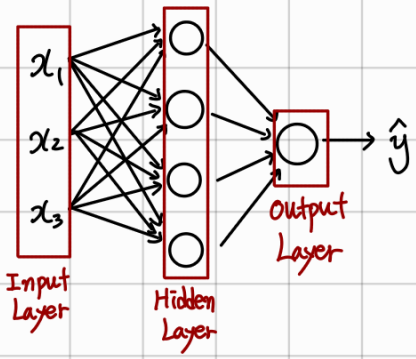
  $J(w,b)=\frac{1}{m}\sum_{i=1}^{m}L(\hat{y}^{(i)},y^{(i)})$

✡ 퀴즈

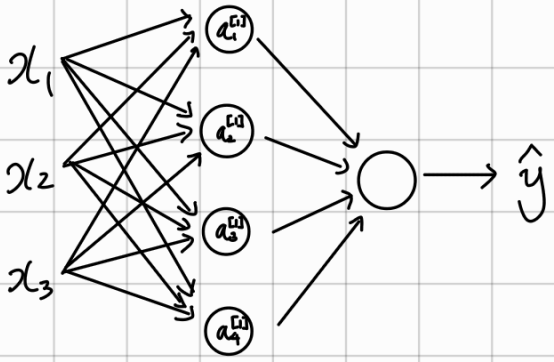
- 물류손실 값 묻는 경우 : $\frac{1}{2}(y-\hat{y})^2$

One hidden layer Neural Network

① Neural Network Representation



② Computing a Neural Network's output



$$z_1^{[1]} = w_1^{[1]T} x + b_1^{[1]}, \quad a_1^{[1]} = \sigma(z_1^{[1]})$$
$$z_2^{[1]} = w_2^{[1]T} x + b_2^{[1]}, \quad a_2^{[1]} = \sigma(z_2^{[1]})$$
$$z_3^{[1]} = w_3^{[1]T} x + b_3^{[1]}, \quad a_3^{[1]} = \sigma(z_3^{[1]})$$
$$z_4^{[1]} = w_4^{[1]T} x + b_4^{[1]}, \quad a_4^{[1]} = \sigma(z_4^{[1]})$$

- Algorithm

Given input x :

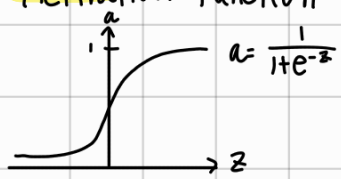
$$z^{[1]} = W^{[1]} \cdot x^{=a^{[0]}} + b^{[1]}$$
$$a^{[1]} = \sigma(z^{[1]})$$
$$z^{[2]} = W^{[2]} a^{[1]} + b^{[2]}$$
$$a^{[2]} = \sigma(z^{[2]})$$

- multiple examples

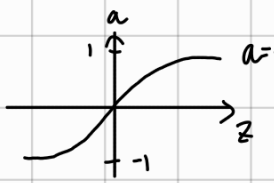
for $i = 1$ to m :

$$z^{[1](i)} = W^{[1]} x^{(i)} + b^{[1]}$$
$$a^{[1](i)} = \sigma(z^{[1](i)})$$
$$z^{[2](i)} = W^{[2]} a^{[1](i)} + b^{[2]}$$
$$a^{[2](i)} = \sigma(z^{[2](i)})$$

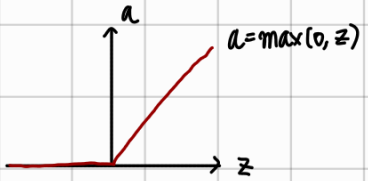
③ Activation function



$a = \frac{1}{1+e^{-z}}$



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



$a = \max(0, z)$

ReLU

$a = \max(0.01z, z)$

Leaky ReLU

- 성능 : Sigmoid < tanh < ReLU < Leaky ReLU

④ Derivatives of activation functions

i) Sigmoid $g(z) = \frac{1}{1+e^{-z}}$

$$\frac{d}{dz} g(z) = \text{slope of } g(x) \text{ at } z = \frac{1}{1+e^{-z}}\left(1 - \frac{1}{1+e^{-z}}\right)$$

$$= g(z)(1-g(z)) = a(1-a)$$

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

$g'(z) = \frac{d}{dz} g(z) = \text{slope of } g(z) \text{ at } z = 1-(\tanh(z))^2$

iii) ReLU & Leaky ReLU

1) ReLU: $g(z) = \max(0, z)$

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

2) Leaky ReLU $= \max(0.01z, z)$

$$g'(z) = \begin{cases} 0.01 & \text{if } z < 0 \\ 1 & \text{if } z \geq 0 \end{cases}$$

⑤ Gradient descent for neural networks

- Parameters: $W^{[1]}$, $b^{[1]}$, $W^{[2]}$, $b^{[2]}$

  $(n^{[1]}, n^{[0]})$ $(n^{[1]}, 1)$ $(n^{[2]}, n^{[1]})$ $(n^{[2]}, 1)$

  $n_x = n^{[0]}, n^{[1]}, n^{[2]} = 1$

- Cost Function: $J(W^{[1]}, b^{[1]}, W^{[2]}, b^{[2]}) = \frac{1}{m}\sum_{i=1}^{n} L(\hat{y}, y)$

- Gradient descent:

  Repeat {

  Compute predicts $(\hat{y}^{(i)}, i=1, \dots, m)$

  $dW^{[1]} = \frac{\partial J}{\partial W^{[1]}}$, $db^{[1]} = \frac{\partial J}{\partial b^{[1]}}$, $\dots$

  $W^{[1]} = W^{[1]} - \alpha dW^{[1]}$

  $b^{[1]} = b^{[1]} - \alpha db^{[1]}$ }

- Formulas for computing derivatives

1) Forwal Propagation

$$z^{[1]} = W^{[1]}X + b^{[1]}$$

$$A^{[1]} = g^{[1]}(z^{[1]})$$

$$z^{[2]} = W^{[2]}A^{[1]} + b^{[2]}$$

$$A^{[2]} = g^{[2]}(z^{[2]}) = \sigma(z^{[2]})$$

2) Back propagation

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

$$dW^{[2]} = \frac{1}{m} dz^{[2]} A^{[1]T}$$

$$db^{[2]} = \frac{1}{m} np.sum(dz^{[2]}, axis=1)$$

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

$$dW^{[1]} = \frac{1}{m} dz^{[1]} X^T$$

$$db^{[1]} = \frac{1}{m} np.sum(dz^{[1]}, axis=1)$$

⑥ Random Initialization

- W : random, b : 0으로 초기화

  ex) $W^{[1]} = np.random.randn((2,2)) * 0.01$

  $b^{[1]} = np.zero((2,1))$

- 심층 신경망은 0.01 아님.

✡ Quiz

- hidden layer에서는 sigmoid보다 tanh를 사용하는 것이 더 효과적
- `x = np.random.rand(4,5)`

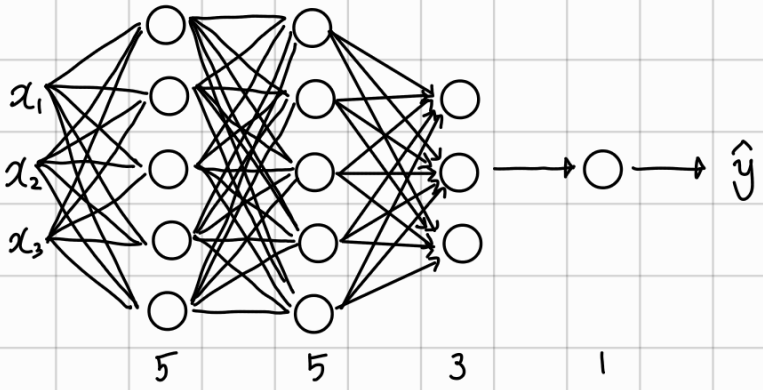
  `y = np.sum(x, axis=1)`

  ⇒ 배열의 각 행에 대해 합이 계산됨.

  `keepdims=True`를 사용하면 두 번째 차원 유지
- identity, 선형 활성화 함수 g(c)=C 사용 시 레이어 구성의 출력은 단일 레이어 수행 결과와 동일

Deep Neural Networks

① Deep L-layer Neural Network



- $L = 4$ (# layers)
- $n^{[l]}$ = # units in layer l, ex) $n^{[1]}=5$, $n^{[2]}=5$, $n^{[3]}=3$, $n^{[4]}=n^{[L]}=1$
- $n^{[0]} = n_x = 3$ → input
- $a^{[l]}$ = activations in layer l $= g^{[l]}(z^{[l]})$, ex) $x=a^{[0]}$, $\hat{y}=a^{[L]}$
- $w^{[l]}$ = weight for $z^{[l]}$

② Forward Propagation in a Deep Network.

- X : $z^{[1]} = w^{[1]}x + b^{[1]}$

  $a^{[1]} = g^{[1]}(z^{[1]})$

  $z^{[2]} = w^{[2]}a^{[1]} + b^{[2]}$

  $a^{[2]} = g^{[2]}(z^{[2]})$

  ⋮

  $z^{[4]} = w^{[4]}a^{[3]} + b^{[4]}$, $a^{[4]} = g^{[4]}(z^{[4]}) = \hat{y}$

  $z^{[l]} = w^{[l]}a^{[l-1]} + b^{[l]}$, $a^{[l]} = g^{[l]}(z^{[l]})$

- Vectorization

  for $l=1,\dots,4$

  $z^{[1]} = w^{[1]}X + b^{[1]} = w^{[1]}A^{[0]} + b^{[1]}$

  $A^{[1]} = g^{[1]}(z^{[1]})$

  $z^{[2]} = w^{[2]}A^{[1]} + b^{[1]}$

  $A^{[2]} = g^{[2]}(z^{[2]})$

  $\hat{Y} = g(z^{[4]}) = A^{[4]}$

③ Matrix size

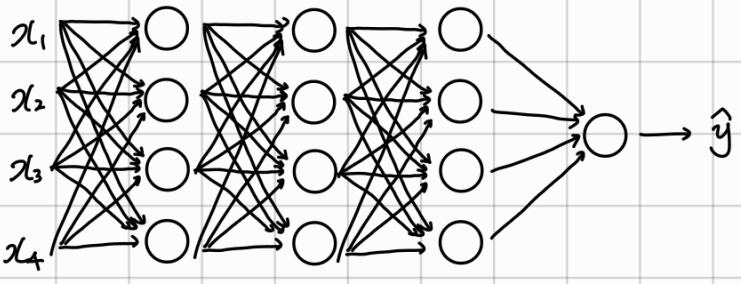
- $w^{[l]}$, $dw^{[l]}$: $(n^{[l]}, n^{[l-1]})$
- $b^{[l]}$, $db^{[l]}$: $(n^{[l]}, 1)$
- $z^{[1]} = w^{[1]} \cdot x + b^{[1]} \Rightarrow (n^{[1]}, 1) = (n^{[1]}, n^{[0]})(n^{[0]}, 1) + (n^{[1]}, 1)$

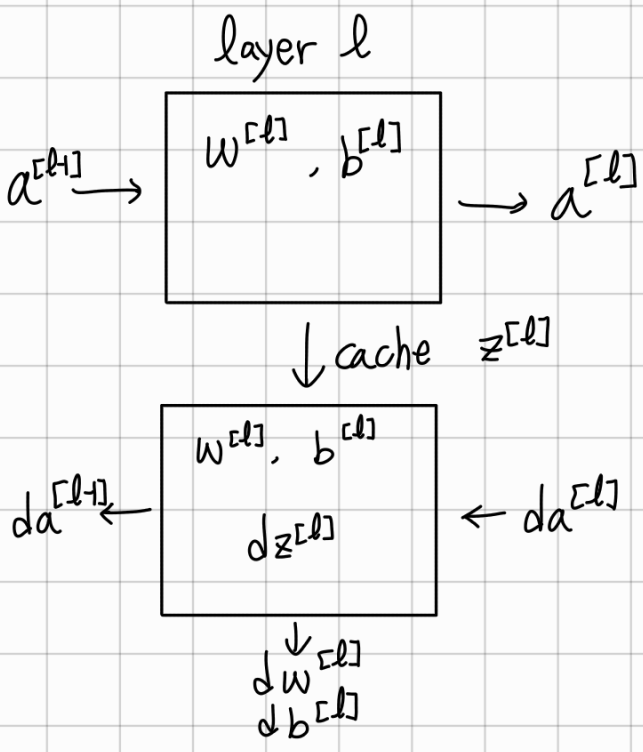
  ↳ 훈련세트의 경우 $(n^{[1]}, m) = (n^{[1]}, n^{[0]})(n^{[0]}, m) + (n^{[1]}, 1)$
- $z^{[l]}$, $a^{[l]}$: $(n^{[l]}, 1)$

  $Z^{[l]}$, $A^{[l]}$: $(n^{[l]}, m)$, $l = 0$ $A^{[0]} = X = (n^{[0]}, m)$

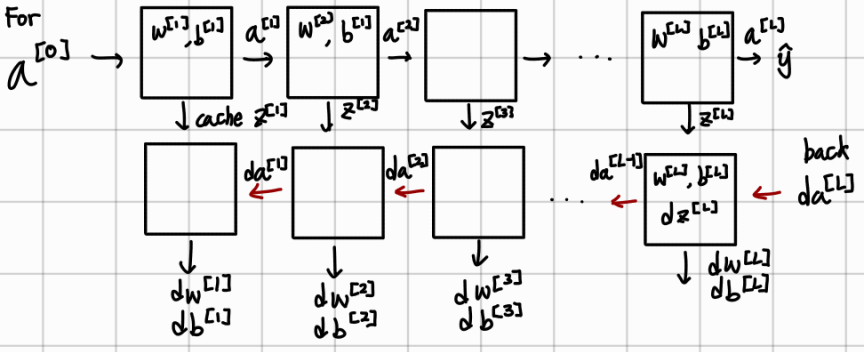
  $dz^{[l]}$, $dA^{[l]}$: $(n^{[l]}, m)$

④ Building blocks of deep neural networks





- layer l : $w^{[l]}$, $b^{[l]}$
- Forwal : Input $a^{[l-1]}$, output $a^{[l]}$
- $z^{[l]} : w^{[l]} a^{[l-1]} + b^{[l]}$ cache $z^{[l]}$
- $a^{[l]} : g^{[l]}(z^{[l]})$
- Backwal : Input $da^{[l]}$, output $da^{[l-1]}$



⑤ Forward and backward propagation

1) Forward propagation

Input : $a^{[l-1]}$

Output : $a^{[l]}$, cache ($z^{[l]}$)

$$z^{[l]} = w^{[l]} \cdot a^{[l-1]} + b^{[l]} \qquad Z^{[l]} = W^{[l]} A^{[l-1]} + b^{[l]}$$

$$a^{[l]} = g^{[l]}(z^{[l]}) \qquad A^{[l]} = g^{[l]}(Z^{[l]})$$

2) Backward propagation

Input : $da^{[l]}$

Output : $da^{[l-1]}$, $dW^{[l]}$, $db^{[l]}$

$$dz^{[l]} = da^{[l]} * g^{[l]\prime}(z^{[l]}) \qquad dZ^{[L]} = A^{[L]} - Y$$

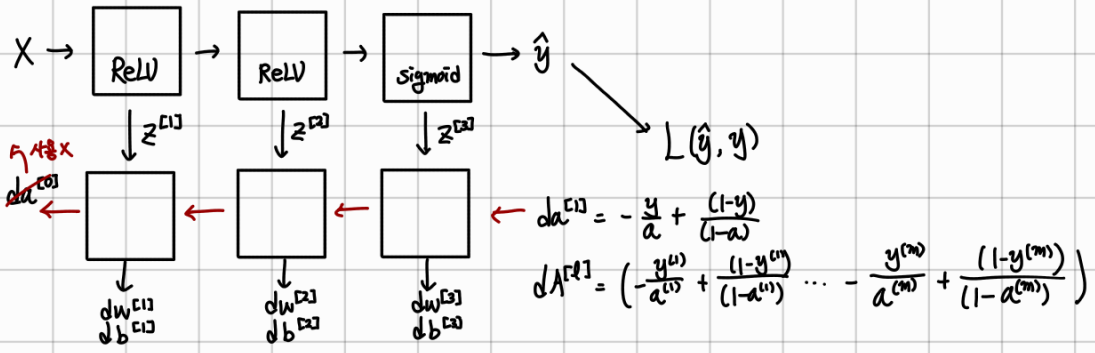
$$dw^{[l]} = dz^{[l]} \cdot a^{[l-1]T} \qquad dW^{[L]} = \frac{1}{m} dZ^{[L]} A^{[L-1]T}$$

$$db^{[l]} = dz^{[l]} \qquad db^{[L]} = \frac{1}{m} np.sum(dZ^{[l]}, axis=1, keepdims=True)$$

$$da^{[l-1]} = w^{[l]T} \cdot dz^{[l]} \qquad dz^{[L-1]} = W^{[L]T} \cdot dz^{[L]} * g'^{[L-1]}(z^{[L-1]})$$

$$dz^{[l]} = w^{[l+1]T} dz^{[l+1]} * g^{[l]\prime} z^{[l]}$$

3) Summary



⑥ Parameter vs Hyperparameters

- Parameters : $w^{[1]}$, $b^{[1]}$, $w^{[2]}$, $b^{[2]}$, ...
- Hyperparameters : learning rate, # iterations, # hidden layers L,
  # hidden units $n^{[1]}$, $n^{[2]}$, ..., Choice of activation function