Deep NN

- Notation
  - N : # layers
  - $n^{[l]}$: # units in layer l
  - $a^{[l]}$: activations in layer l
  - $W^{[l]}$: weights for $z^{[l]}$

ex.

$L = 4$

$n^{[1]} = 5$, $n^{[2]} = 5$ $n^{[3]} = 3$, $n^{[4]} = 1$

$n^{[0]} = n_x = 3$

<Forward Propagation>

$$Z^{[l]} = W^{[l]} * A^{[l]} + b^{[l]}$$

$$A^{[l]} = g^{[l]}(Z^{[l]})$$

$$(A^{[0]} = X)$$

<Backward Propagation>

$$dZ^{[l]} = dA^{[l]} * g^{[l]\prime}(Z^{[l]})$$

$$dW^{[l]} = dZ^{[l]} \cdot A^{[l-1]}$$

$$db^{[l]} = dZ^{[l]}$$

$$dA^{[l-1]} = W^{[l]T} \cdot dZ^{[l]}$$

$$dZ^{[l]} = W^{[l+1]T} dZ^{[l+1]} * g^{[l]\prime}(Z^{[l]})$$

(vectorized ver.)

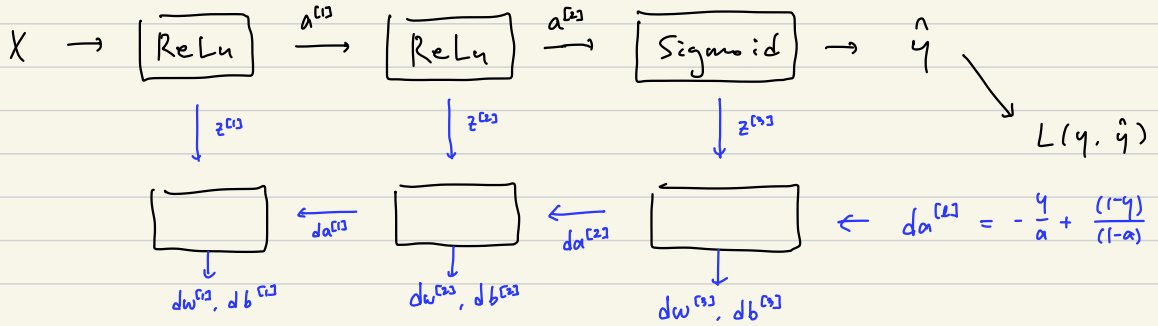
$$dZ^{[l]} = dA^{[l]} * g^{[l]\prime}(Z^{[l]})$$

$$dW^{[l]} = \frac{1}{m} dZ^{[l]} \cdot A^{[l-1]T}$$

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

$$dA^{[l-1]} = W^{[l]T} \cdot dZ^{[l]}$$

<Forward / Backward Propagation 과정>



- Forward
  - input : $a^{[l-1]}$ / output : $a^{[l]}$, $a^{[0]} = X$
  - $Z^{[l]}$, $W^{[l]}$, $b^{[l]}$을 캐시에 저장

- Backward
  - input : $da^{[l]}$ / output : $da^{[l-1]}$
  - da를 업데이트하기 위해 $dW^{[l]}$, $db^{[l]}$ 계산
  - 이를 위해 캐시에 저장된 z, w, b 사용.

〈 Forward propagation in DNN 〉

- single training sample x.

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

- vectorized ver.

for $l = 1, 2, \cdots$

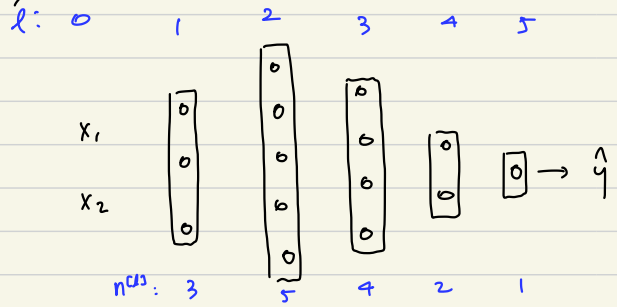
$$\begin{cases} Z^{[l]} = W^{[l]} \cdot A^{[l-1]} + b^{[l]} \\ A^{[l]} = g^{[l]}(Z^{[l]}) \end{cases}$$

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

← 연방향으로 stack

- $l = 1, 2, \cdots$ 에 대한 for loop 사용
- 하지만 어쩔 수 없음..

〈 행렬 차원 계산 〉



$$z^{[1]} = w^{[1]} \cdot x + b^{[1]}$$
$(n^{[1]}, 1)$ $(3, 2)$ $(n^{[0]}, 1)$
$(3, 1)$ $(2, 1)$

$$z^{[2]} = w^{[2]} \cdot a^{[1]} + b^{[2]}$$
$(n^{[2]}, 1)$ $(n^{[2]}, n^{[1]})$ $(n^{[1]}, 1)$
$(5, 1)$ $(5, 3)$ $(3, 1)$

- generalize

$$\begin{cases} w^{[l]} : (n^{[l]}, n^{[l-1]}) \\ b^{[l]} : (n^{[l]}, 1) \end{cases} \qquad \begin{cases} dw^{[l]} : w^{[l]} \text{와 같음} \\ db^{[l]} : b^{[l]} \text{ 〃} \end{cases}$$

- $z^{[l]}$, $a^{[l]}$의 차원은 같음

<벡터화 행렬 차원 계산>

$$Z^{[1]} = W^{[1]} \cdot X + b^{[1]}$$

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

- generalize

$$\begin{cases} Z^{[l]},\ A^{[l]} : (n^{[l]}, m) \\ dZ^{[l]},\ dA^{[l]} : (n^{[l]}, m) \end{cases}$$

(m: # training set)

<Building DNN>

$$W^{[l]} := W^{[l]} - \alpha dW^{[l]}$$
$$b^{[l]} := b^{[l]} - \alpha db^{[l]}$$

<변수 vs. 하이퍼 파라미터>

- Parameters : $W^{[l]}$, $b^{[l]}$
- Hyperparameters : learning rate α
  # iterations
  # hidden layers
  # hidden units
  activation functions.
  momentum, mini-batch, etc...

  } control W, b

- DL은 경험적이다
  - 여러 하이퍼 파라미터를 조절하며 손실 함수가 최소가 되는 값을 찾아야 함.
- DL은 구조적 데이터에 많이 적용된다.