딥러닝 1단계 4주차

이승목

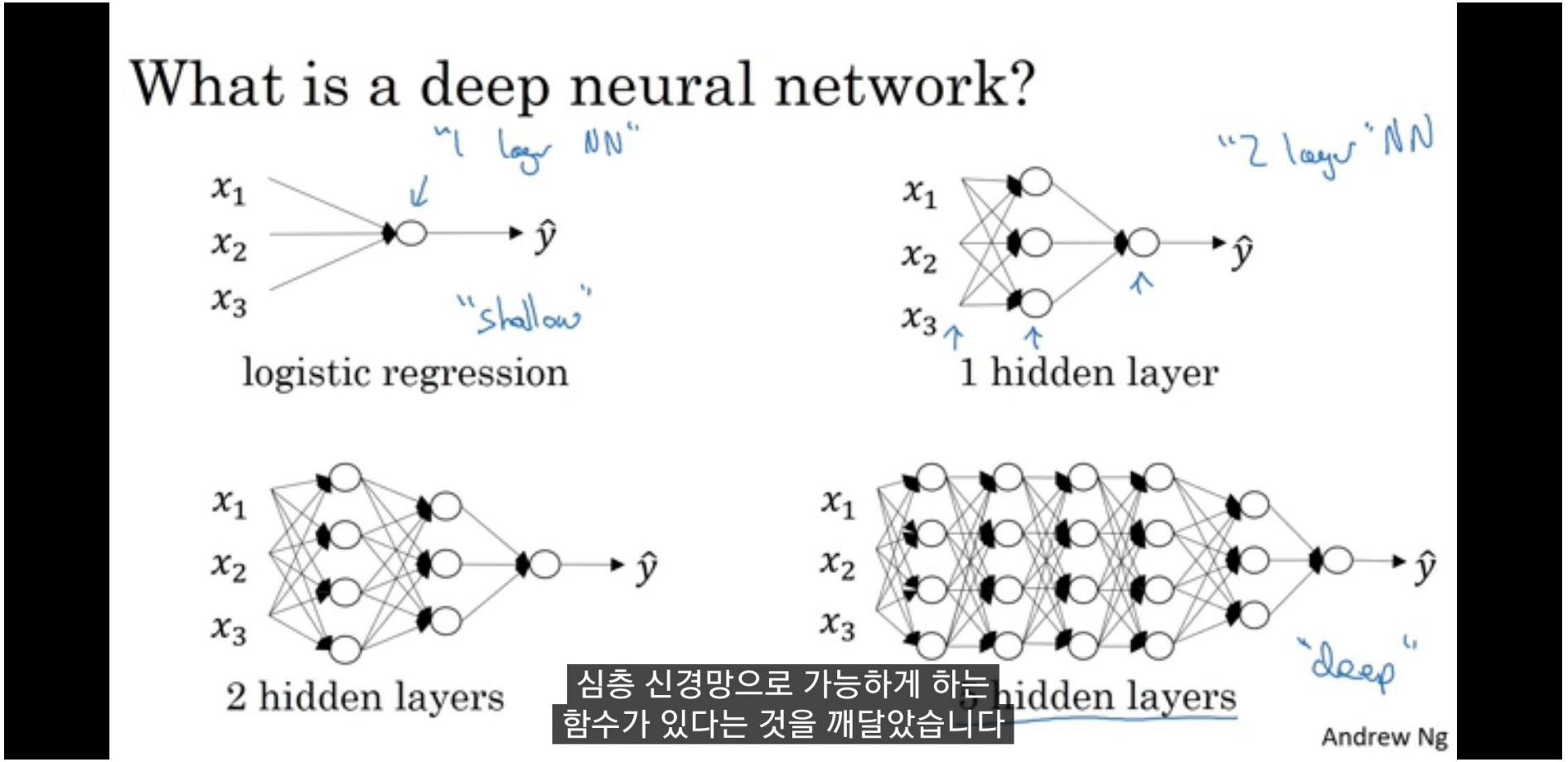
2020.05.26.

범위

- Edwith에 순서가 잘못 업로드됨.

- 발표자료 순서는 강의 순서와도 또 다름

드디어 딥러닝



심층 신경망으로 가능하게 하는 함수가 있다는 것을 깨달았습니다

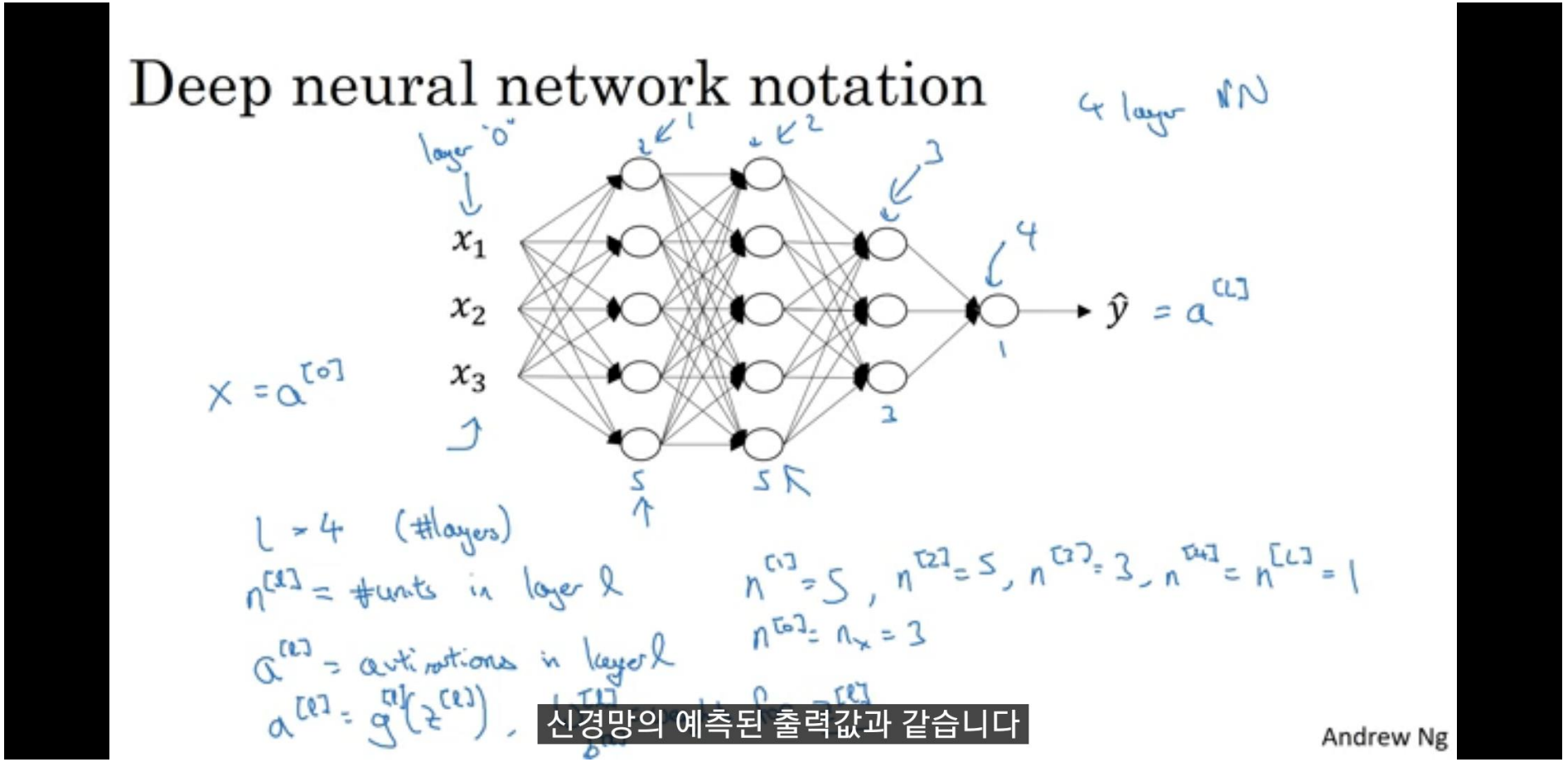
데이터 표기법

- 데이터: (x, y). $x \in \mathbb{R}^{n_x}, y \in \{0, 1\}$
- m: 훈련 데이터 수. $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \cdots, (x^{(m)}, y^{(m)})\}$
  - 경우에 따라 m_{train}, m_{test} 등으로 표기하기도
- 데이터 행렬 $X = \begin{pmatrix} | & & | \\ x^{(1)} & \cdots & x^{(m)} \\ | & & | \end{pmatrix}$. $X \in \mathbb{R}^{n_x \times m}$
- $Y = (y^{(1)}, \cdots, y^{(m)})$. $Y \in \mathbb{R}^{1 \times m}$

딥러닝 표기법

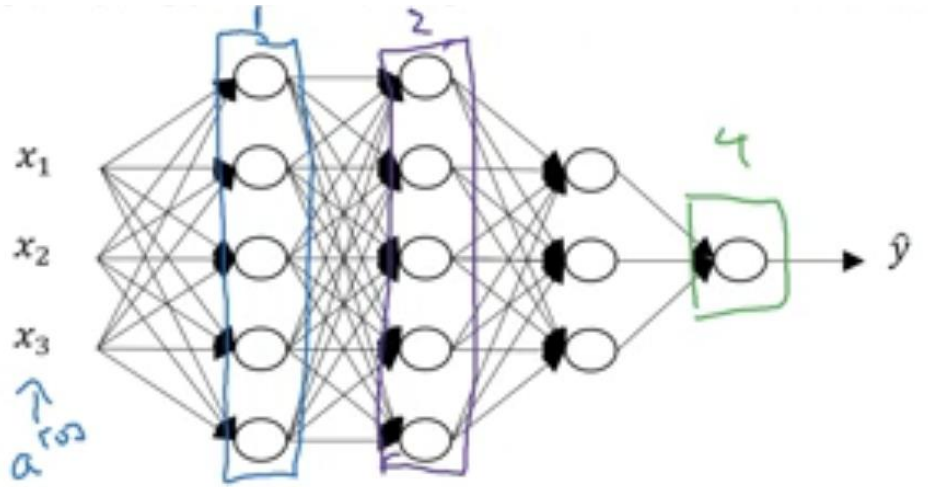
- 네트워크 층의 수: L
- l번째 층에 있는 유닛 개수: $n^{[l]}$
- l번째 층의 가중치 행렬: $w^{[l]}$
- l번째 층의 bias 벡터: $b^{[l]}$
- l번째 층의 가중치 행렬을 곱한 결과: $z^{[l]}$
- l번째 층의 활성화함수: $g^{[l]}$
- l번째 층의 활성값: $a^{[l]}$
  - 입력 데이터 $x^{(i)} = a^{[0]}$
  - 예측된 출력값 $\hat{y} = a^{[L]}$

딥러닝 표기법


Deep neural network notation
4 layer NN
layer "0"
x_1
x_2
x_3
$\hat{y} = a^{[L]}$
$X = a^{[0]}$
$L = 4$ (#layers)
$n^{[l]}$ = #units in layer l
$a^{[l]}$ = activations in layer l
$a^{[l]} = g^{[l]}(z^{[l]})$
$n^{[1]} = 5, n^{[2]} = 5, n^{[3]} = 3, n^{[4]} = n^{[L]} = 1$
$n^{[0]} = n_x = 3$
신경망의 예측된 출력값과 같습니다
Andrew Ng

딥러닝에서 Forward Propagation

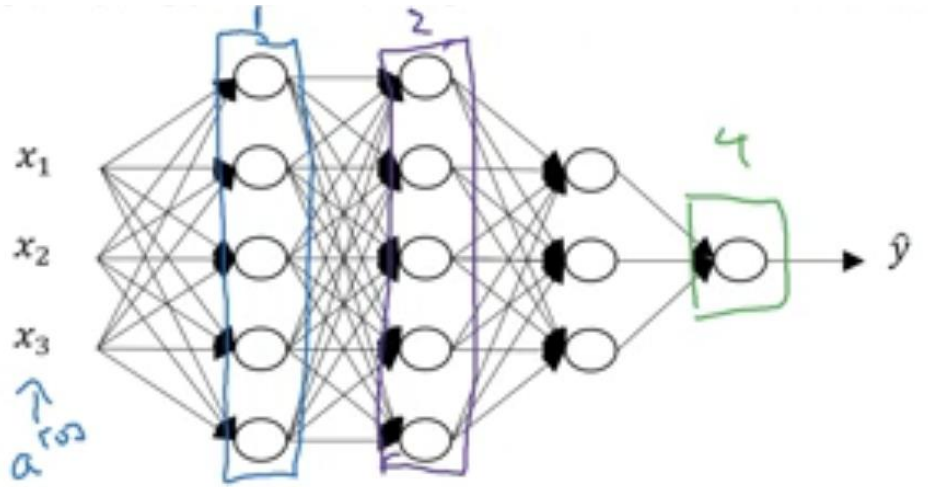
- $z^{[1]} = w^{[1]} \cdot a^{[0]} + b^{[1]}$
- $a^{[1]} = g^{[1]}(z^{[1]})$



딥러닝에서 Forward Propagation

- $z^{[1]} = w^{[1]} \cdot a^{[0]} + b^{[1]}$
- $a^{[1]} = g^{[1]}(z^{[1]})$

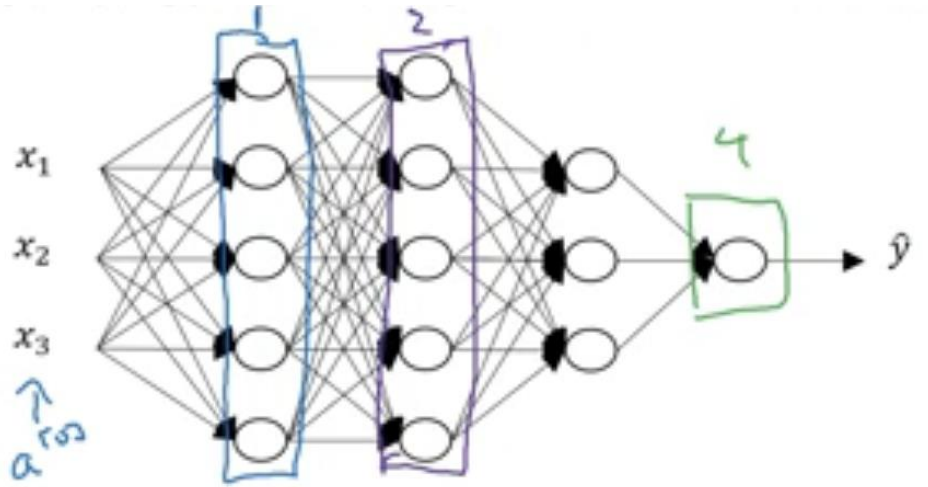
- $z^{[l]} = w^{[l]} \cdot a^{[l-1]} + b^{[l]}$
- $a^{[l]} = g^{[l]}(z^{[l]})$



딥러닝에서 Forward Propagation (Vec)

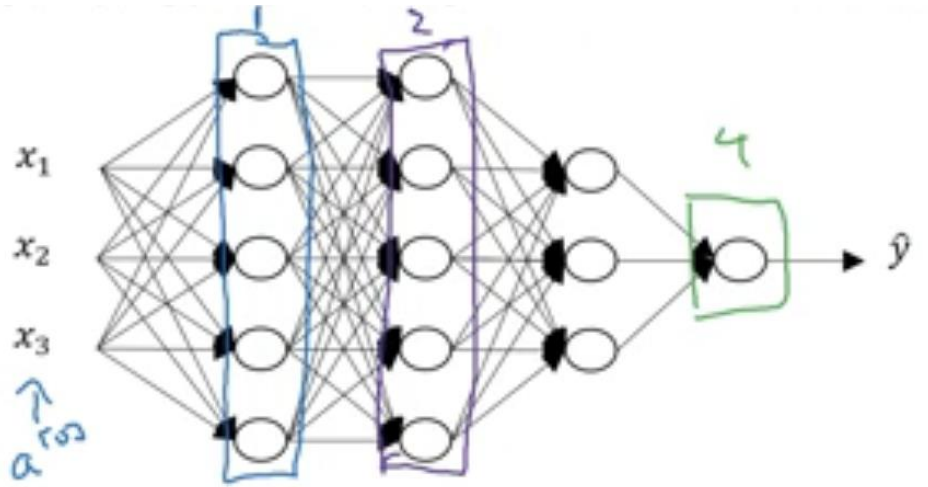
- Vectorization: $X = \left(x^{(1)} \cdots x^{(m)}\right)$
  - $A^{[0]} = X$

- $Z^{[l]} = w^{[l]} \cdot A^{[l-1]} + b^{[l]}$
- $A^{[l]} = g^{[l]}\left(Z^{[l]}\right)$
  - $g^{[l]}$은 element-wise하게 작용.



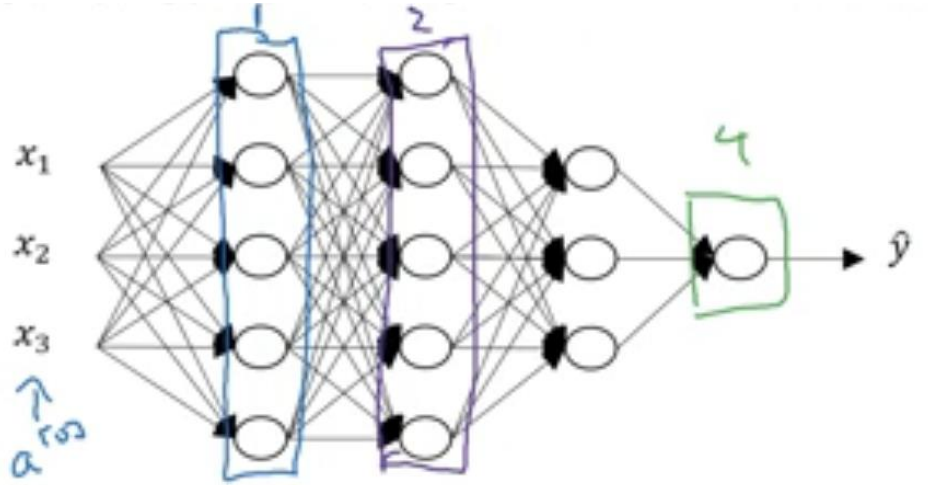
딥러닝에서 Forward Propagation (Vec)

- Vectorization: $X = \left(x^{(1)} \cdots x^{(m)}\right)$
  - $A^{[0]} = X$
- for $l = 1 \sim 4$:
  - $Z^{[l]} = w^{[l]} \cdot A^{[l-1]} + b^{[l]}$
  - $A^{[l]} = g^{[l]}\left(Z^{[l]}\right)$
- $\hat{Y} = g\left(Z^{[4]}\right) = A^{[4]}$

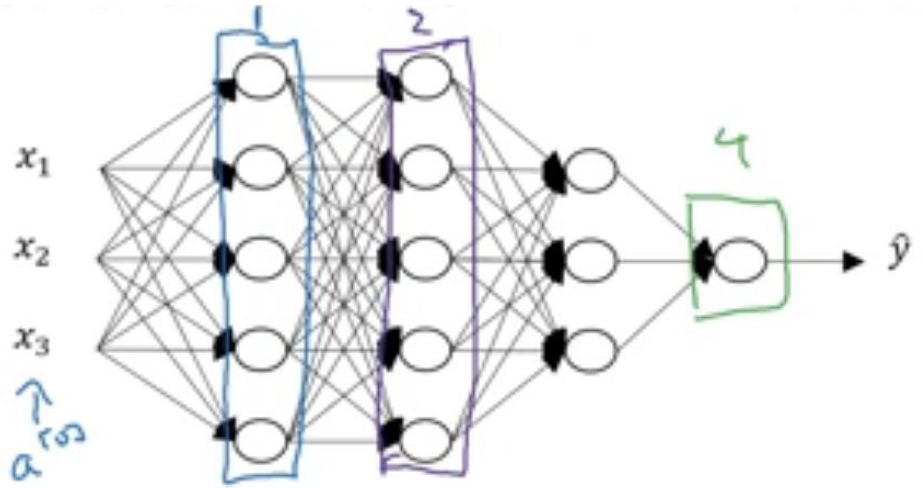


딥러닝에서 Backward Propagation

- $dz^{[l]} = da^{[l]} * g^{[l]'}(z^{[l]})$
  - $*$는 element-wise 곱
- $dw^{[l]} = dz^{[l]} \cdot a^{[l-1]}$
- $db^{[l]} = dz^{[l]}$
- $da^{[l-1]} = w^{[l]^T} \cdot dz^{[l]}$
  - $da^{[L]} = -\frac{y}{a} + \frac{1-y}{1-a}$

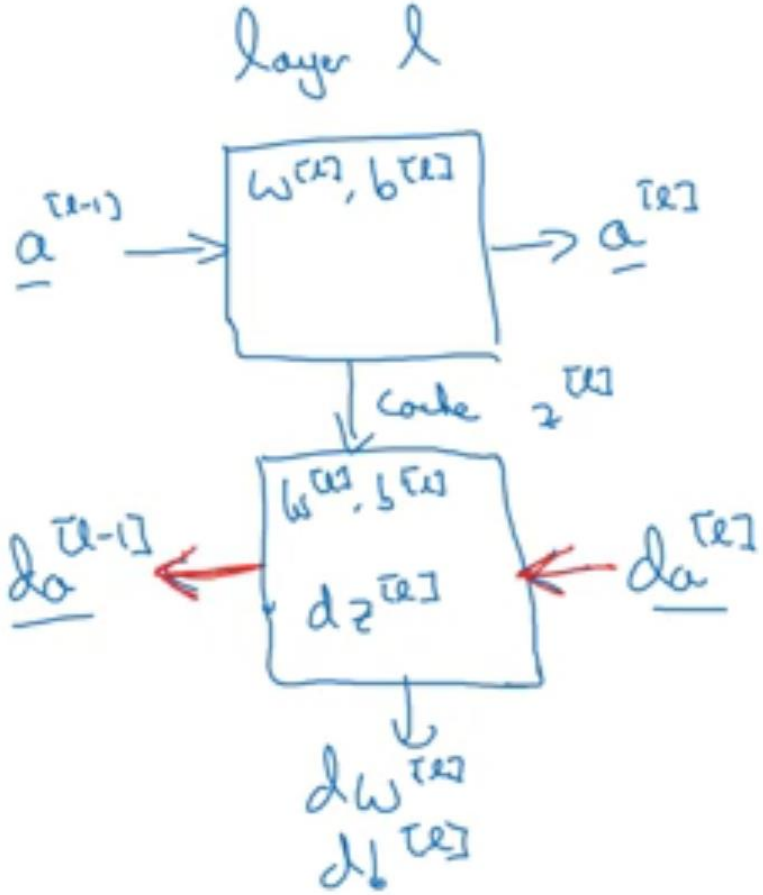


딥러닝에서 Backward Propagation (Vec)

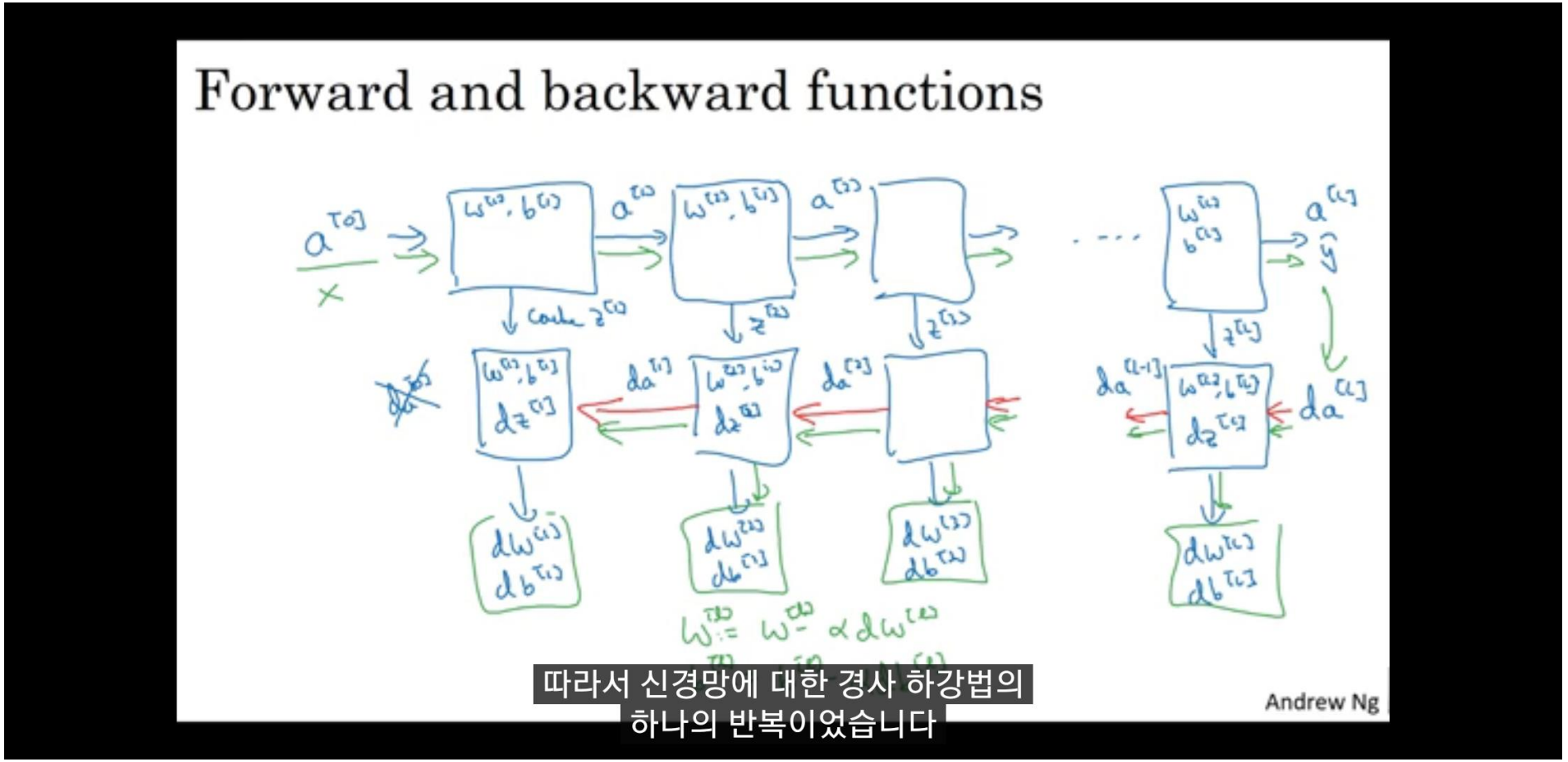


- $dZ^{[l]} = dA^{[l]} * g^{[l]'}(Z^{[l]})$
  - $*$는 element-wise 곱
- $dw^{[l]} = \frac{1}{m} dZ^{[l]} \cdot A^{[l-1]^T}$
- $db^{[l]} = \frac{1}{m} \text{np.sum}(dZ^{[l]}, \text{axis} = 1, \text{keepdims} = \text{True})$
- $dA^{[l-1]} = w^{[l]^T} \cdot dZ^{[l]}$

딥러닝 데이터 흐름 – 레이어



딥러닝 데이터 흐름 – 네트워크

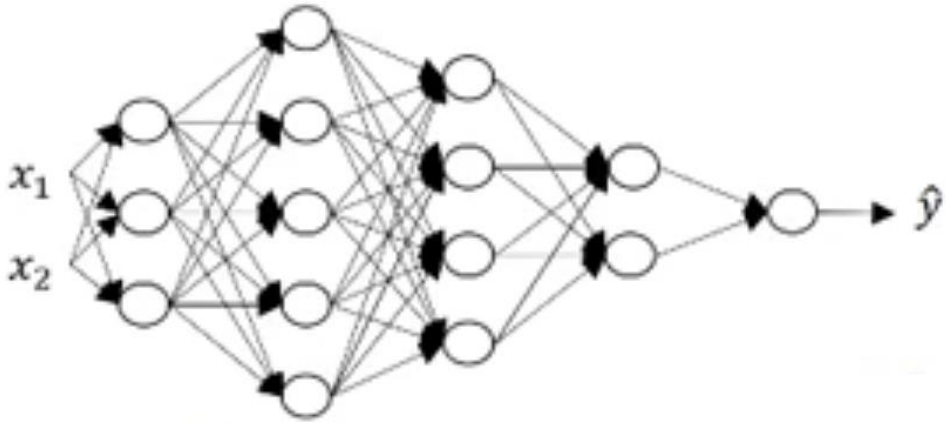


☆ 딥러닝 데이터 흐름 – 텐서플로우

```
# Layer using build
class LinearBuild(Layer):  # inherit Layer
    """y = w.x + b"""

    def __init__(self, units=32):
        super(LinearBuild, self).__init__()
        self.units = units

    def build(self, input_shape):
        self.w = self.add_weight(shape=(input_shape[-1], self.units),
                                 initializer='random_normal',
                                 trainable=True)
        self.b = self.add_weight(shape=(self.units,),
                                 initializer='random_normal',
                                 trainable=True)

    def call(self, inputs, **kwargs):
        return tf.matmul(inputs, self.w) + self.b
```

차원 확인

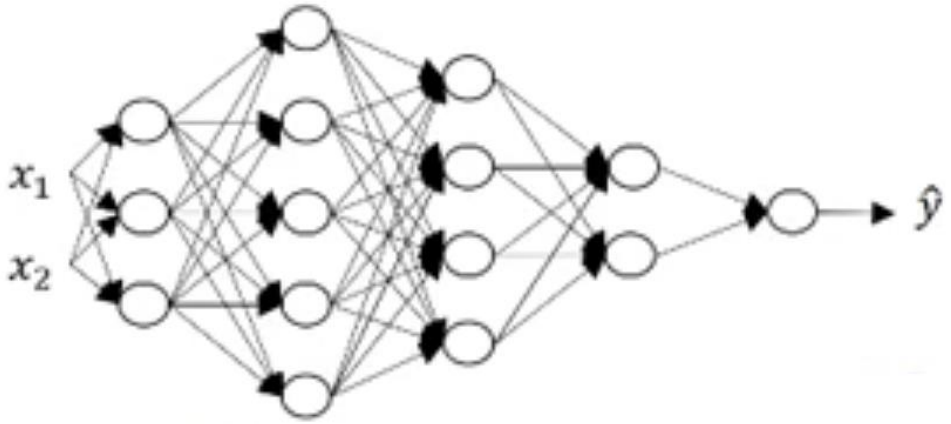
- $x = a^{[0]}: \left(n^{[0]}, 1\right)$
- $z^{[l]}, a^{[l]}: \left(n^{[l]}, 1\right)$
  - $dz^{[l]}, da^{[l]}: \left(n^{[l]}, 1\right)$
- $w^{[l]}: \left(n^{[l]}, n^{[l-1]}\right)$
  - $dw^{[l]}: \left(n^{[l]}, n^{[l-1]}\right)$
- $b^{[l]}: (n^{[l]}, 1)$
  - $db^{[l]}: (n^{[l]}, 1)$



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

차원 확인 (Vec)

- $X = A^{[0]}: \left(n^{[0]}, m\right)$
- $Z^{[l]}, A^{[l]}: \left(n^{[l]}, m\right)$
  - $dZ^{[l]}, dA^{[l]}: \left(n^{[l]}, m\right)$
- $w^{[l]}: \left(n^{[l]}, n^{[l-1]}\right)$
  - $dw^{[l]}: \left(n^{[l]}, n^{[l-1]}\right)$
- $b^{[l]}: (n^{[l]}, 1)$
  - $db^{[l]}: (n^{[l]}, 1)$



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

매개변수와 초매개변수

Parameters: $W^{[1]}, b^{[1]}, W^{[2]}, b^{[2]}, W^{[3]}, b^{[3]}$ ...

Hyperparameters

learning rate α

#iterations

#hidden layers L

#hidden units $n^{[1]}, n^{[2]}, \ldots$

choice of activation function

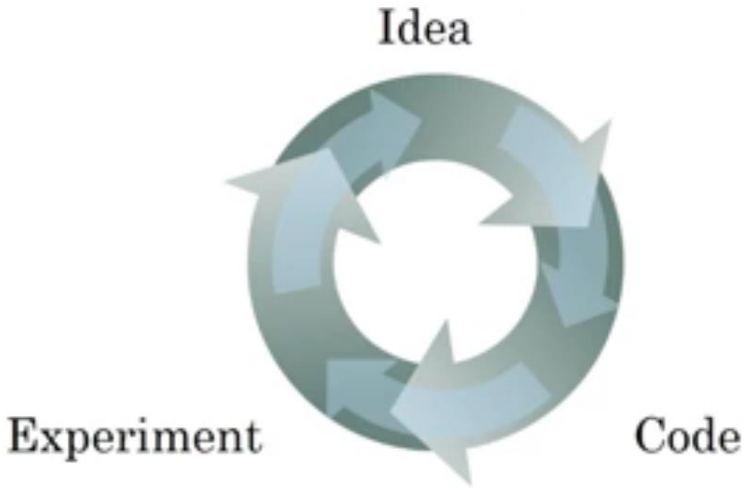
옛날에는 Parameter로 불리기도 했다.

Hyperparameter: Parameter를 결정하는 Parameter

Later: Momentum, mini-batch size, regularizations, ...

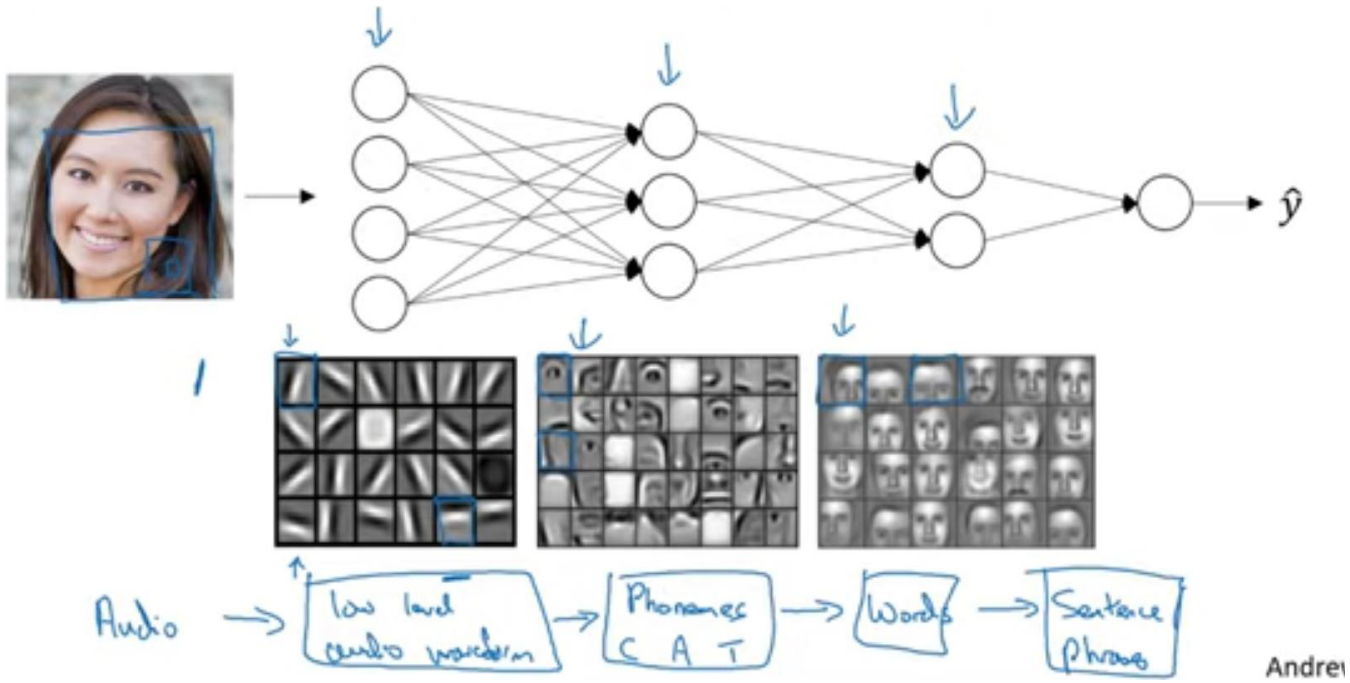
초매개변수 조정

- Empirical process!
- 최적값은 적용 분야에 따라, 사용하는 하드웨어에 따라 다름.
  - 그래서 최적값은 매년 변함.
- 한 번 찾은 최적값도 여러 번에 걸쳐 체크해볼 필요가 있음.
- 체계적으로 그 값을 조정하는 방법은 두 번째 강의에서 다룰 예정.



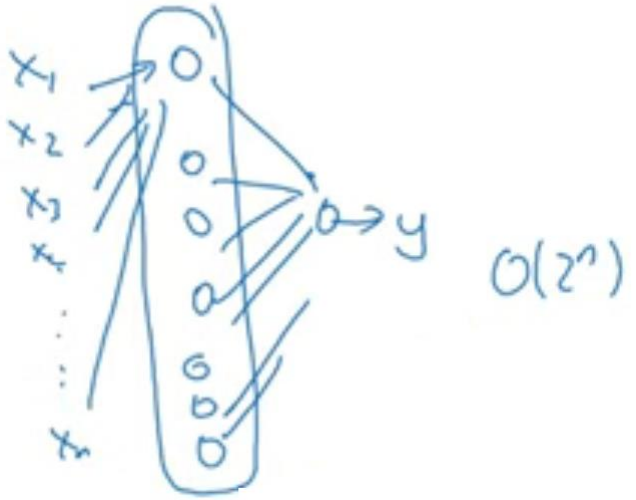
왜 딥러닝은 잘 작동하는가?

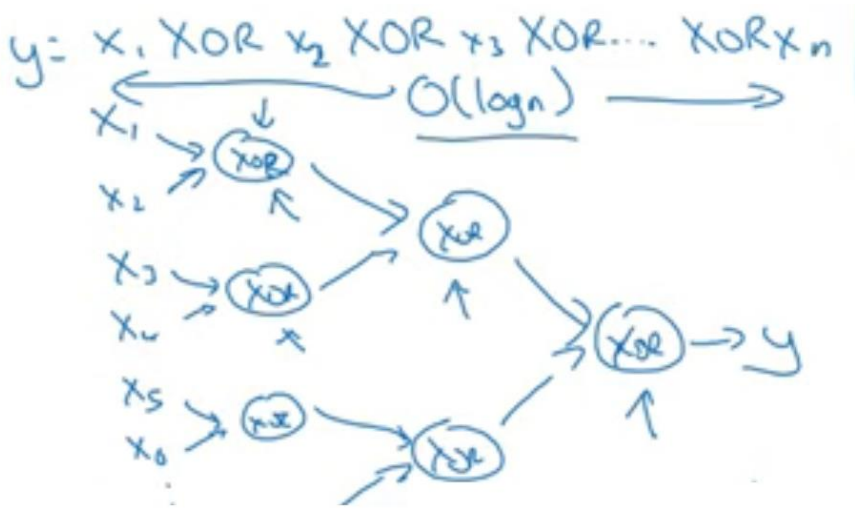
- 직관적인 설명: 층이 쌓여갈 수록 단순한 특성에서부터 복잡한 특성까지 잡아낼 능력을 갖게 된다.



왜 딥러닝은 잘 작동하는가?

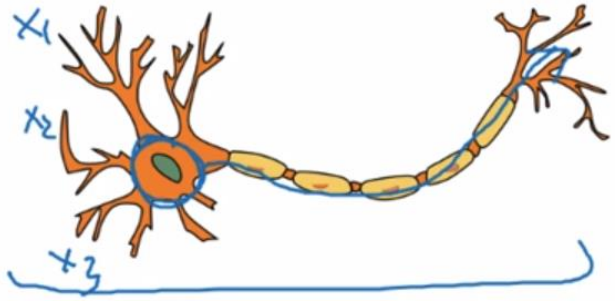
- 회로이론 설명: 깊은 층의 구조로 만들면 더 작게 구현할 수 있는 함수가 있다.
- Ex. $x_1 \otimes x_2 \otimes \cdots \otimes x_n$의 경우 1층으로 구현하려면 $O(2^n)$개의 뉴런이 필요하지만, 여러 층으로 만들면 $O(\log n)$ 층 구조에 $O(n)$개의 뉴런으로 만들 수 있다.





딥러닝과 뇌의 상관관계

- 연관성
  - 한 layer의 forward propagation은 뉴런의 작동 방식과 유사해 보인다.
  - 특히 CNN은 시신경의 작동 방식과 유사하다.
- 불분명한 지점
  - 하나의 뉴런이 작동하는 정확한 방식에 대해서는 잘 알지 못한다.
  - 뇌의 학습 방식에 대해서는 전혀 모른다.
- 딥러닝과 뇌의 비유는 점점 멀어져간다.



다음 스터디

- 딥러닝 1단계 강의 끝
- 다음 스터디부터는 딥러닝 2단계: 심층 신경망 성능 향상시키기를 진행
  - 변수 초기화, 정규화, 미니 배치 경사하강법, RMSprop, Adam 등