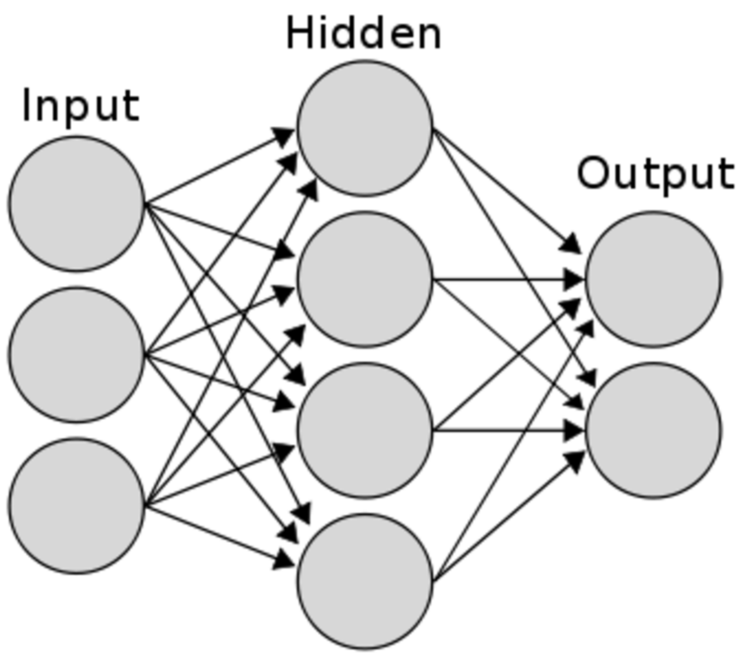
4장. 얕은 신경망 네트워크

1. 신경망 네트워크 개요

- 신경망: 시그모이드 유닛을 쌓아서 만들 수 있음.
- **표기법**
  z^[1] = W^[1]+b^[1]
  → 층(노드값)을 위첨자 [1]로 표현
- **x^(i): i번째 훈련 샘플**
- 로지스틱 회귀: z, a계산 + 마지막에 손실 계산
  신경망: z,a를 **여러번** 계산 + 마지막에 손실 계산

2. 신경망 네트워크의 구성 알아보기

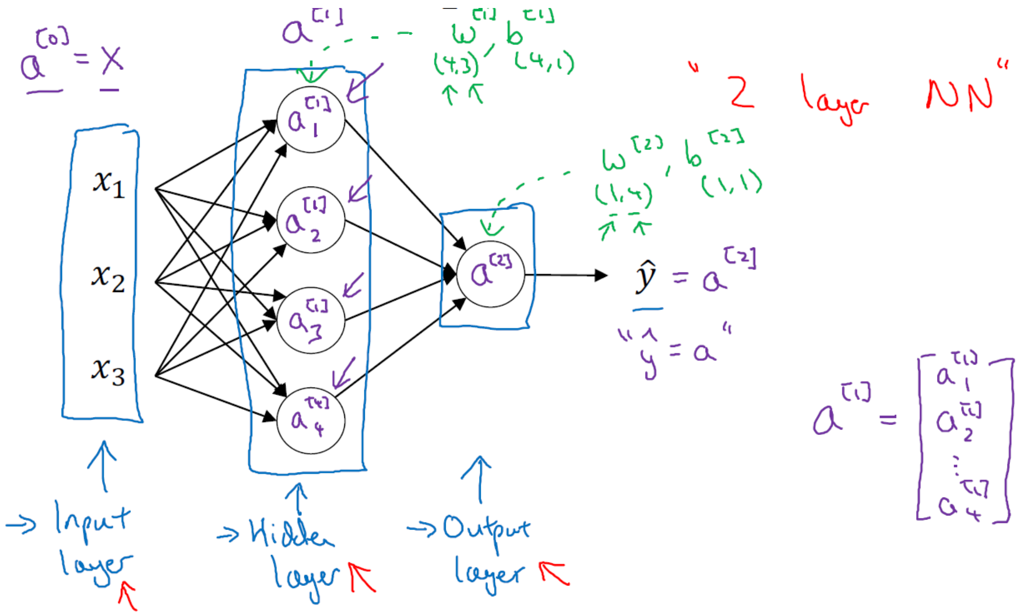
입력 특성 x_1, x_2, x_3: **신경망의 입력 층(Input Layer)**
마지막 층: **출력층**, 예측값인 y_hat 계산 책임



- 지도학습으로 훈련 시키는 신경망: 훈련 세트가 X와 Y로 이루어져 있음.

→ 은닉층의 실제 값은 훈련 세트에 기록X
= **은닉층은** 훈련 세트에서 볼 수 없다.

- 입력값의 다른 표기법: **a^[0] = x**
  → a: 활성값, **신경망의 층들이 다음 층으로 전달해주는 값**
  → a^[0]: 입력층의 활성값
- 은닉층: **a^[1] = 첫 번째 층**
  → 은닉노드가 4개: (1,4) 행렬 (=열벡터)
- 출력층: **a^[2] = 두 번째 층**
  → yhat = a^[2]

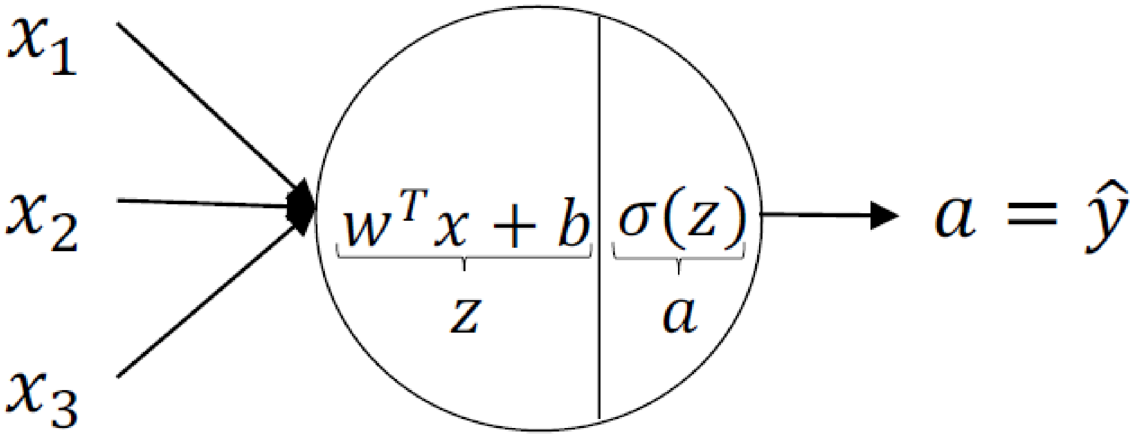


→ 은닉층을 1개 가지는 2층 신경망(입력층은 세지 X)

- w^[1]: (4,3) 행렬 ← 은닉 노드가 4개, 입력 특성이 3개
- b^[1]: (4,1) 벡터
- w^[2]: (1,4) 행렬 ← 은닉 노드가 4개, 출력층 노드가 1개
- b^[2] : (1,1) 벡터

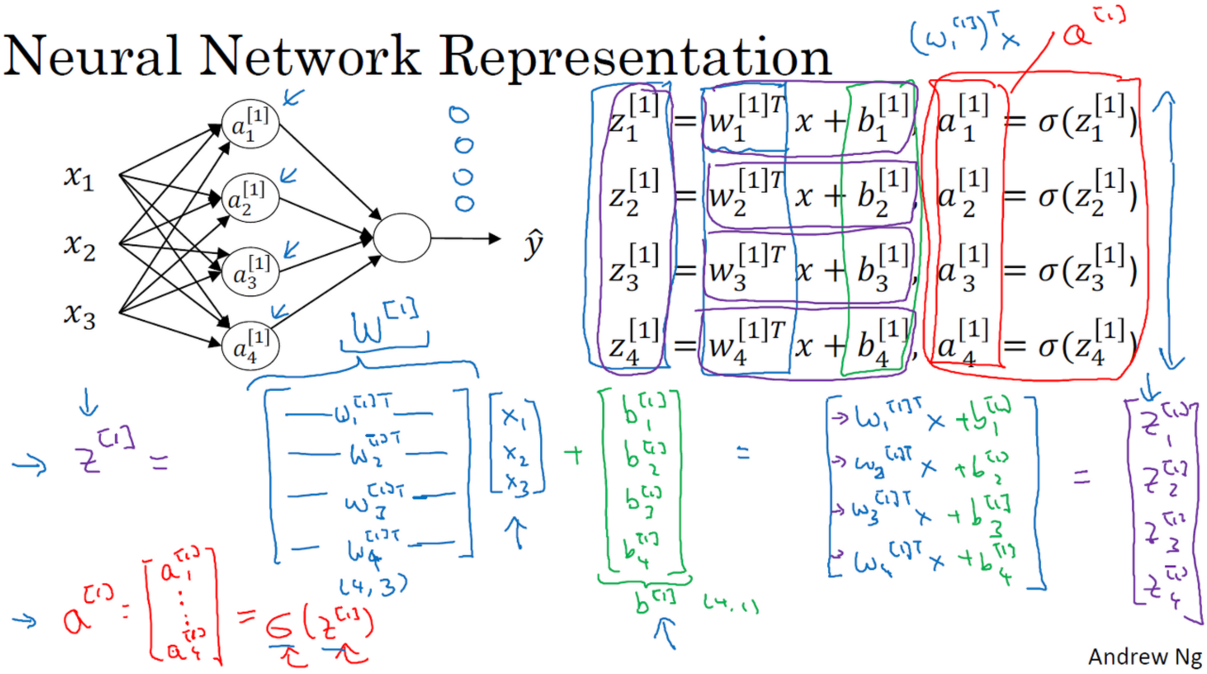
추가적으로 신경망 $a^{[m]}_n$에서 m은 은닉층, 출력층 등인 레이어(layer)를 의미하며, n은 해당 레이어의 구성 요소인 노드(node)를 의미합니다.

3. 신경망 네트워크 출력의 계산



→ **z = w^T*X + b (w=가중치, b=잔차) 계산 +** 활성화 함수 sigmoid 함수가 실행

- 예측값 ŷ
  - 심층 신경망(multi hidden layer)인 경우에는 다음 층(layer)에서 새로운 입력값으로 적용
  - 얕은 신경망(One hidden layer)에서 결과로 출력



- a_i^[l]에서

→ **l: 몇 번째 층인지**

→ **i: 해당 에서 몇 번째 노드인지**

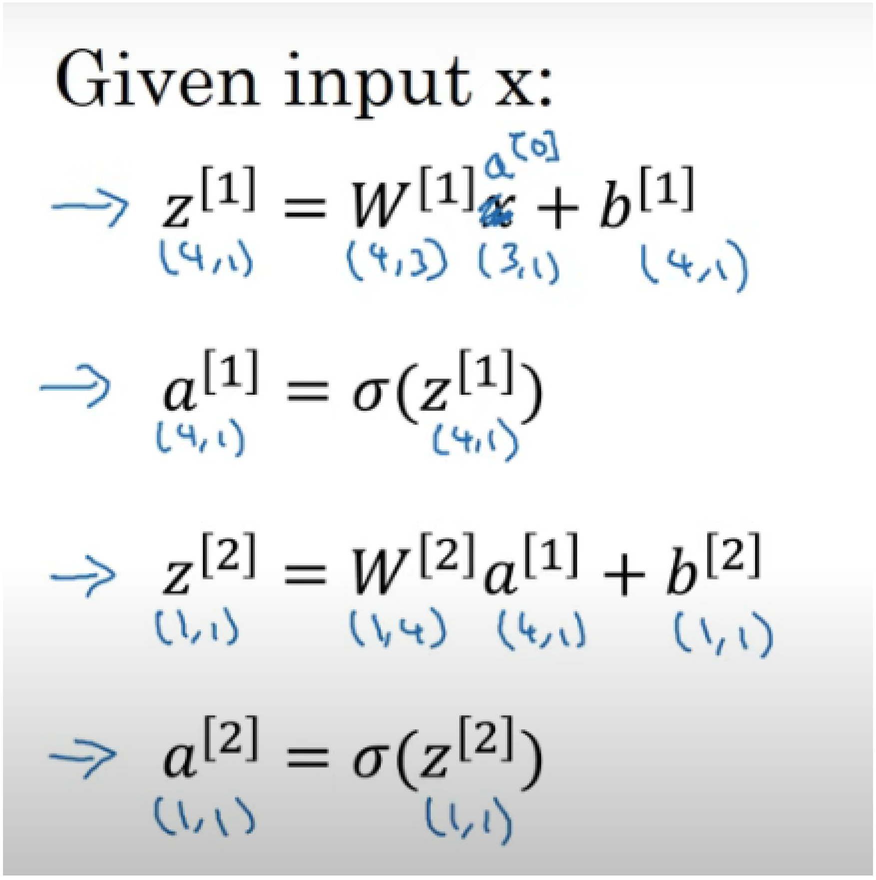
- [1]: 은닉층(Hidden layer), [2]: 출력층(Output layer), [0]: 입력층
- **z^[1]**: 각 z를 열벡터로 쌓은 벡터

! Tip. 벡터화 할 때, 한 층에 노드가 여러 개이면 세로로 쌓기!

- **W^[1]**: w를 쌓아서 만든 (4,3) 행렬
- **b^[1]:** (4,1) 벡터
- **a^[1]**: a_1^[1]부터 a_4^[1]까지 쌓은 벡터

= 시그모이드함수(z^[1]) → z^[1]의 각 원소들의 시그모이드 값 계산

<출력값을 계산하는 벡터화된 구현>



- **x: a^[0], yhat : a^[2]**
- 출력층 매개변수: w^[2], b^[2]
- 마지막 출력 유닛: 로지스틱 회귀와 유사(w대신 (1,4)차원 w^[2] 사용 & b 대신 (1,1) 차원 b^[2] 사용)

4. 많은 샘플에 대한 벡터화

- 훈련 샘플이 m개
  - 첫 훈련 샘플인 x^(1)에 적용하여 첫 샘플의 예측 값인 yhat^(1)=a^2 계산
  - 이 과정을 x^(2),...,x^(m)까지 적용 → yhat^(m)까지 계산
- 표기법
  - a^[2](i)에서 (i): i번째 훈련 샘플, [2]: 두 번째 층 의미
- 모든 훈련 샘플의 예측 값을 벡터화되지 않은 방법으로 계산한다면?

for i = 1 to m:

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

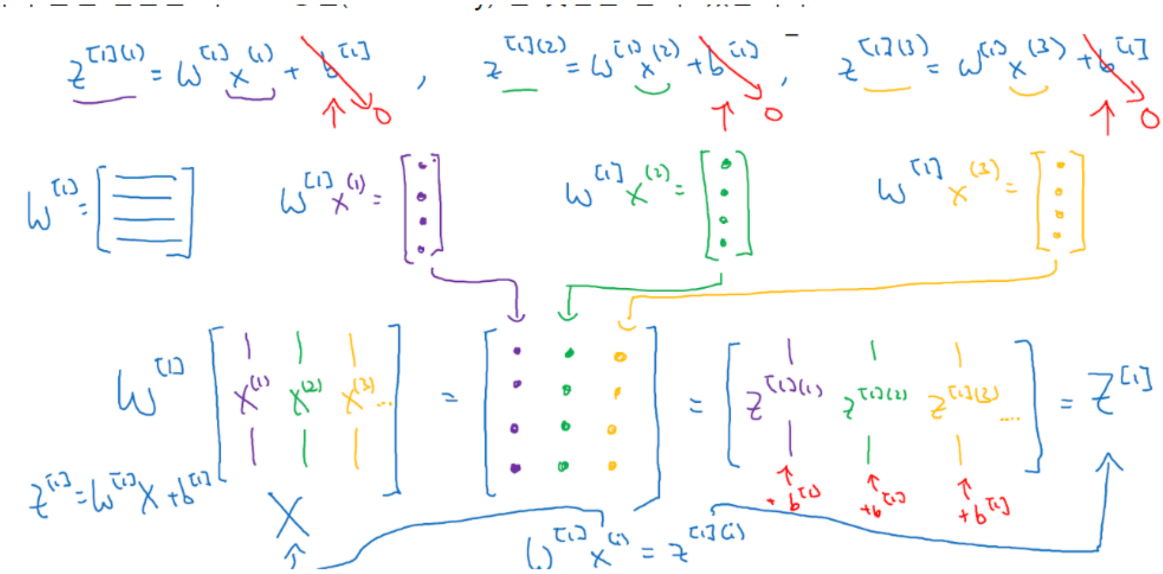
→ for문을 없애자!

- X: 훈련 샘플이 열로 쌓은 행렬 (nx*m 행렬)
  - 벡터였던 소문자 x를 열로 쌓아, 행렬인 대문자 X를 얻음
- Z^[1]: z^1부터 z^1까지 열로 쌓은 행렬
- A^[1]: a^1부터 a^1까지 열로 쌓은 행렬
  - A: 활성화 결과 행
  - 왼쪽 위에 있는 값((1,1)번째 요소): 첫 은닉 유닛의 첫 훈련 샘플의 활성값
  - 그 아래 값((2,1)번째 요소): 첫 훈련 샘플의 두번째 은닉 유닛의 활성값
  - 세로: 은닉 유닛의 번호
  - 가로: 훈련 샘플의 번호

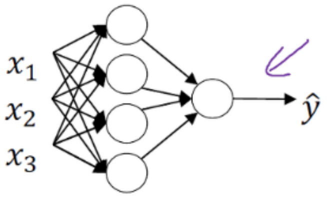
5. 벡터화 구현에 대한 설명

- 정방향 전파 계산

- 첫 훈련 샘플에 대해 Z^1=W^[1]x^(1) + b^[1] 계산
- 두 번째 훈련 샘플에 대해서 Z^1=W^[1]x^(2) + b^[1] 계산
- 세 번째 훈련 샘플에 대해서 Z^1=W^[1]x^(3) + b^[1] 계산
- → b=0으로 가정
  W^[1] * x^[1]: 열 벡터
  W^[1] * x^[2]: 열 벡터
  W^[1] * x^[3]: 열 벡터
- 행렬 X: x^(1), x^(2), x^(3)을 모두 가로로 쌓아 만든 것
- W^[1] * X → z^1, z^1, z^1이 열 벡터로 표기된 것



- b가 0이 아니라면?
  - 파이썬 broadcasting으로, 동일하게 열(column) 벡터 항목이 더해짐.



$$X = \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix}$$

$$A^{[1]} = \begin{bmatrix} | & | & & | \\ a^{[1](1)} & a^{[1](2)} & \cdots & a^{[1](m)} \\ | & | & & | \end{bmatrix}$$

```
for i = 1 to m
```

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

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

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

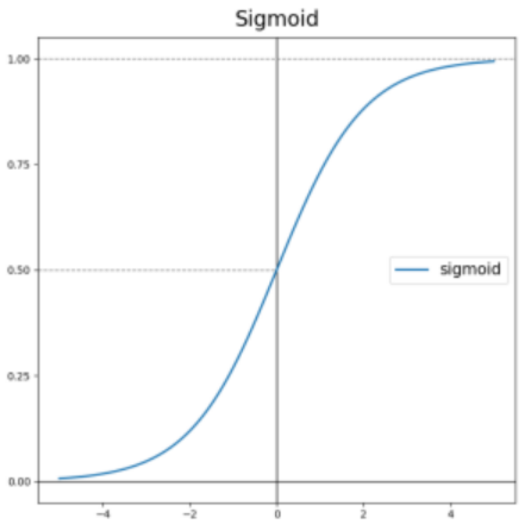
- z[1] = w[1]A[0] + b[1] 에서,
  - A[0]: 입력 특성 벡터 x로서 입력층을 의미
- 입력값을 열로 쌓으면 -> 결과도 열로 쌓인 값
- 심층 신경망(multi hidden layer): 2층 신경망과 동일한 기능과 동일한 형태를 여러 번 반복하는 것을 의미

6. 활성화 함수

- 은닉층과 출력층에서 어떤 활성화 함수를 쓸 것인가?
  → 시그모이드 함수보다 더 좋은 함수가 있다면..?
  - ex) g: 시그모이드 함수가 아닌 비선형 함수
    - **쌍곡 탄젠트 함수(tanh 함수)**: -1 ~ +1
      은닉 유닛에 대해, g(z^[1])을 tanh(z)로 두면, 시그모이드 함수보다 good
      → **평균 값이 0에 더 가깝 = 데이터를 원점으로 이동하는 효과**

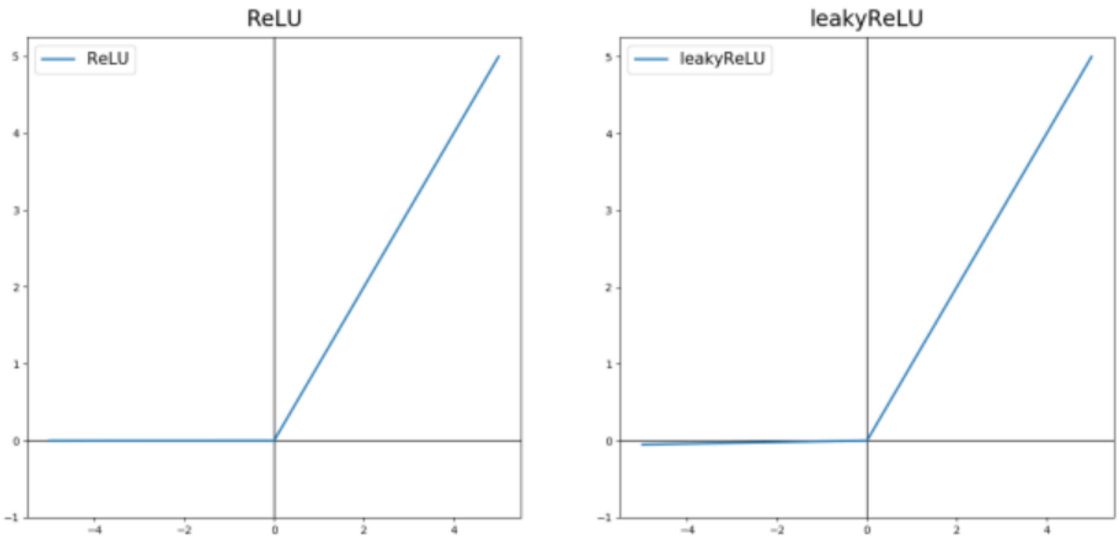
- Sigmoid
  - $a = \dfrac{1}{1+e^{-z}}$
- Tanh
  - $a = \dfrac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$

- 출력층에서 시그모이드 활성화 함수를 쓰는 예외: **이진 분류** 할 때
- **시그모이드 함수**
  - z가 크거나 작으면, 함수의 기울기가 0에 가까워짐 → 경사 하강법 느려짐



- **ReLU 함수**
  - 머신러닝에서 인기
  - a = max(0,z)
  - z가 양수: 도함수=1, z가 음수: 도함수=0
    → 활성화 함수의 기본값으로 많이 사용
  - 은닉층에 어떤 함수를 써야 할지 모르겠으면, ReLU 쓰기
  - 신경망을 훨씬 더 빠르게 학습시킬 수 있음. 은닉 유닛의 z는 0보다 큰 경우가 많기 때문
  - 단점: z가 음수일 때, 도함수가 0이다

- **leaky ReLU**: z가 음수일 때, 약간의 기울기를 준 함수(많이 쓰이지는 X)
    - • *a*=*max*(0.01*z*,*z*)



7. 왜 비선형 활성 함수를 써야 할까?

- 정방향 전파 계산
    - a^[1] = g^[1]에서 g를 없애고 z^[1]로 대체
      → g(z) = z : **선형 활성화 함수(항등함수)**
    - a^[1] = z^[1] = W^[1]*x+b^[1]
    - a^[2] = z^[2] = W^[2]*a^[1]+b^[2]
      =(W^[2]*W^[1])*x + (W^[2]*b^[1]+b^[2])
      = W'x + b'
- 선형 활성화 함수 or 항등 활성화 함수 사용 시, 신경망은 **입력의 선형식 만을 출력**
  = **은닉층이 없는 것**
- 두 선형 함수의 조합 = 하나의 선형 함수
- g(z) = z: 회귀 문제에 대한 머신러닝 할 때 이용(**y가 실수인 경우**)
- 은닉 유닛: **비선형 함수 사용(ReLU, tanh, leaky ReLU,..)**

8. 활성화 함수의 미분

1) 시그모이드 활성화 함수 g(z)

- 기울기

- d/dz g(z) = slope of g(x) at z = g(z)(1-g(z)) = g'(z): 입력변수 z에 대한 g의 도함수

- 시그모이드
  - $g(z) = \frac{1}{1+e^{-z}}$
  - $g'(z) = \frac{d}{dz}g(z) = g(z)(1-g(z))$

2) tanh 활성화 함수

- 기울기
  - d/dz g(z) = slopr of g(z) at z = g'(z) = 1 - (tanh(z))^2
  - a = g(z) = tanh(z)일 때, g'(z)=1-a^2

- Tanh
  - $g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$
  - $g'(z) = 1 - (g(z))^2$

3) ReLU 함수

- 기울기
  - g'(z) = 0(if z<0), 1(if z>= 0)
  - g'은 활성화 함수 g(z)의 서브 경사이므로, 경사 하강법 잘 작동

→ z=0일 때의 도함수를 0으로 하든, 1로 하든 상관 없음.

- ReLU
  - $g(z) = max(0, z)$
  - $g'(z) = 0$ (z < 0인 경우)
  - $g'(z) = 1$ (z >= 0인 경우)

4) Leaky ReLU 함수

- 기울기
  - g'(z) = 0.01(z<0), 1(z>=0)
  - ReLU 함수와 마찬가지로, z=0일 때의 도함수를 0.01로 해도, 1로 해도 상관 없음.

- Leaky ReLU
  - $g(z) = max(0.01z, z)$
  - $g'(z) = 0.01$ (z < 0인 경우)
  - $g'(z) = 1$ (z >= 0인 경우)

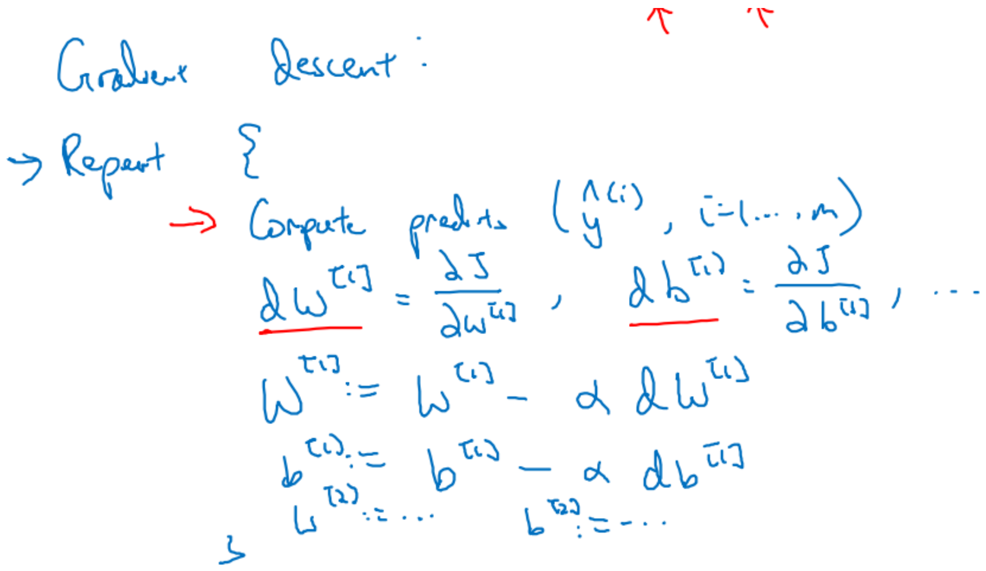
9. 신경망 네트워크와 경사 하강법

- **단일층 신경망**
  - 파라미터: **w^[1], b^[1], w^[2], b^[2]**
  - **n_x**
    - n^[0]: 입력특성
    - n^[1]: 은닉 유닛
    - n^[2]: 출력유닛
  - 비용 함수 - 이진 분류를 하고 있다고 가정한 경우
    - J(w^[1], b^[1], w^[2], b^[2]) = 손실함수의 평균

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

    - L: 신경망이 예측한 yhat값에 대한 손실
  - 변수들 훈련시키기 위해 **경사 하강법** 사용

- 0이 아닌 값으로 변수를 초기화하는 것이 중요
- 초기화 후, 예측값 계산(=yhat^[i] 계산, i=1,..,m)
- **도함수 계산**
    - dw^[1] = dJ/dw^[1] : 변수 w^[i]에 대한 비용 함수의 도함수
    - db^[1]: 변수 b^[1]에 대한 비용 함수의 도함수
    - W^[1] = W^[1] - alpha*dW^[1]*
    - *b^[1] = b^[1] - alpha*db^[1]
      (alpha: 학습률 의미)



    - 이 과정을 변수가 수렴할 때 까지 반복!

- 역전파 단계 도함수 계산
    - dz^[2]계산 : A^[2]-Y, Y: (1,m) 행렬(m가지의 모든 샘플을 가로로 나열)
    - dW^[2] = 1/m*dz^[2]*A^[1].T
    - db^[2] = 1/m * **np.sum(dz^[2], axis=1, keepdims=True)** ← (n^[2], 1) 벡터
    - n^[2]가 1일 때, keepdims의 영향 없음

10. 역전파에 대한 이해

〈역전파에 대한 이해〉

① Logistic regression

$$\left.\begin{matrix} x \\ w \\ b \end{matrix}\right\} \rightarrow z = w^T x + b \rightarrow a = \sigma(z) \rightarrow L(a, y)$$

→ 정방향

← 역방향 (da 계산 → dz 계산 → dw, db 계산)

$L(a, y) = -y\log a - (1-y)\log(1-a)$

i) $da = \frac{d}{da} L(a, y) = -\frac{y}{a} + \frac{1-y}{1-a}$

ii) $dz = da \cdot g'(z)$ → $g(z)$: 시그모이드 함수

$\frac{dL}{dz} = \frac{dL}{da} \cdot \boxed{\frac{da}{dz}}$, $\frac{da}{dz} = \frac{d}{dz} g(z) = g'(z)$

$dz = da \cdot g'(z)$

$= a - y$

iii) $dw = dz \cdot x$

$db = dz$

②

$$\left.\begin{matrix} x \\ W^{[1]} \\ b^{[1]} \end{matrix}\right\} \rightarrow z^{[1]} = W^{[1]}x + b^{[1]} \rightarrow a^{[1]} = \sigma(z^{[1]}) \rightarrow z^{[2]} = W^{[2]}a^{[1]} + b^{[2]} \rightarrow a^{[2]} = \sigma(z^{[2]}) \rightarrow L(a^{[2]}, y)$$

($W^{[2]}$, $b^{[2]}$ → $z^{[2]}$)

$da^{[1]}$

i) $dz^{[2]} = a^{[2]} - y$

$dW^{[2]} = dz^{[2]} a^{[1]T}$ → $dw = dz \cdot x$

$db^{[2]} = dz^{[2]}$

* 하나의 출력을 가지는 경우, W: 행벡터

ii) $dz^{[1]} = W^{[2]T} dz^{[2]} \cdot g^{[1]\prime}(z^{[1]})$

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

iii) $dW^{[1]} = dz^{[1]} \cdot x^T$ ← x : $a^{[0]}$의 역할

$db^{[1]} = dz^{[1]}$

③ 도출된 역전파 식

1) $dz^{[2]} = a^{[2]} - y$ → $dZ^{[2]} = A^{[2]} - Y$

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

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

4) $dz^{[1]} = W^{[2]T} dz^{[2]} * g^{[1]\prime}(z^{[1]})$ → $dZ^{[1]} = W^{[2]T} dZ^{[2]} * g^{[1]\prime}(Z^{[1]})$ ('요소별 곱셈' 의미)

5) $dW^{[1]} = dz^{[1]} x^T$ → $\frac{1}{m} dZ^{[1]} X^T$

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

④ 벡터화

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

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

→ $\begin{bmatrix} | & | & & | \\ z^{[1](1)} & z^{[1](2)} & \cdots & z^{[1](m)} \\ | & | & & | \end{bmatrix} = Z^{[1]}$

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

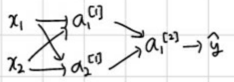
11. 랜덤 초기화

〈랜덤 초기화〉

① 신경망 훈련 시, 변수 초기화

로지스틱 회귀 : 모두 0으로 초기화 ok
⇓
경사하강법 적용하면, 동작 X

왜?



입력특성 : 2개 은닉층 : 2개
$n^{[0]}=2$ $n^{[1]}=2$

$W^{[1]} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$, $b^{[1]} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ 가정

$\Rightarrow a_1^{[1]} = a_2^{[1]}$ & $dz_1^{[1]} = dz_2^{[1]}$ (가중치가 동일)

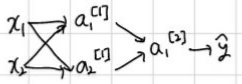
$\Rightarrow W^{[2]} = [0 \ 0]$

: Symmetric (대칭적) = 두 은닉유닛이 항상 같은 함수를 계산
= 〃 출력유닛에 같은 영향
= 은닉유닛이 1개

$dw = \begin{bmatrix} u & v \\ u & v \end{bmatrix}$

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

② 다른 함수 계산 위해서, 각각 다른 유닛 필요



$W^{[1]} = \underbrace{\text{np.random.randn}}_{\text{가우시안 랜덤 변수 생성}}((2,2)) * 0.01$ → 작은 임의의 수로 초기화

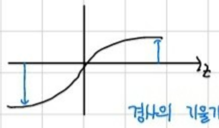
$b^{[1]} = \text{np.zeros}((2,1))$ → b는 0으로 초기화해도 ok

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

$b^{[2]} = 0$

가중치의 초기값 : 매우 작은값인 것이 좋음.

가중치가 너무 크면,
$z^{[1]} = W^{[1]}x + b^{[1]}$
$a^{[1]} = g^{[1]}(z^{[1]})$



경사의 기울기가 낮음.
= 경사하강법 느리게 적용
= 학습속도 ↓