AI and Deep Learning

Deep Learning

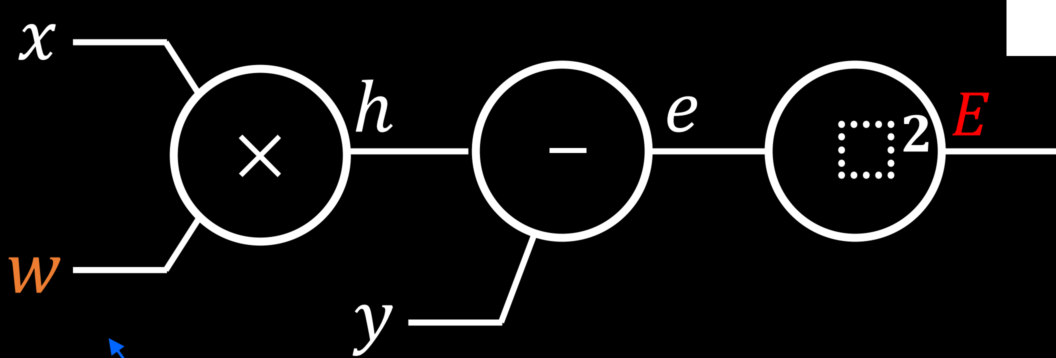
Jeju National University
Yung-Cheol Byun

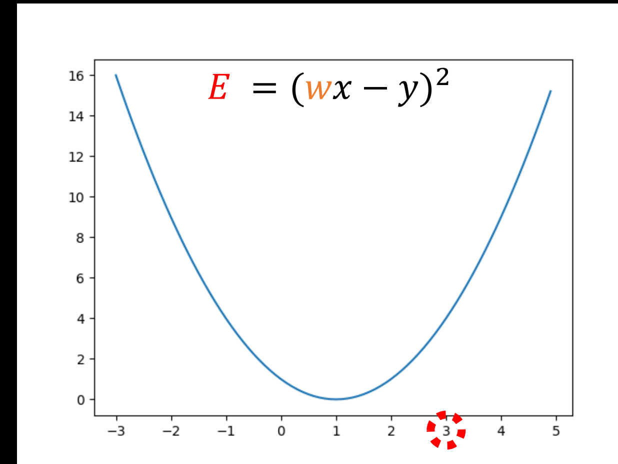
```
#----- a neuron
w = tf.Variable(tf.random_normal([1]))
hypo = w * x_data

#----- learning
cost = (hypo - y_data) ** 2   #cost = Error (E)
```

$$\boxed{E = (wx - y)^2}$$

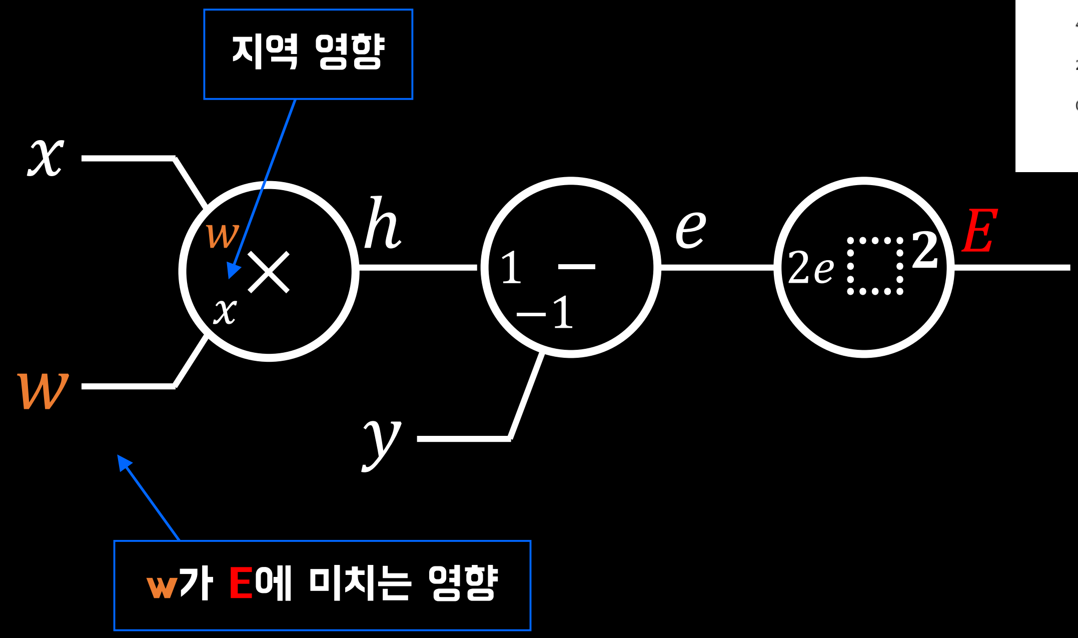
오류 계산 그래프



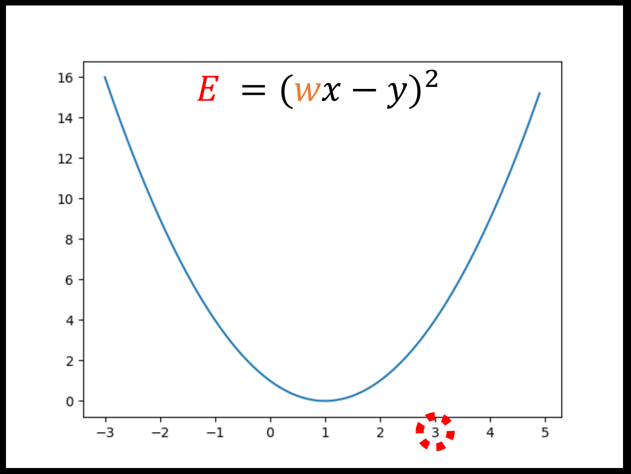


w가 E에 미치는 영향

오류 계산 그래프

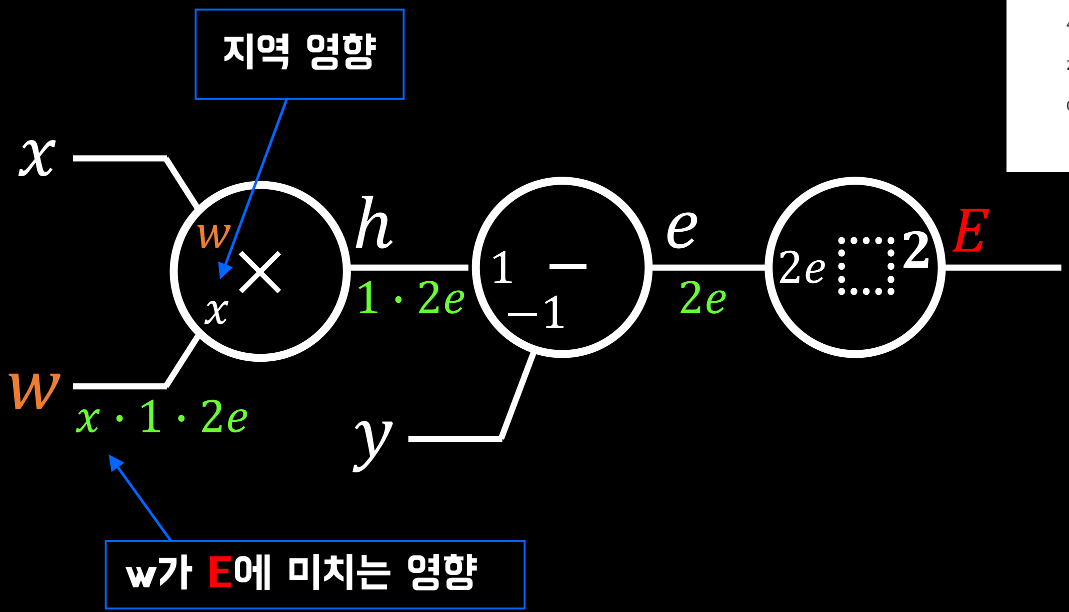


지역 영향
x
w
w
×
x
h
1 −
−1
y
e
2e
2
E
w가 E에 미치는 영향

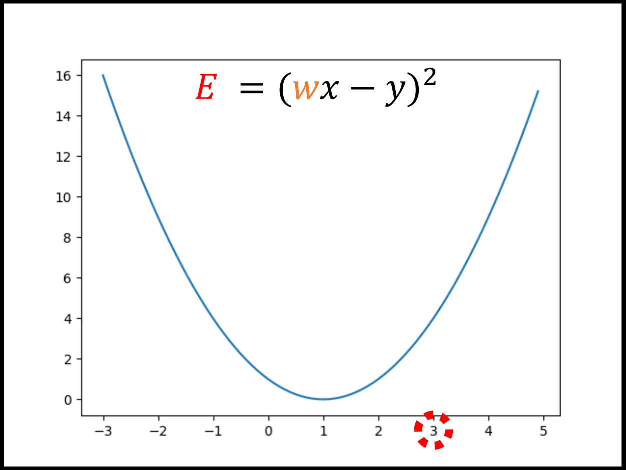


$E = (wx - y)^2$
16
14
12
10
8
6
4
2
0
−3
−2
−1
0
1
2
3
4
5

오류 계산 그래프

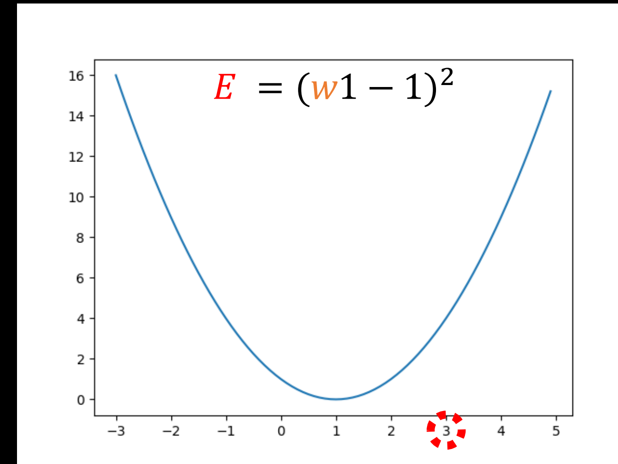


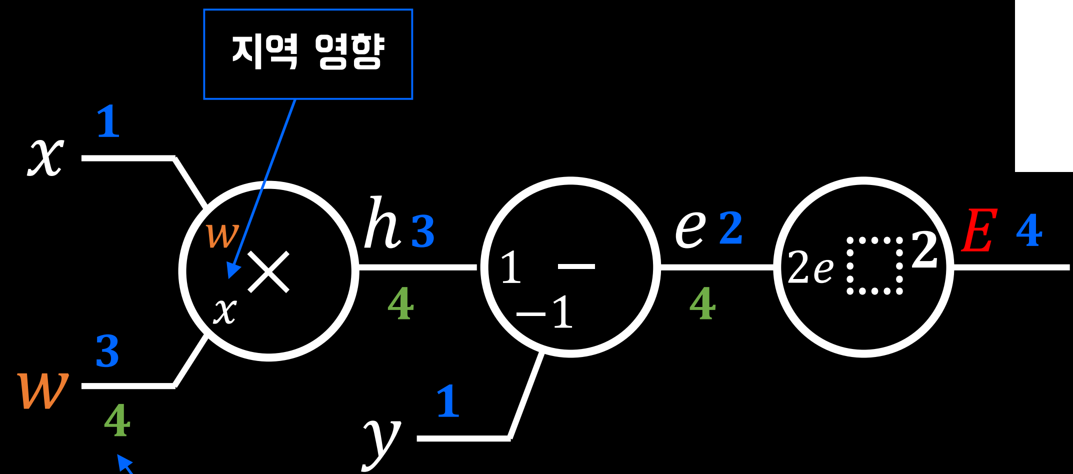
지역 영향
x
w
x
h
1 · 2e
1 −
−1
e
2e
2e
2
E
w
x · 1 · 2e
y
w가 E에 미치는 영향



$E = (wx - y)^2$
16
14
12
10
8
6
4
2
0
−3
−2
−1
0
1
2
3
4
5

오류 계산 그래프





w가 E에 미치는 영향

$$1 * 1 * 2e = 1 * 1 * 2(w1 - 1) = 2(w1 - 1)$$

기울기, 미치는 영향

$$E = (w \cdot 1 - 1)^2$$

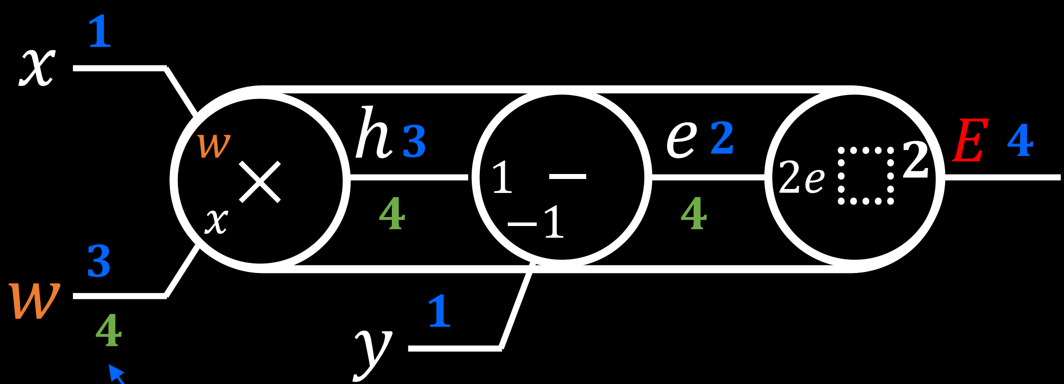
$$1 * 1 * 2e = 1 * 1 * 2(w \cdot 1 - 1) = 2(w \cdot 1 - 1)$$

$$\boxed{\frac{\partial E}{\partial w} = \frac{\partial}{\partial w} E = \frac{\partial}{\partial w} (w \cdot 1 - 1)^2 = 2(w \cdot 1 - 1)}$$

기울기, 미치는 영향

- 미치는 영향(기울기, gradient)는 어떻게 구했나?
  - 입력을 1만큼(혹은 아주 조금, 0.00001) 증가해서 출력이 얼마나 증가하는지 확인해 봄.
  - 또는 직접 미분할 수도 있음.

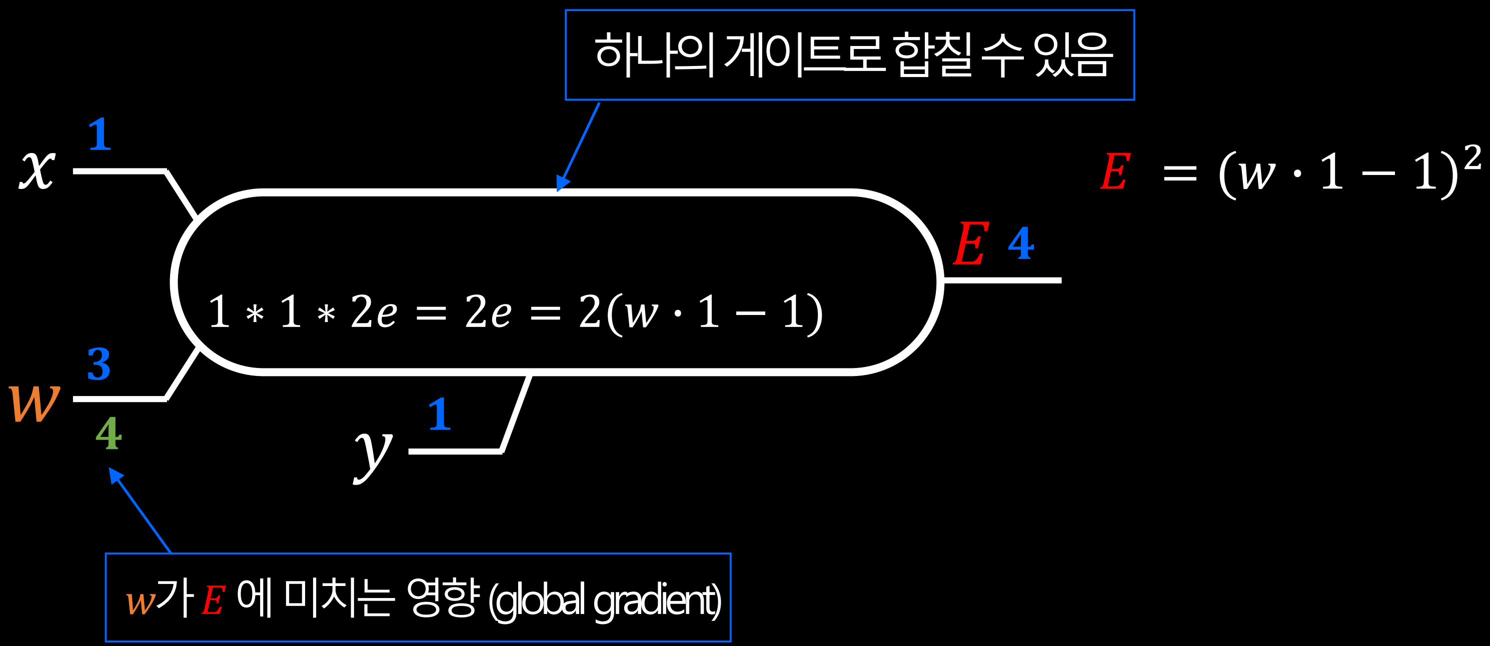
연산 합치기



$$E = (w \cdot 1 - 1)^2$$

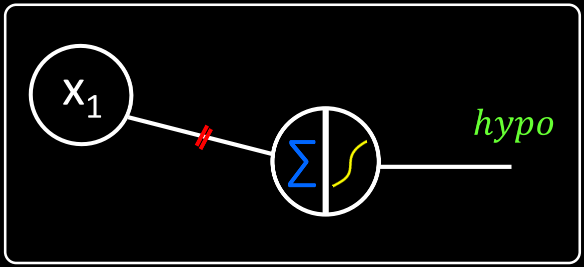
w가 E 에 미치는 영향 (global gradient)

연산 합치기



하나의 게이트로 합칠 수 있음
x 1
w 3
4
y 1
$1 * 1 * 2e = 2e = 2(w \cdot 1 - 1)$
E 4
$E = (w \cdot 1 - 1)^2$
w가 E 에 미치는 영향 (global gradient)

오류 계산 그래프



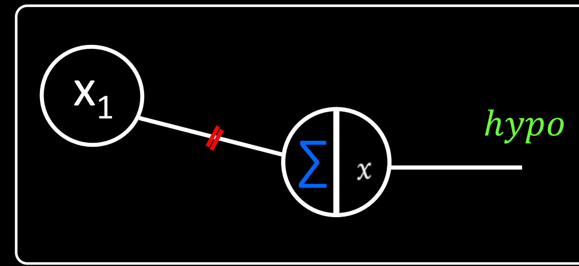
로지스틱 리그레션
신경 세포 1개 있을 때

$$cost(E) = -y\log(hypo) - (1-y)\log(1-hypo)$$

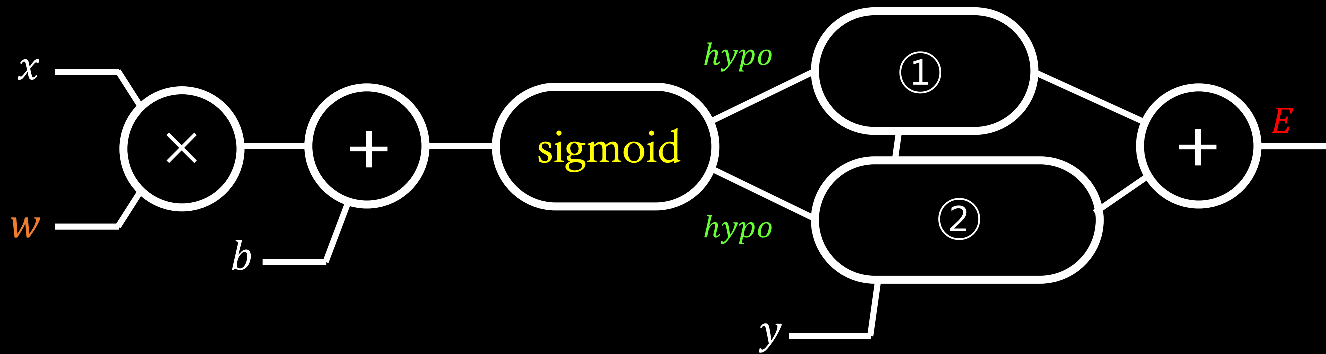
$$hypo = \frac{1}{1+e^{-wx}}$$

w가 cost에 미치는 영향은?
w를 아주 조금 바꿨을 때 cost는 어떻게 변하나?
계산 그래프를 활용하거나 또는 직접 미분하거나...

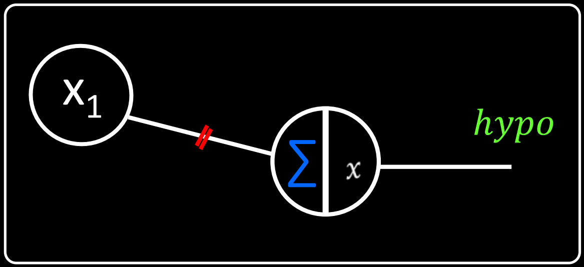
오류 계산 그래프



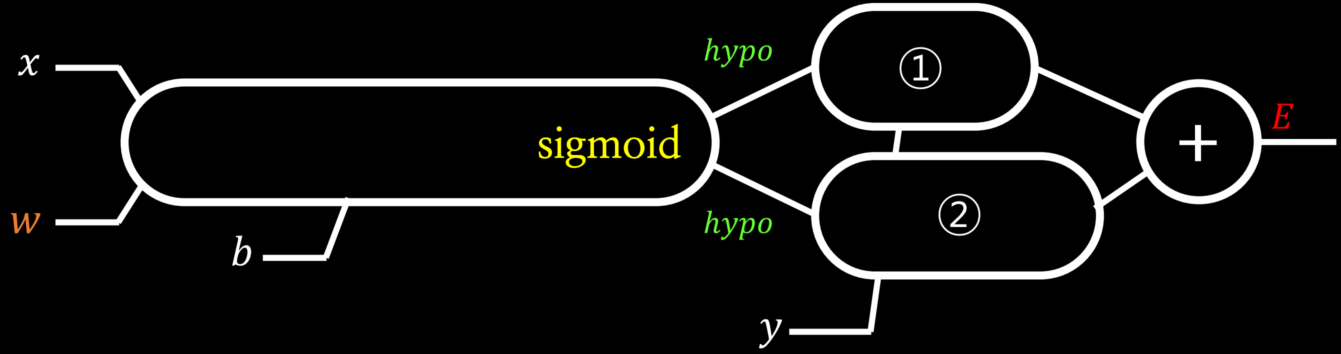
$$E = \overset{\textcircled{1}}{\boxed{-(y \cdot \log(hypo)}} + \overset{\textcircled{2}}{\boxed{(1-y) \cdot \log(1-hypo))}})$$



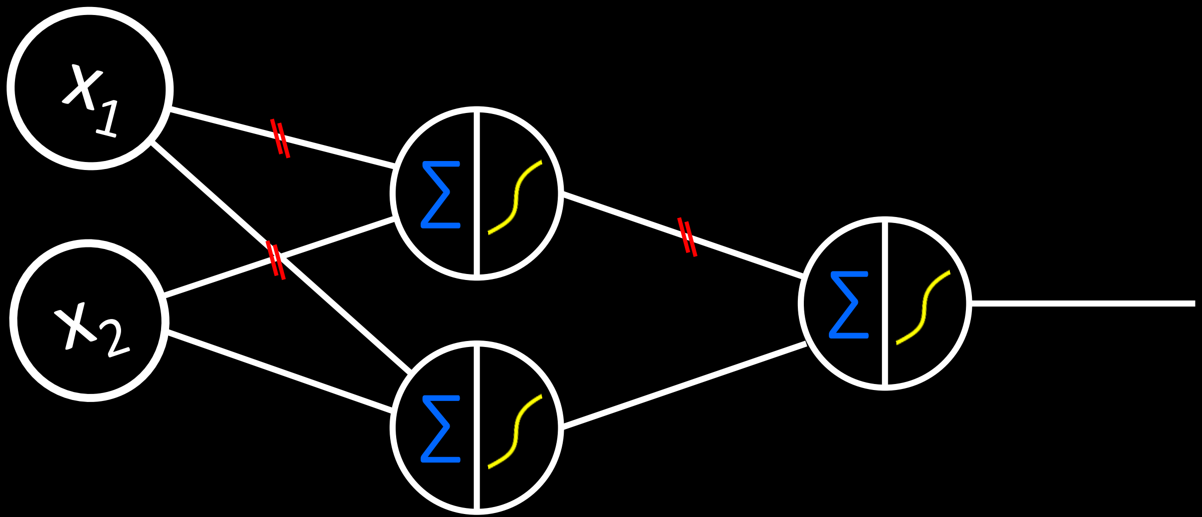
오류 계산 그래프



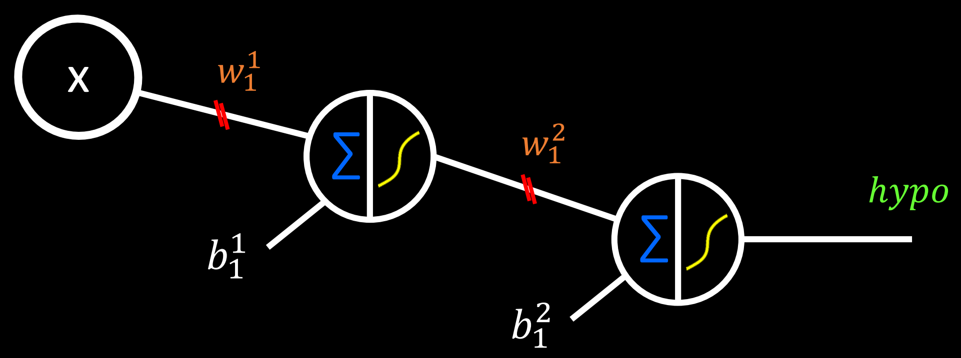
$$E = \overset{①}{\boxed{-(y \cdot \log(hypo)}} + \overset{②}{\boxed{(1-y) \cdot \log(1-hypo))}})$$



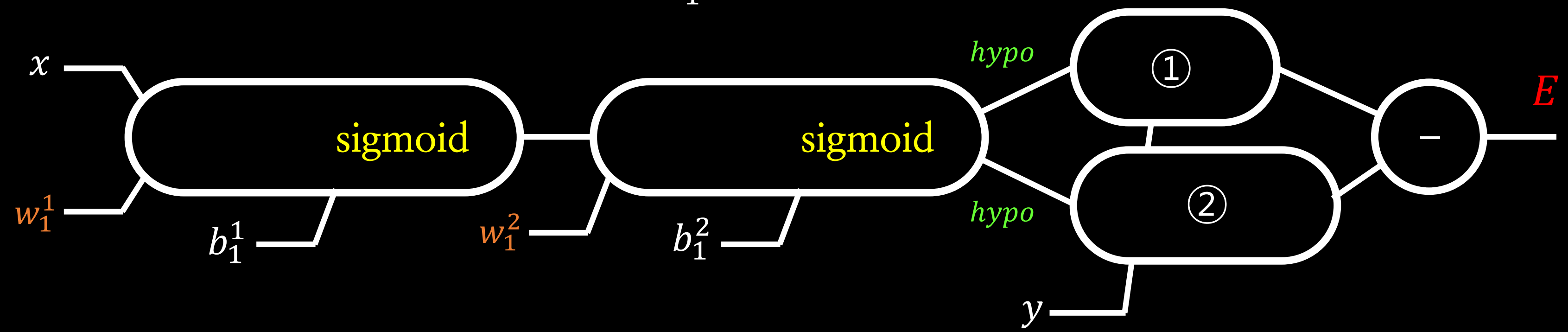
오류 계산 그래프



오류 계산 그래프



x
w_1^1
b_1^1
w_1^2
b_1^2
hypo



x
w_1^1
sigmoid
b_1^1
w_1^2
sigmoid
b_1^2
hypo
hypo
①
②
y
−
E

사라지는 영향력

- Local gradient는 뉴런의 sigmoid function을 미분한 것
- sigmoid를 미분하면 (1-sigmoid(x)) * sigmoid(x)
- sigmoid는 입력 값을 0~1 사이의 값으로 squash
- 따라서 0~1 사이의 값의 곱은?
- 100층 신경망의 경우 100개의 sigmoid 연산

사라지는 영향력

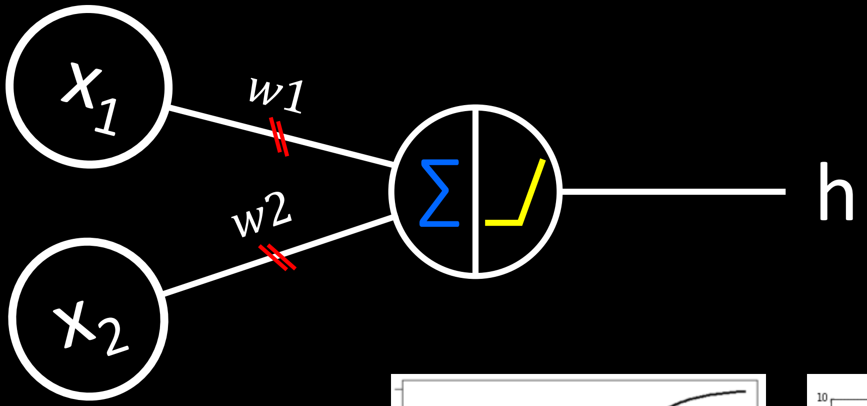
- w가 E에 미치는 양향은 수많은 1 이하의 곱
- 신경망의 왼쪽으로 갈수록 sigmoid가 점점 많아져서 1 이하의 곱이 많아져서 w가 E에 미치는 영향(기울기, gradient)은 거의 0
- 즉, 영향력이 사라짐 (Vanishing Gradient)
- $w = w - \alpha \times$ (미치는 영향~거의 0)
- 신경망의 왼쪽에 있는 뉴런들의 w, b가 거의 갱신되지 않음

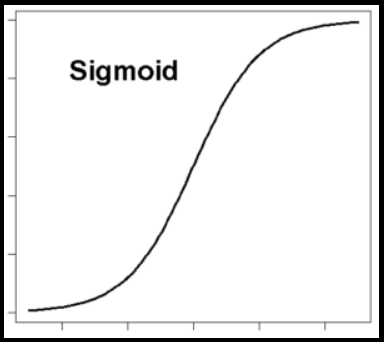
(실습) 18.py

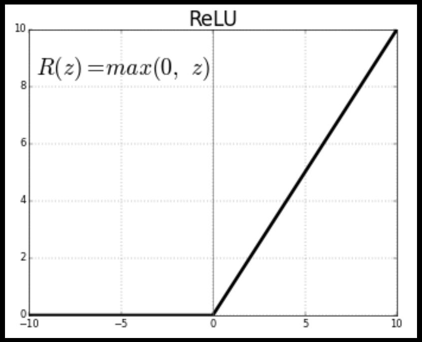
- 4층으로 구성된 신경망으로 XOR 문제를 해결하고자 했으나 Vanishing Gradient 때문에 실패

ReLU

Logistic 함수 대신 **ReLU**를 사용함으로써 Vanishing Gradient 문제 해결







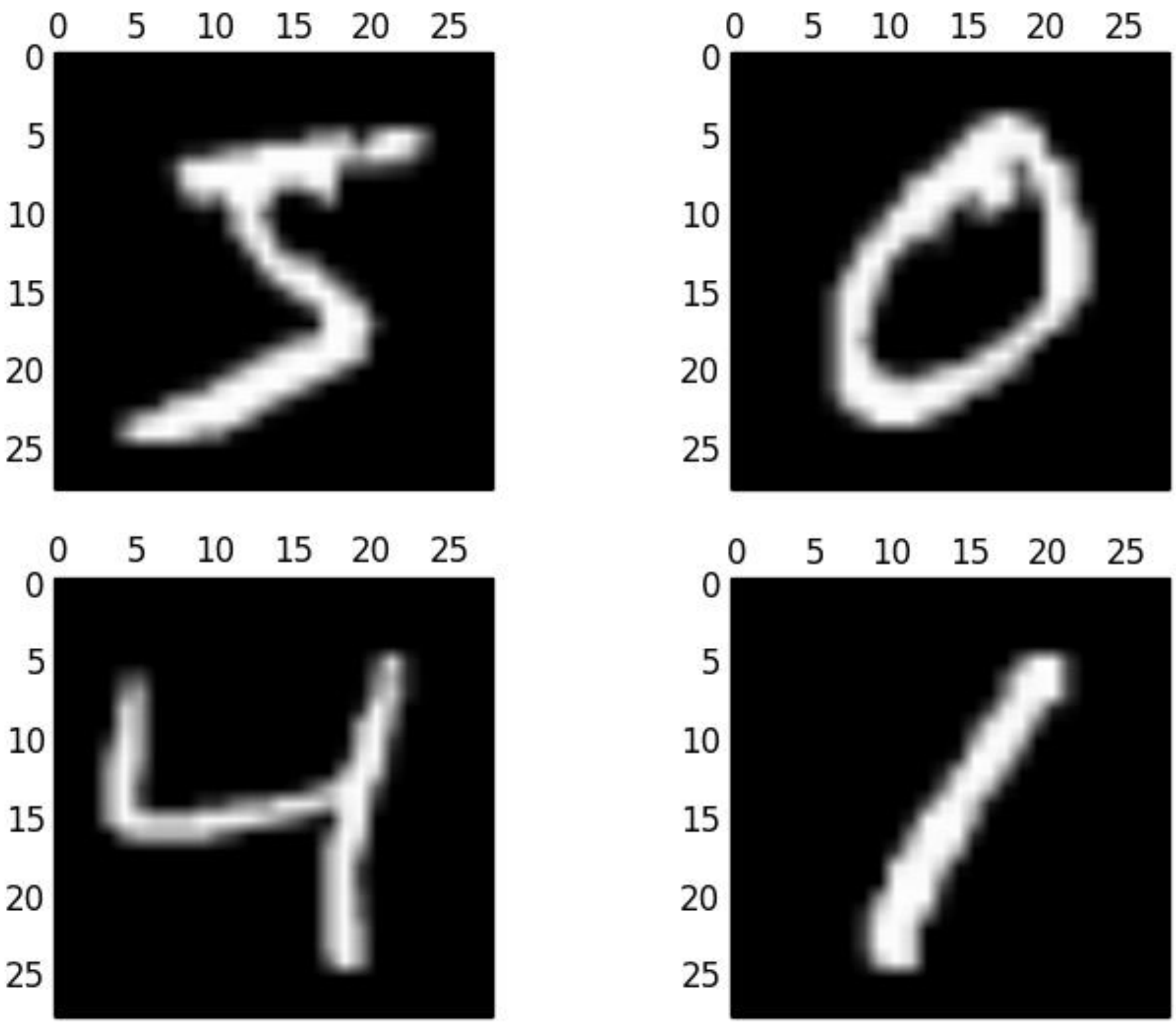
(실습) 19.py

- ReLU를 이용하여 deep 신경망에서도 역전파 학습이 잘 됨을 보임.

MNIST

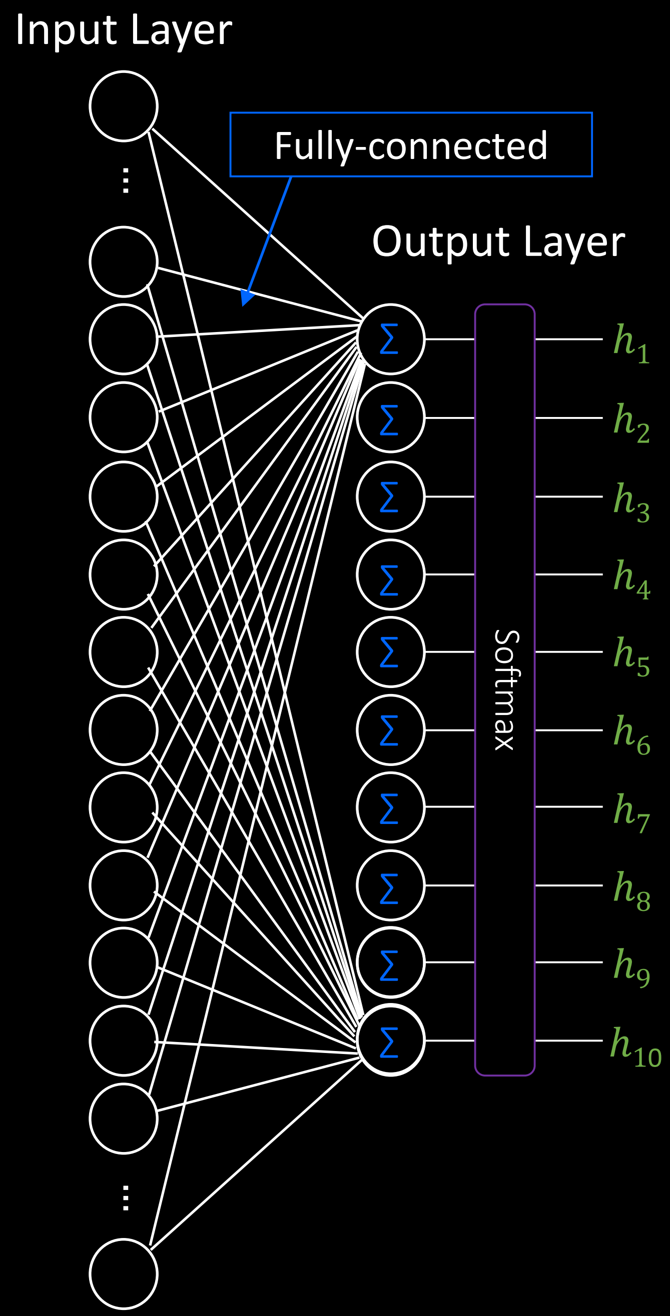
Modified National Institute of Standards and Technology (USA)

MNIST

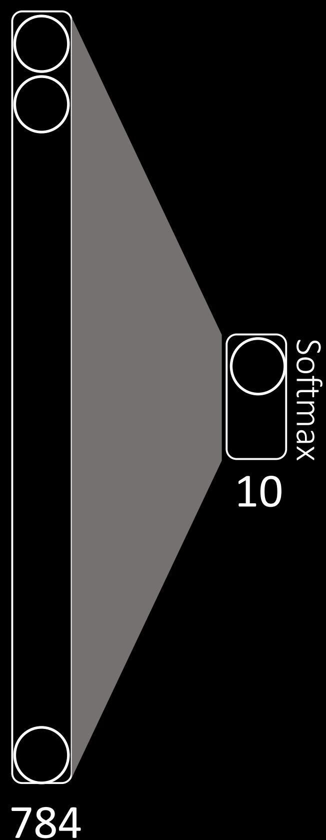

0 5 10 15 20 25
0 5 10 15 20 25
0 5 10 15 20 25
0 5 10 15 20 25
0 5 10 15 20 25
0 5 10 15 20 25
0 5 10 15 20 25
0 5 10 15 20 25

(Lab) 20.py

- 60,000 training images + 10,000 testing images
- Input image : 28 * 28 pixels → 784 pixels
- 784 dimension
- 10 classes (output: 0 ~ 9)
- Softmax
- 90.23% of recognition rate

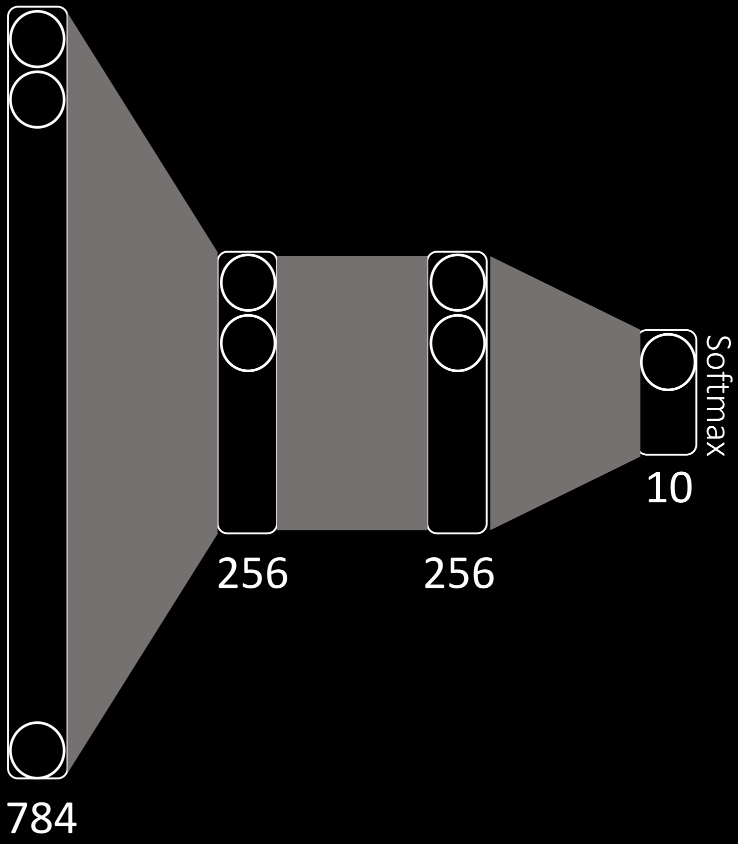


Input Layer
Fully-connected
Output Layer
Softmax
h_1
h_2
h_3
h_4
h_5
h_6
h_7
h_8
h_9
h_{10}


Softmax
10
784

(Lab) 21.py

- Deep Neural Network (4-layer)
- ReLU
- 94.55% accuracy

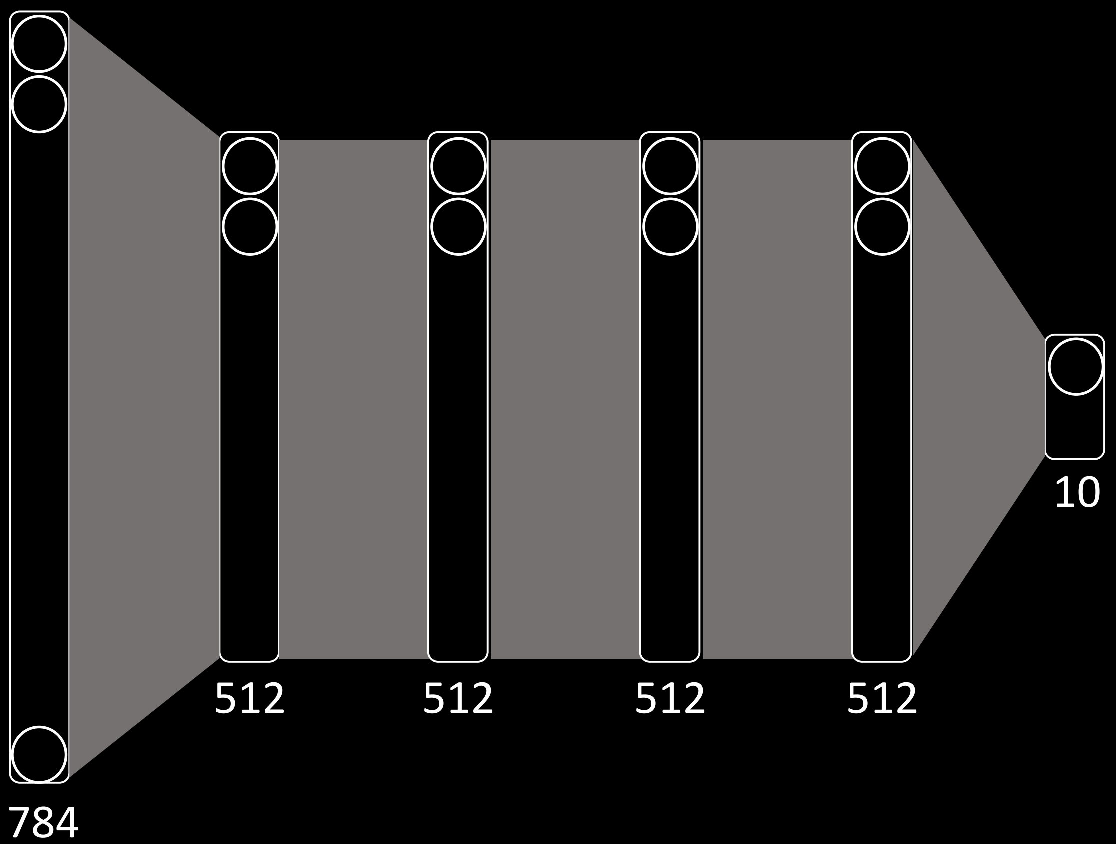


(Lab) 22.py

- 시냅스 가중치(w)와 바이어스 (b)를 적절히 초기화
- Accuracy: 97.83%

(Lab) 23.py

- 시냅스 가중치(w)와 바이어스(b)를 적절히 초기화
- Deeper (DNN) -> 6 layers
- 97.23% of accuracy


784
512
512
512
512
10

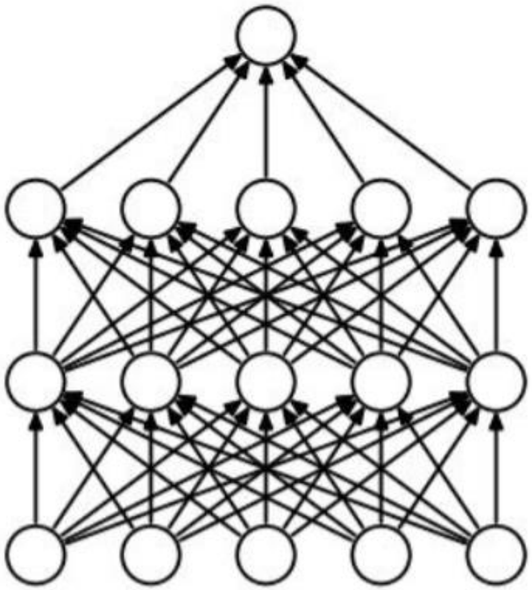
오버 피팅(over-fitting)

- 신경망의 깊이와 너비가 클 수록(deep & wide) 결정 경계는 매우 복잡
- 학습 데이터에 대해 지나치게 학습하여 기가 막히게 잘됨
- 하지만 테스트 데이터에 대해서는 에러가 많이 남 -> 오버 피팅

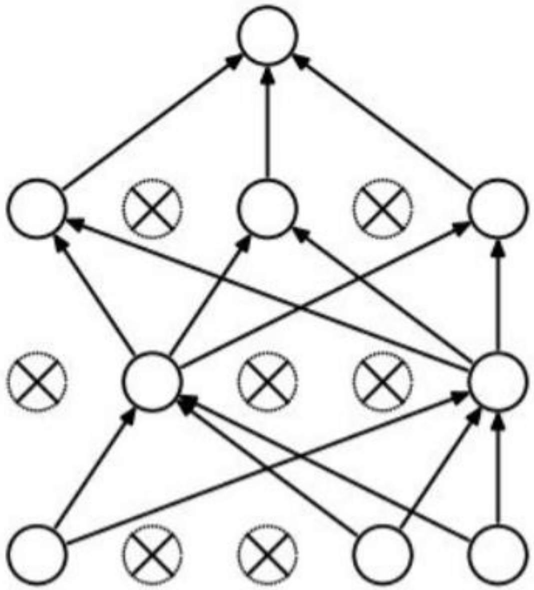
신경 세포가 많을 수록(deep & wide) 결정 경계
복잡하므로 학습 시 신경세포를 배제(drop-out)

Regularization: **Dropout**

"randomly set some neurons to zero in the forward pass"



(a) Standard Neural Net

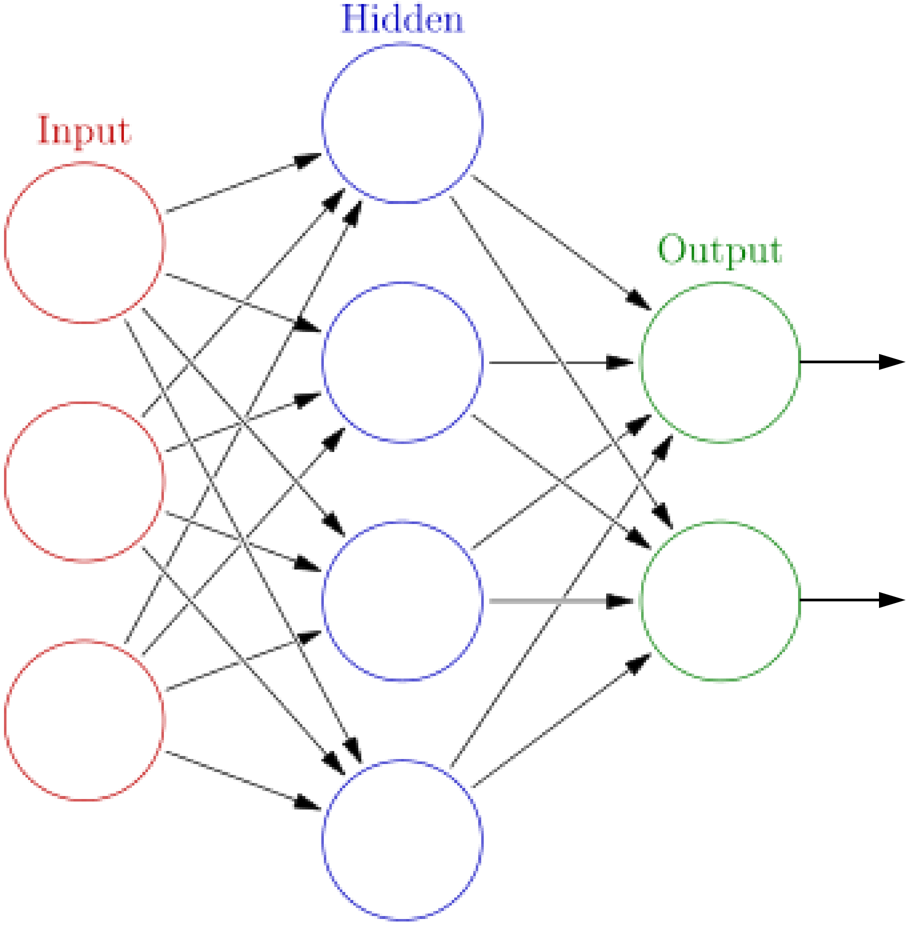


(b) After applying dropout.

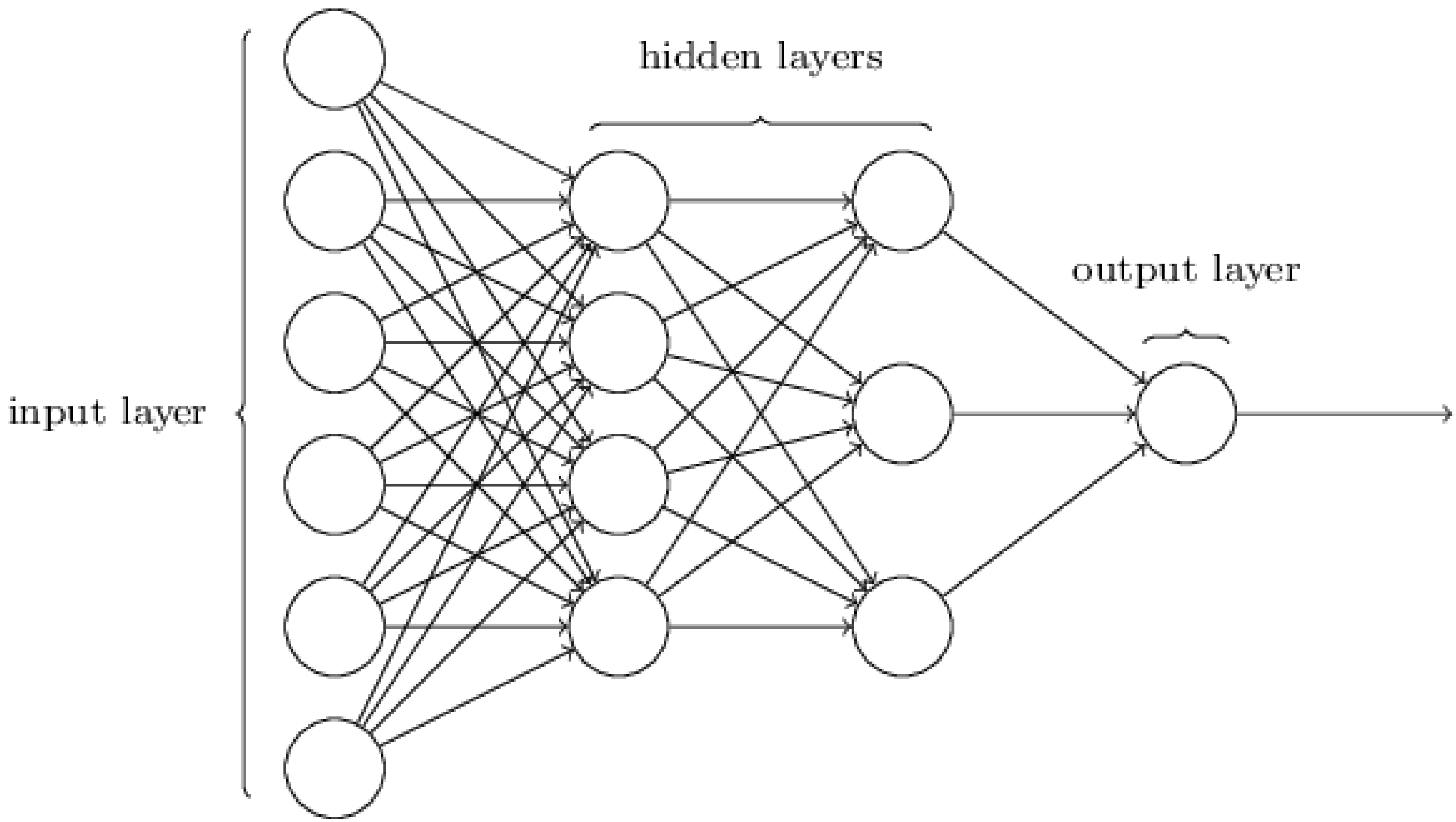
[Srivastava et al., 2014]

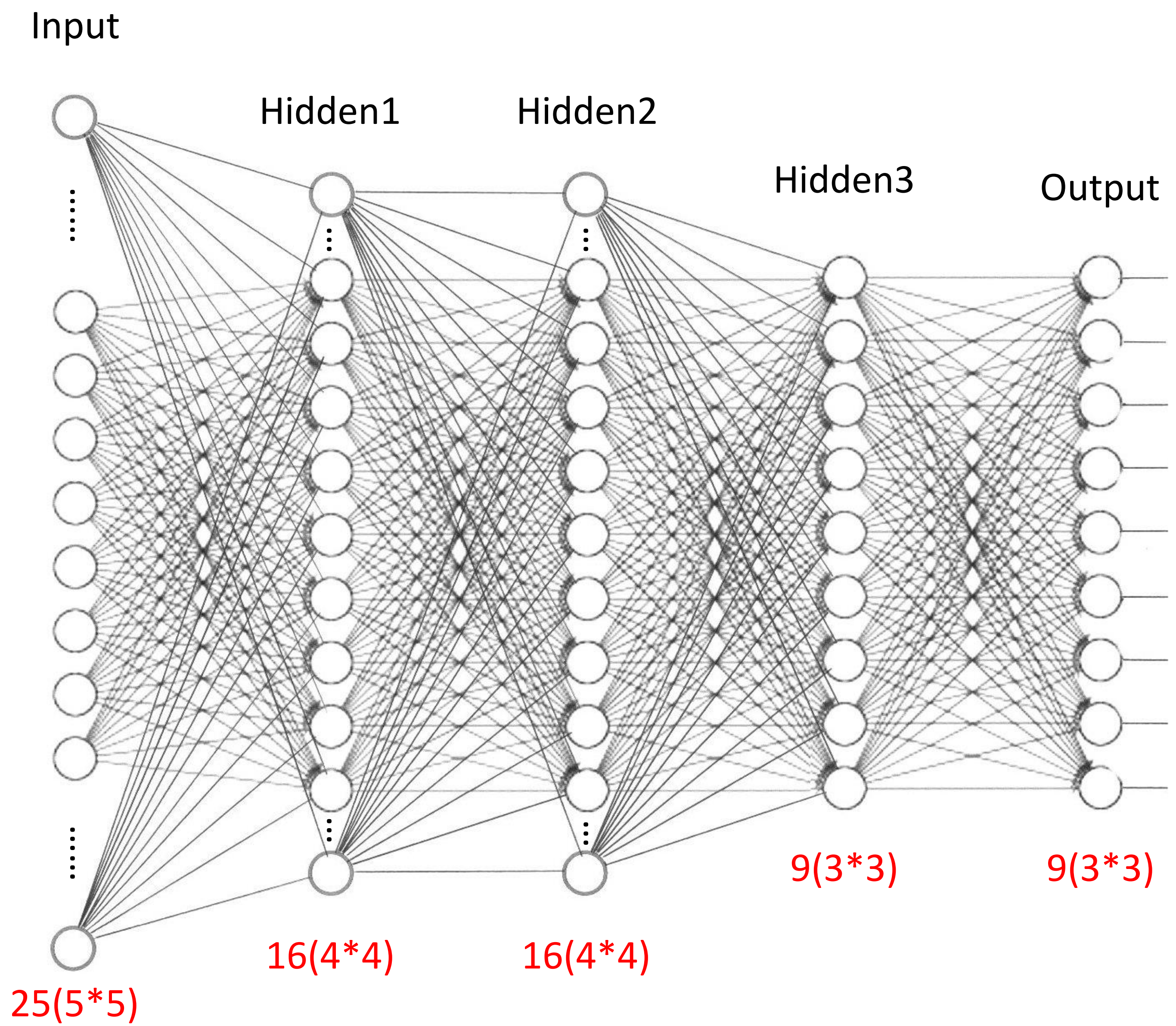
(실습) 24.py

- 시냅스 가중치(w)와 바이어스(b)를 적절히 초기화
- Deeper (DNN) -> 6개 층
- Dropout
- 98.13% of accuracy



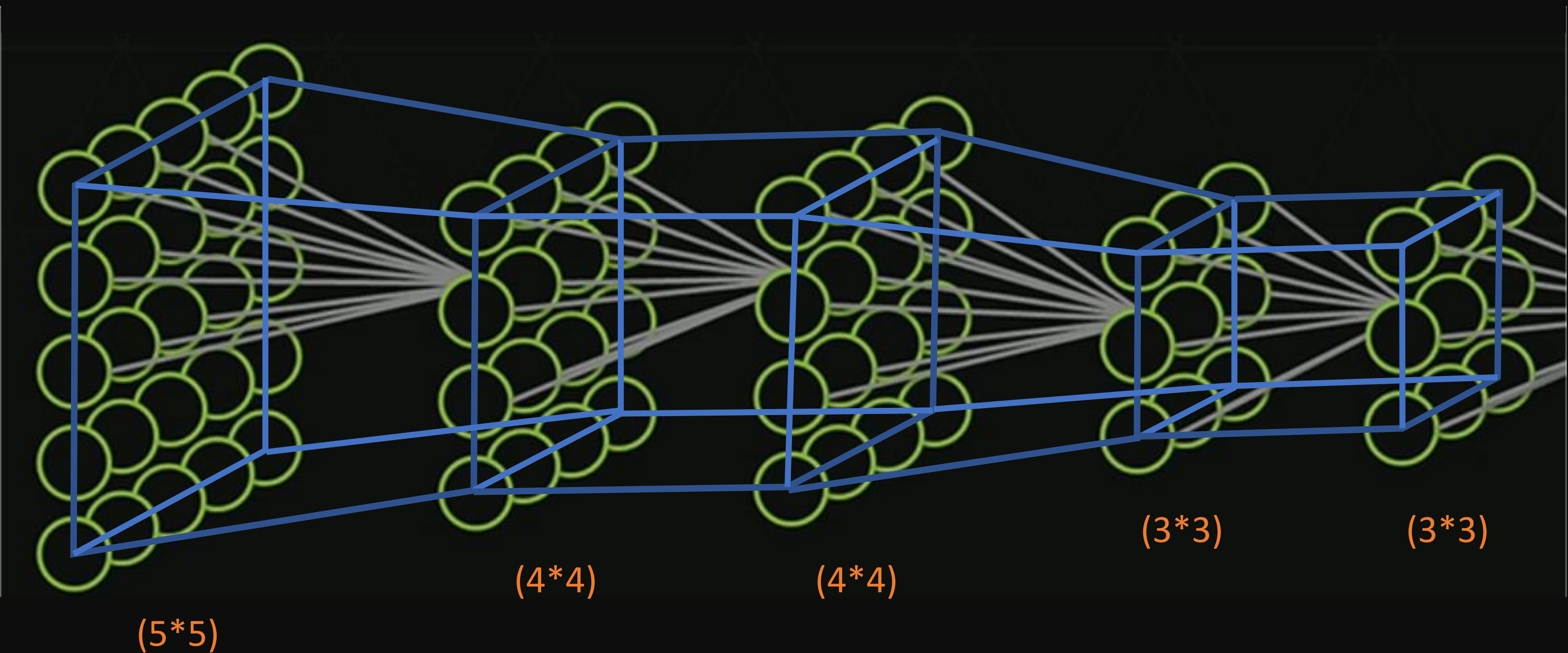
Hidden
Input
Output


hidden layers
output layer
input layer



Fully connected, so how many connections(parameters) are there?

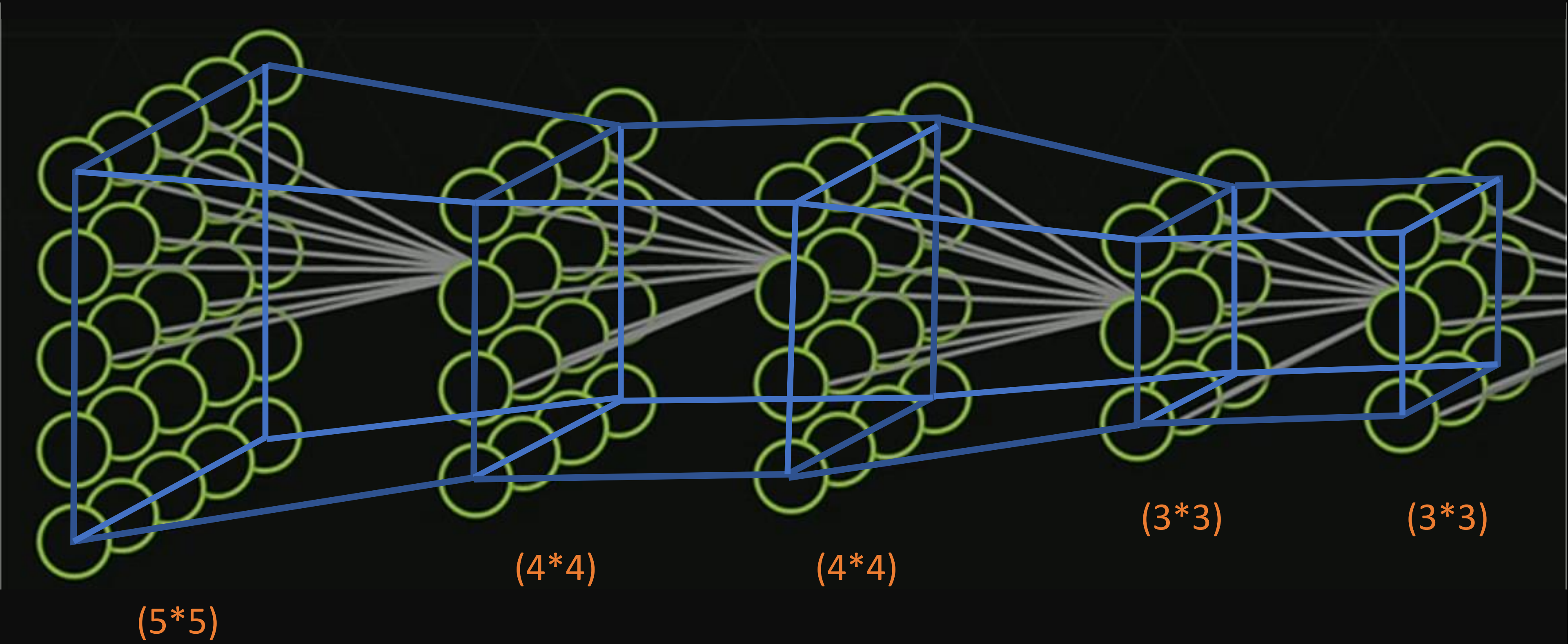
$25 * 16 + 16 * 16 + 16 * 9 + 9 * 9 = 881$



(5*5)
(4*4)
(4*4)
(3*3)
(3*3)

Fully connected, so how many connections are there?

$25 * 16 + 16 * 16 + 16 * 9 + 9 * 9 = 881$



Geoffrey Hinton, Yann LeCun, Yoshua Bengio, Andrew Ng

Google Baidu 百度 facebook IBM

Deep Learning

- in early 2000s (2006, 2010, 2012)
- Deep Neural Networks
- Weight initialization methods
- Activation functions (ReLU)
- Dropout (2014)
- Big data
- GPU