DNN

XOR 연산 과정 시각화

202192020 이연승

목차

1. 무엇이 중요한가?
2. DNN의 서사
3. 선형변환과 그 한계
4. XOR
5. 비선형변환
6. 예제로 이해하기 - 순전파, Loss, 역전파, 파라미터 업데이트, 순전파
7. 코드로 이해
8. QnA

이 발표를 듣고있는 여러분들에게...

당신은 AI를 공부하면서

어떤 것이 가장 중요하다고

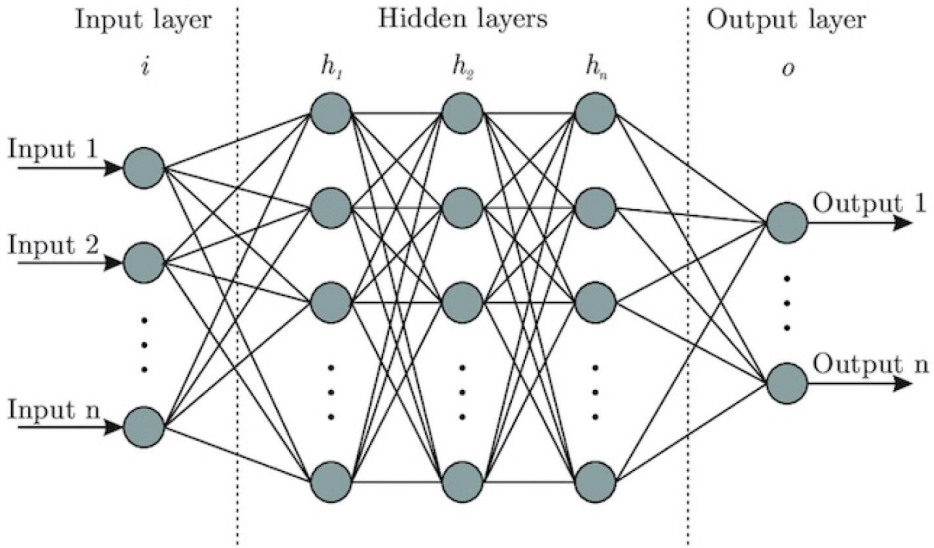
생각하십니까?

쏟아지고 있는 여러 모델들, 근본적인 개념

오늘날까지 발표된 논문들이나 모델들 - 대부분 여러 기법들을 혼합

근본적으로 빠지지 않는 한가지 개념

DNN



이번 시간에는 강력하고 중요한 **DNN**을 자세하게 살펴봅시다

DNN의 서사 - DNN은 '신'이다.

1950-60년대
로젠블롯의 퍼셉트론 - 머신이 스스로 학습할 수 있다!!
마빈 민스키 - XOR 문제 못푼다

1980~90년대
럼멜 하트, 힌튼 등-Backpropagation(역전파) 제시
층을 조금만 깊게 쌓으면 학습이 안 되는 **Vanishing Gradient**

DNN의 서사 - DNN은 '신'이다.

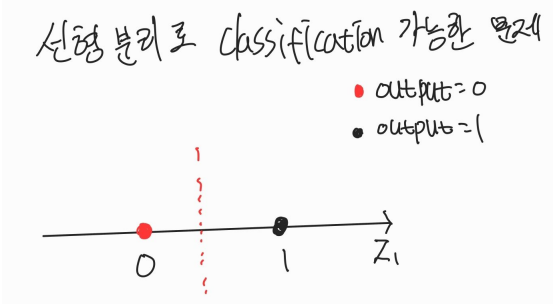
2006년 캐나다의 삼총사 등장 - 힌튼, 얀 르쿤, 요슈아 벤지오
Science 지에 논문 발표-Reducing the Dimensionality of Data with Neural Networks

신경망을 사용한 데이터의 차원 축소

2009~2012년
Hinton·Deng 팀 음성 인식대회에서 DNN이 기존 GMM 모델을 크게 이
이미지 쪽에서는 AlexNet 같은 딥 CNN으로 ImageNet 대회를 박살

이후로 DNN이 널리 알려지며 관련 연구, 논문들이 진행되기 시작

선형변환과 그 한계

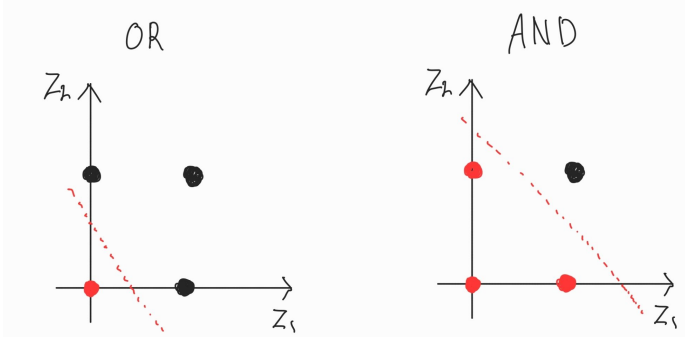


기존의 단순한 퍼셉트론 - 선형변환 y=Wx+b

and 게이트

or 게이트

nand 게이트



선으로 해당 게이트를 구별하여 나타낼 수 있음

선형변환을 아무리 많이, 복잡하게 수행해도 결국 직선임.

선형변환, 선형결합으로 풀 수 없는 문제는 어떤 방식으로 풀어야 할까?

XOR문제

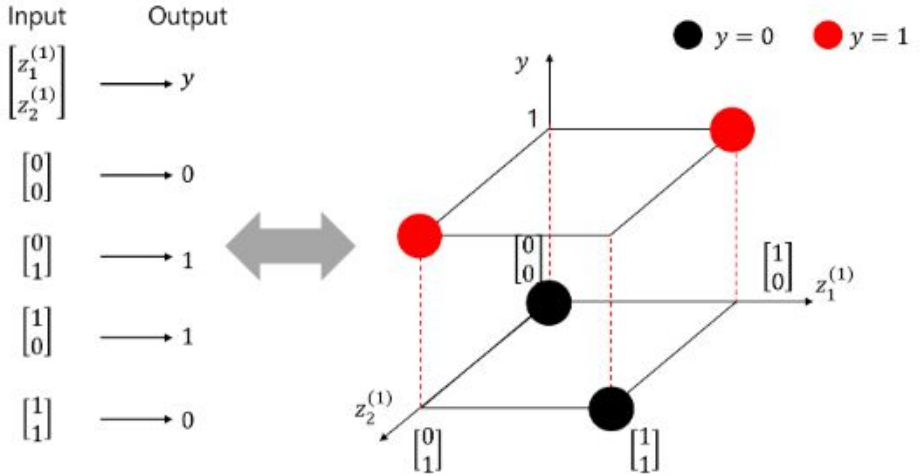
xor문제를 선형결합, 선형변환으로 풀 수 있을까?

(0,0) = 0

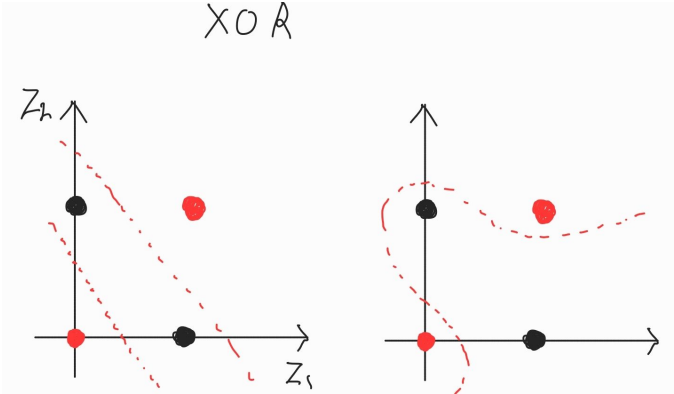
(0,1) = 1

(1,0) = 1

(1,1) = 0



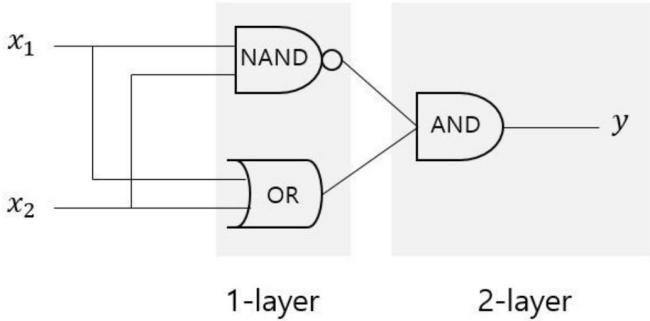
직선 하나로는 어떻게 해도 이 네 점을 구분할 수 없음



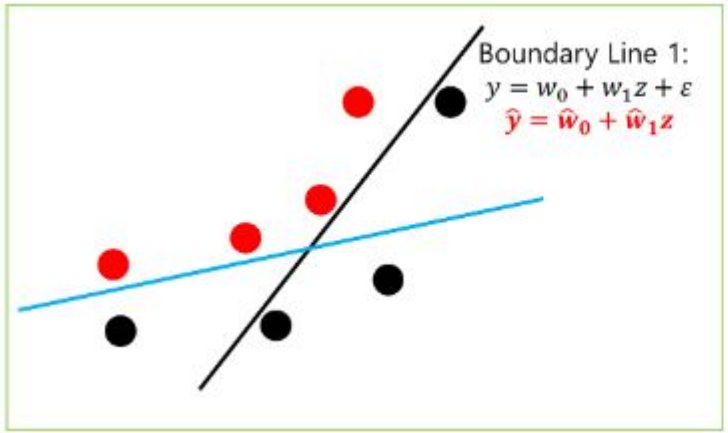
비선형변환

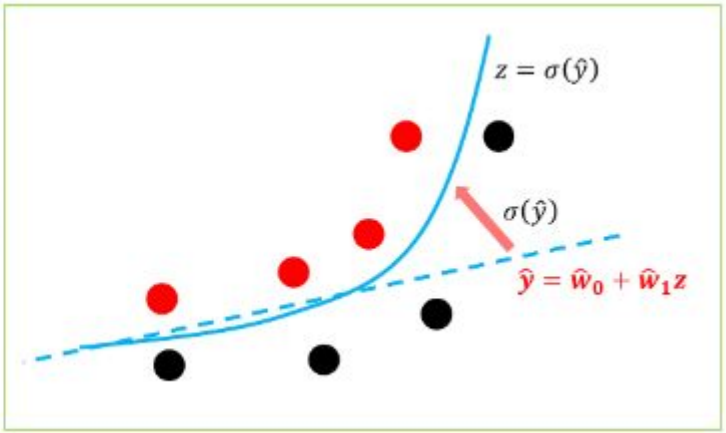
and, or 처럼 단층 퍼셉트론으로는 xor문제를 풀 수 없

선형변환만으로는 이 문제를 풀 수 없음



선형변환에 비선형성을 추가한다면?

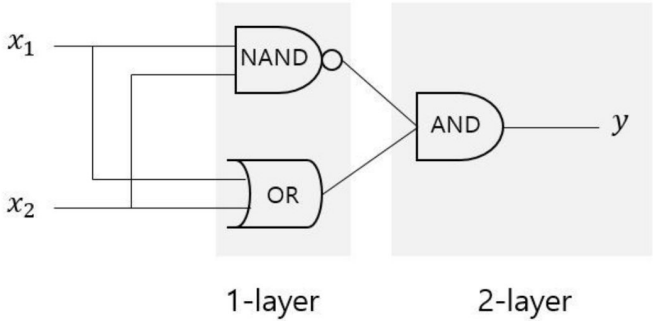




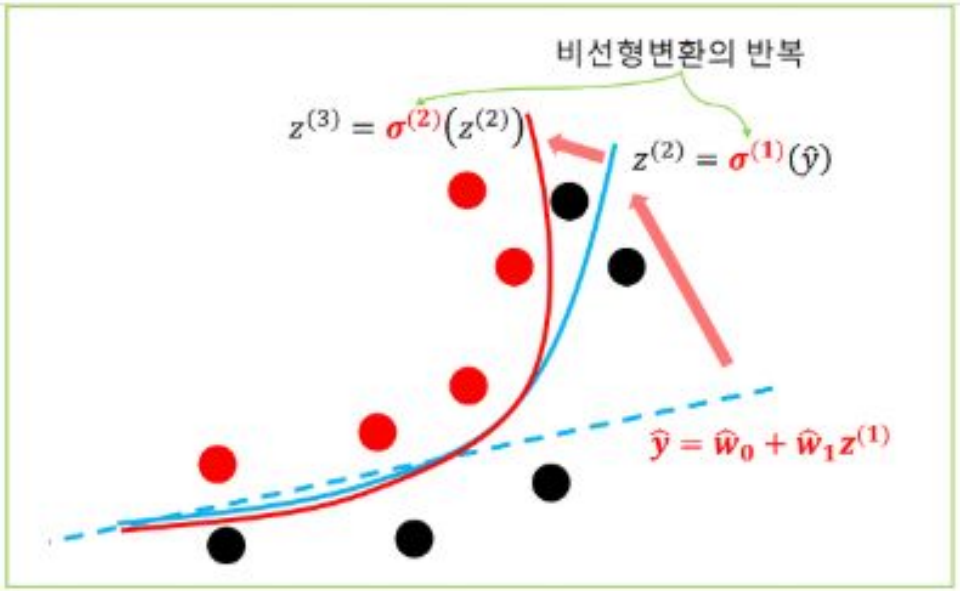
비선형변환

and, or 처럼 단층 퍼셉트론으로는 xor문제를 풀 수 없

선형변환만으로는 이 문제를 풀 수 없음



선형변환에 비선형성을 추가한다면?

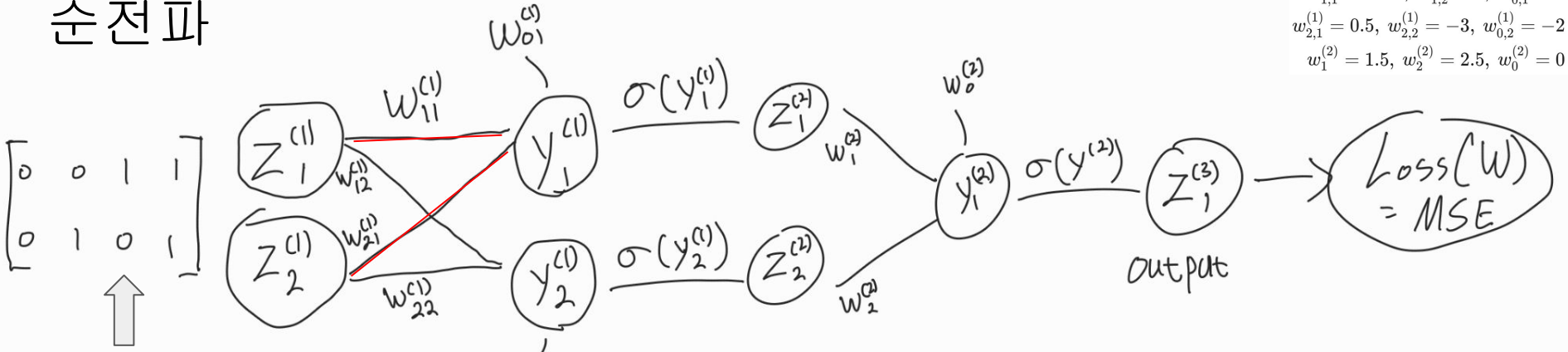


XOR - input (1,0) 을 예제로 알아봅시다

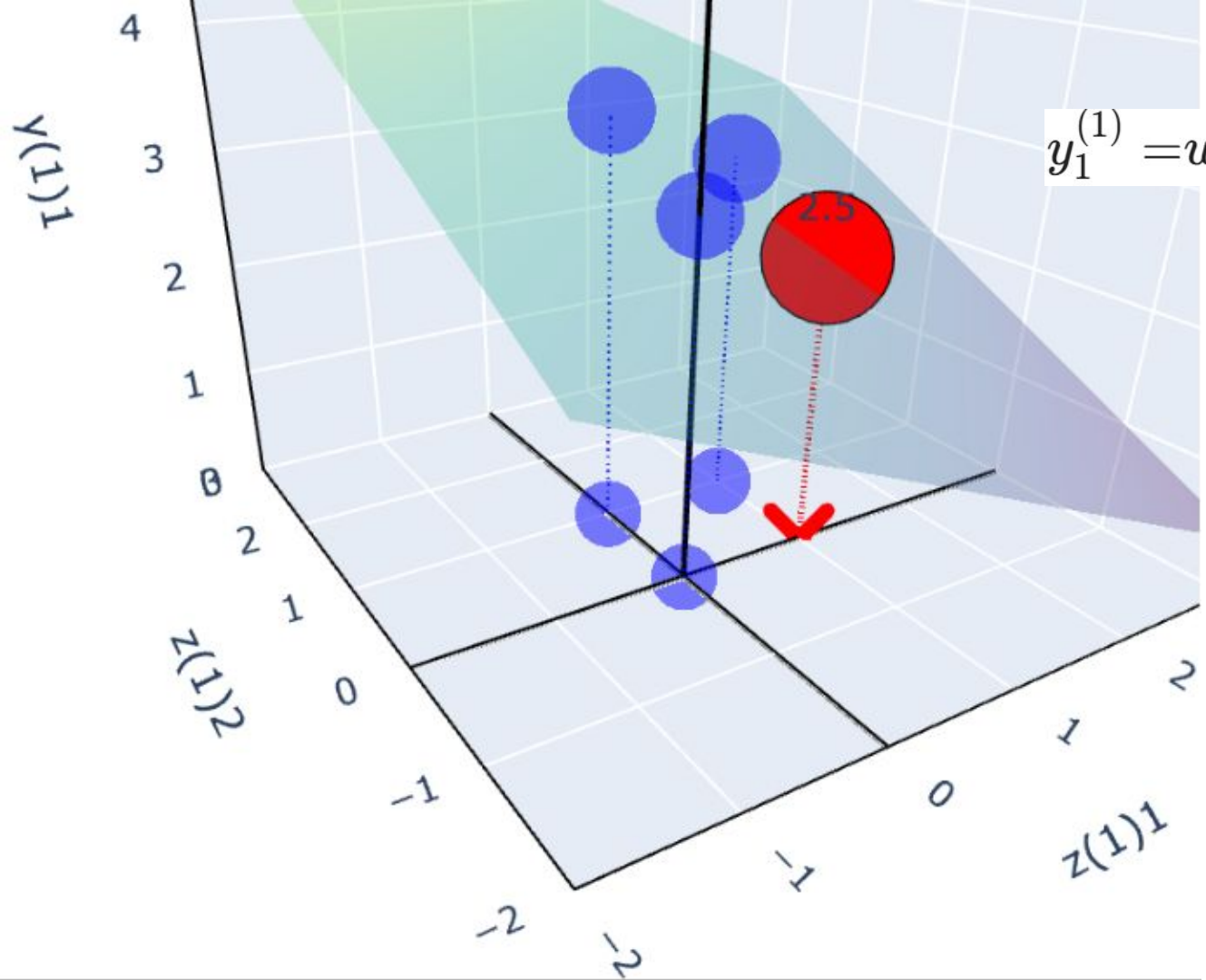
생각해보기
DNN으로 결국 얻고싶은 것은 무엇일까요?

순전파

$$w_{1,1}^{(1)} = -0.5,\ w_{1,2}^{(1)} = 2,\ w_{0,1}^{(1)} = 3$$
$$w_{2,1}^{(1)} = 0.5,\ w_{2,2}^{(1)} = -3,\ w_{0,2}^{(1)} = -2$$
$$w_{1}^{(2)} = 1.5,\ w_{2}^{(2)} = 2.5,\ w_{0}^{(2)} = 0$$



$$y_1^{(1)} = w_{0,1}^{(1)} + w_{1,1}^{(1)} z_1^{(1)} + w_{2,1}^{(1)} z_2^{(1)}$$



$w_{1,1}^{(1)} = -0.5,\ w_{1,2}^{(1)} = 2,\ w_{0,1}^{(1)} = 3$

$w_{2,1}^{(1)} = 0.5,\ w_{2,2}^{(1)} = -3,\ w_{0,2}^{(1)} = -2$

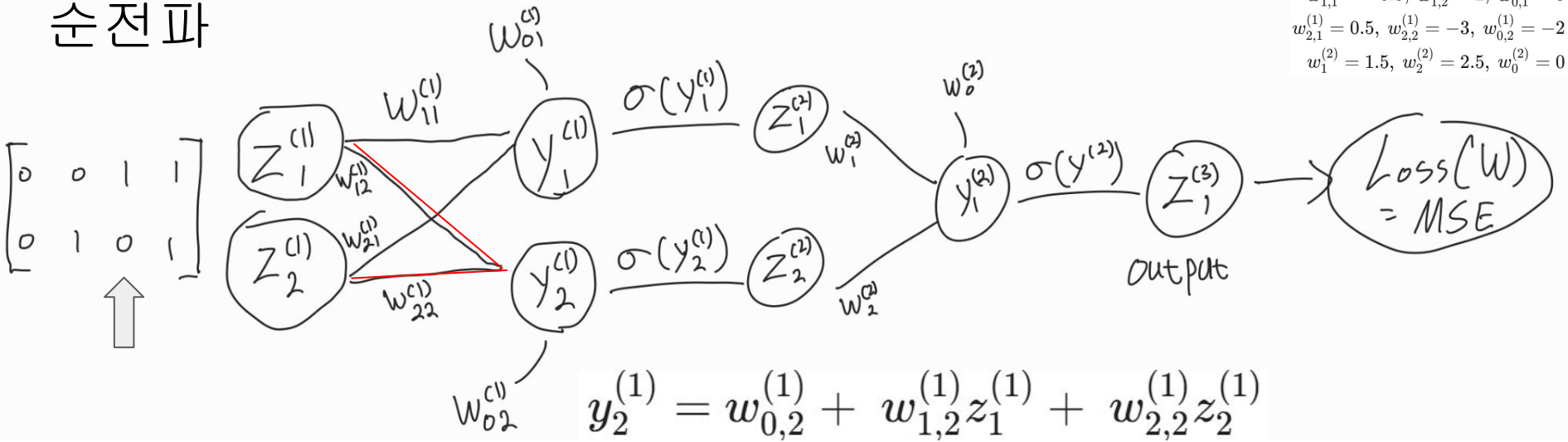
$w_{1}^{(2)} = 1.5,\ w_{2}^{(2)} = 2.5,\ w_{0}^{(2)} = 0$

$$y_1^{(1)} = w_{0,1}^{(1)} + w_{1,1}^{(1)} z_1^{(1)} + w_{2,1}^{(1)} z_2^{(1)}$$

$y_1^{(1)}$ = 2.5

순전파

$$w_{1,1}^{(1)} = -0.5,\ w_{1,2}^{(1)} = 2,\ w_{0,1}^{(1)} = 3$$
$$w_{2,1}^{(1)} = 0.5,\ w_{2,2}^{(1)} = -3,\ w_{0,2}^{(1)} = -2$$
$$w_{1}^{(2)} = 1.5,\ w_{2}^{(2)} = 2.5,\ w_{0}^{(2)} = 0$$



$$y_2^{(1)} = w_{0,2}^{(1)} + w_{1,2}^{(1)} z_1^{(1)} + w_{2,2}^{(1)} z_2^{(1)}$$

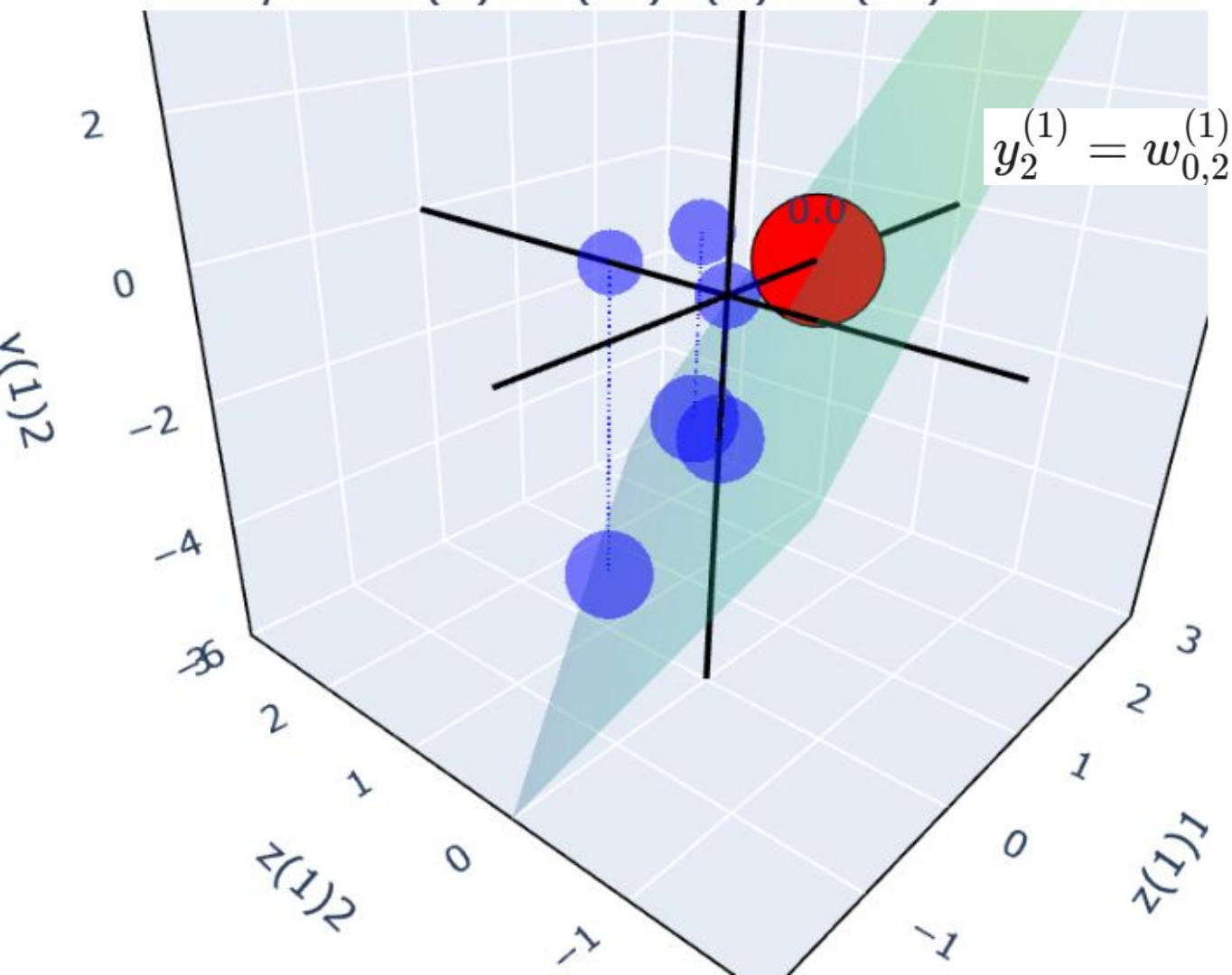
$w_{1,1}^{(1)} = -0.5,\ w_{1,2}^{(1)} = 2,\ w_{0,1}^{(1)} = 3$

$w_{2,1}^{(1)} = 0.5,\ w_{2,2}^{(1)} = -3,\ w_{0,2}^{(1)} = -2$

$w_1^{(2)} = 1.5,\ w_2^{(2)} = 2.5,\ w_0^{(2)} = 0$

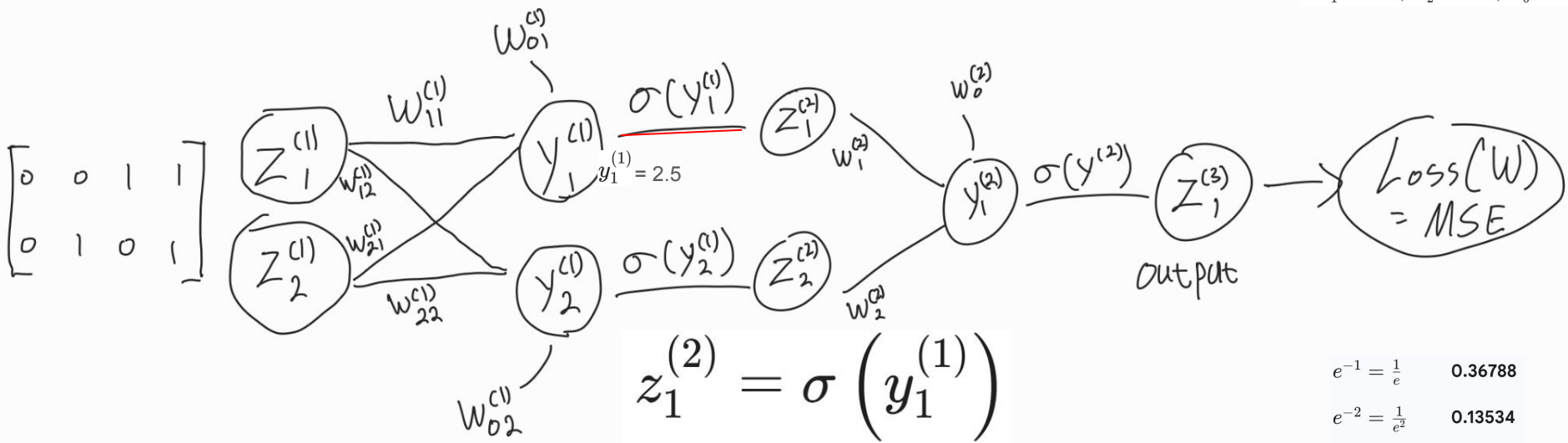
$$y_2^{(1)} = w_{0,2}^{(1)} + w_{1,2}^{(1)} z_1^{(1)} + w_{2,2}^{(1)} z_2^{(1)}$$

$$y_2^{(1)} = 0$$



순전파

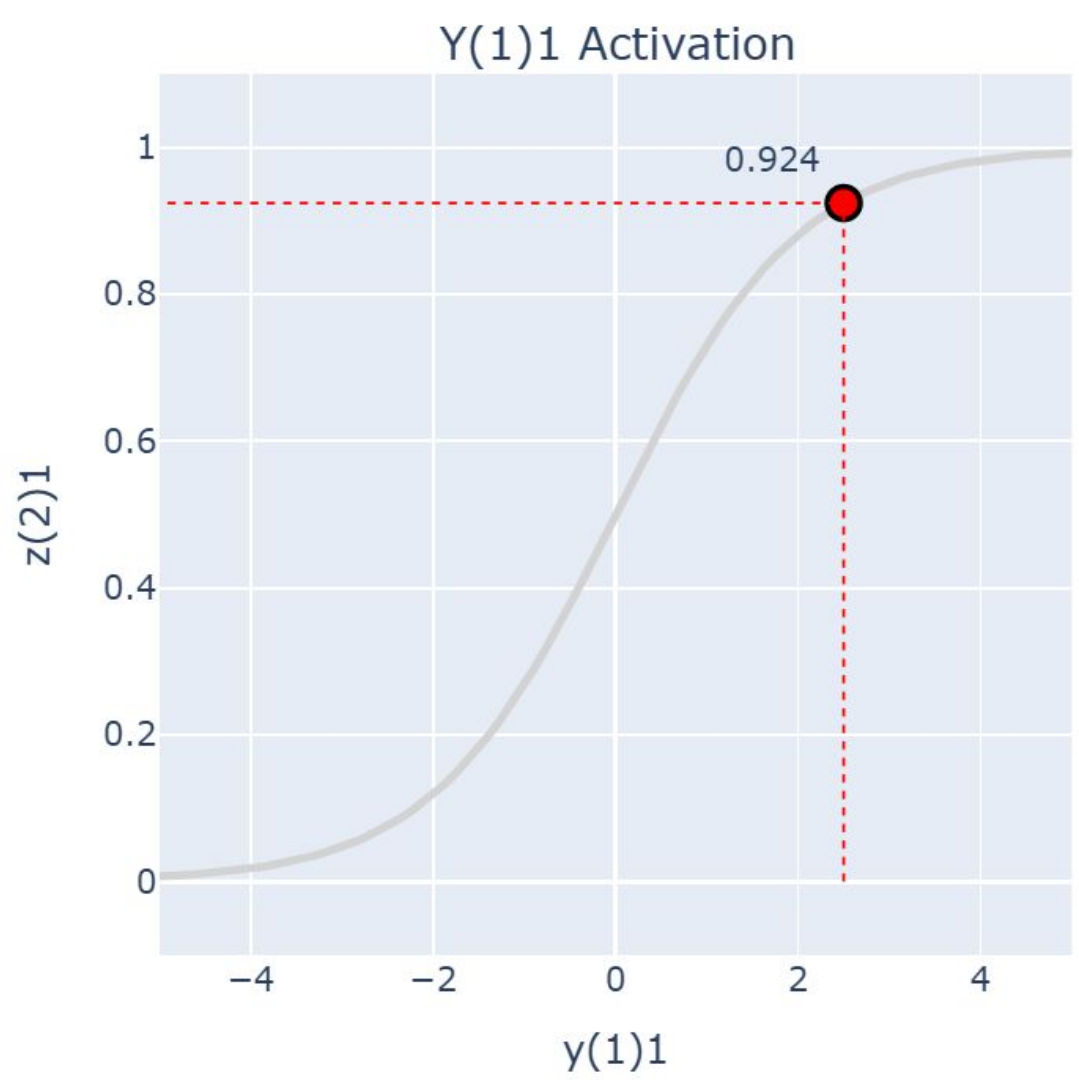
$$w_{1,1}^{(1)} = -0.5,\ w_{1,2}^{(1)} = 2,\ w_{0,1}^{(1)} = 3$$
$$w_{2,1}^{(1)} = 0.5,\ w_{2,2}^{(1)} = -3,\ w_{0,2}^{(1)} = -2$$
$$w_{1}^{(2)} = 1.5,\ w_{2}^{(2)} = 2.5,\ w_{0}^{(2)} = 0$$



$$z_1^{(2)} = \sigma\left(y_1^{(1)}\right)$$

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

$e^{-1} = \frac{1}{e}$	0.36788
$e^{-2} = \frac{1}{e^2}$	0.13534
$e^{-3} = \frac{1}{e^3}$	0.04979
$e^{-4} = \frac{1}{e^4}$	0.01832
$e^{-5} = \frac{1}{e^5}$	0.00674
$e^{-6} = \frac{1}{e^6}$	0.00248
$e^{-7} = \frac{1}{e^7}$	0.00091
$e^{-8} = \frac{1}{e^8}$	0.00034
$e^{-9} = \frac{1}{e^9}$	0.00012
$e^{-10} = \frac{1}{e^{10}}$	0.00005



$w_{1,1}^{(1)} = -0.5,\ w_{1,2}^{(1)} = 2,\ w_{0,1}^{(1)} = 3$

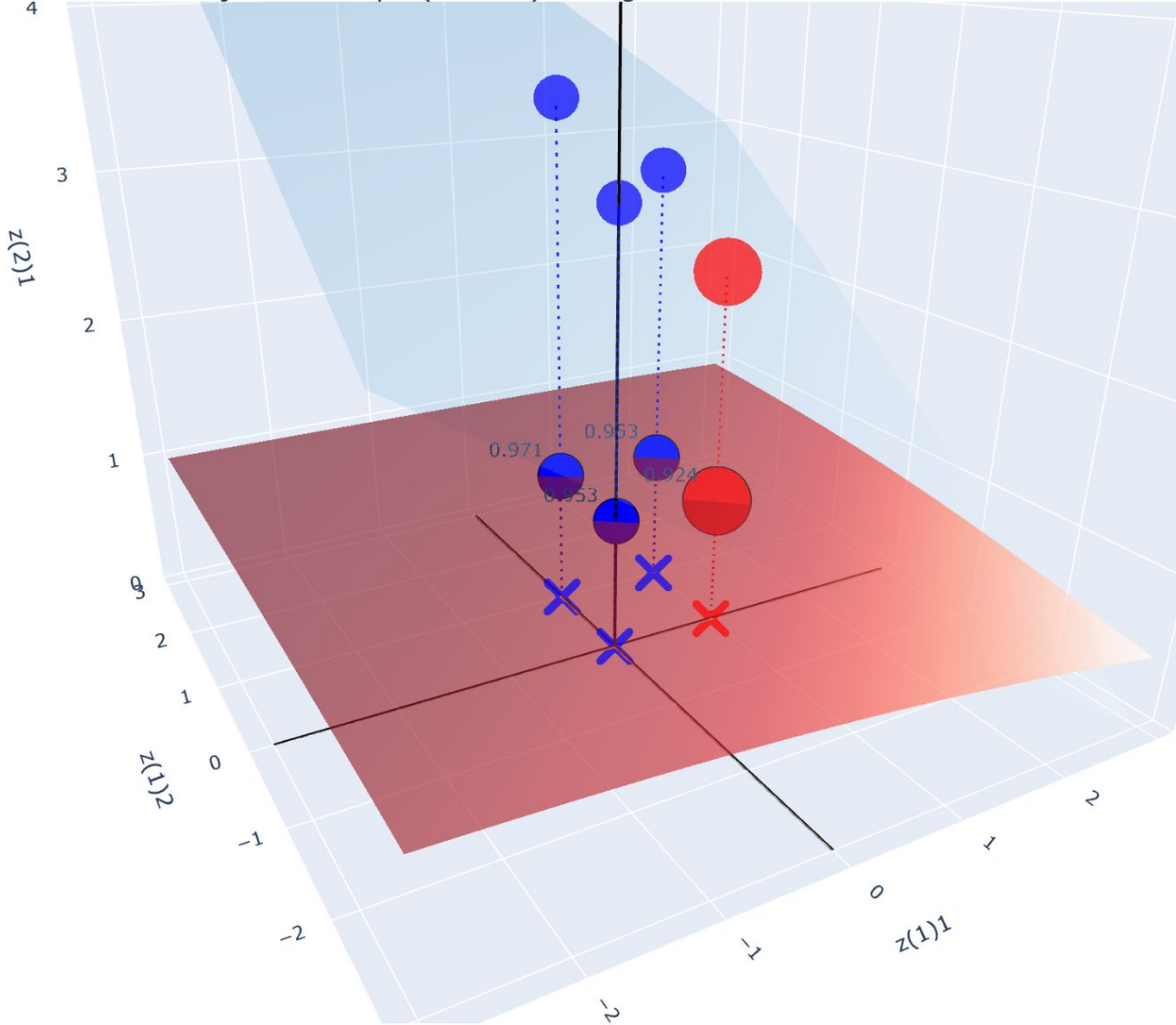
$w_{2,1}^{(1)} = 0.5,\ w_{2,2}^{(1)} = -3,\ w_{0,2}^{(1)} = -2$

$w_{1}^{(2)} = 1.5,\ w_{2}^{(2)} = 2.5,\ w_{0}^{(2)} = 0$

$$z_1^{(2)} = \sigma\left(y_1^{(1)}\right)$$

$y_1^{(1)} = 2.5$

$z_1^{(2)} \approx 0.924$



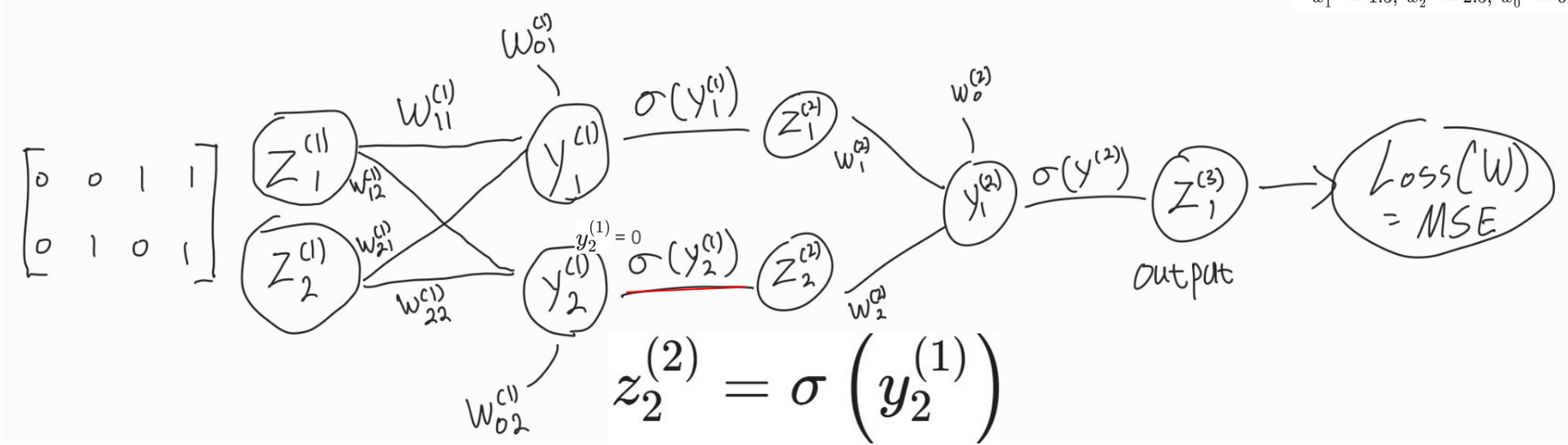
$$z_1^{(2)} = \sigma^{(1)}\left(y_1^{(1)}\right),$$

$$y_1^{(1)} = w_{0,1}^{(1)} + w_{1,1}^{(1)} z_1^{(1)} + w_{2,1}^{(1)} z_2^{(1)}$$

$$\rightarrow z_1^{(2)} = \sigma^{(1)}\left(w_{0,1}^{(1)} + w_{1,1}^{(1)} z_1^{(1)} + w_{2,1}^{(1)} z_2^{(1)}\right)$$

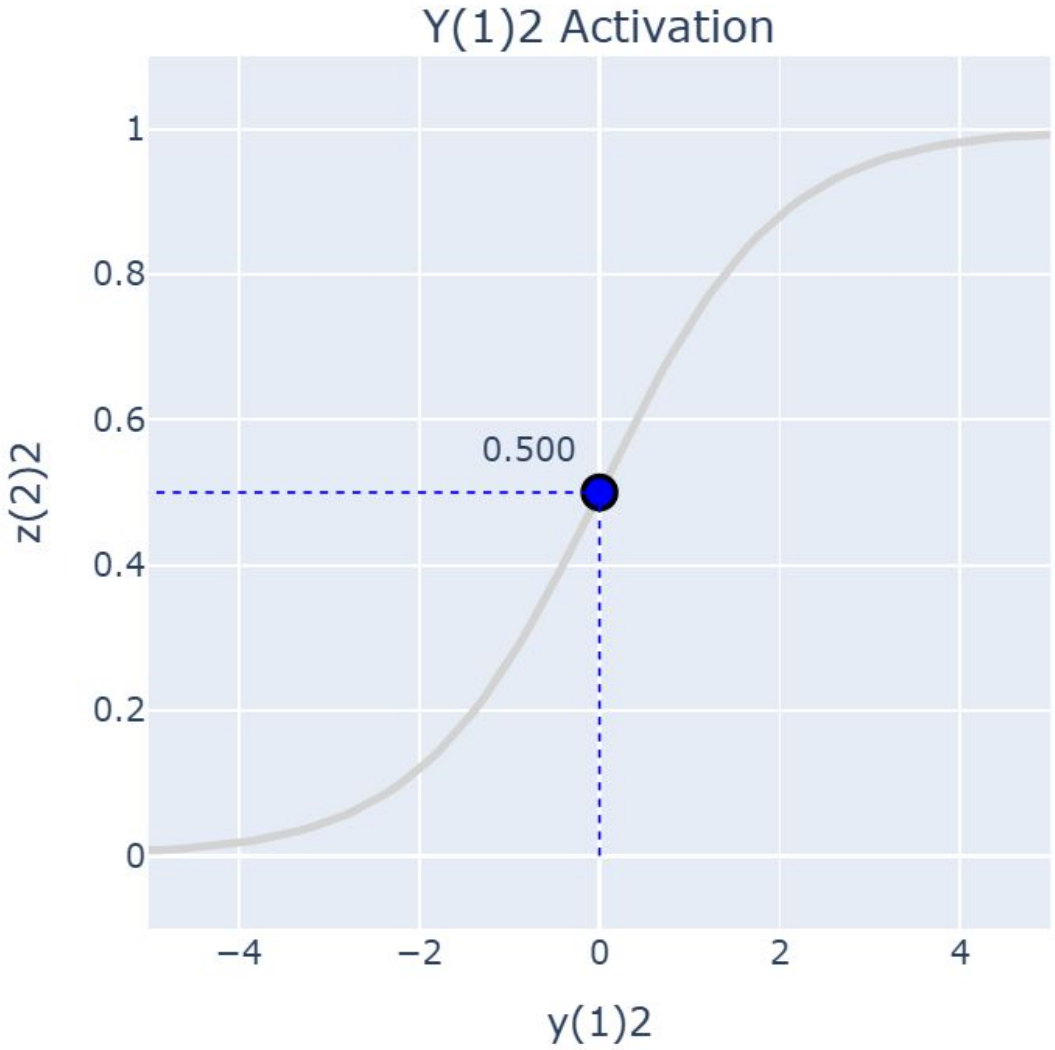
순전파

$w_{1,1}^{(1)} = -0.5,\ w_{1,2}^{(1)} = 2,\ w_{0,1}^{(1)} = 3$
$w_{2,1}^{(1)} = 0.5,\ w_{2,2}^{(1)} = -3,\ w_{0,2}^{(1)} = -2$
$w_{1}^{(2)} = 1.5,\ w_{2}^{(2)} = 2.5,\ w_{0}^{(2)} = 0$



$$z_2^{(2)} = \sigma\left(y_2^{(1)}\right)$$

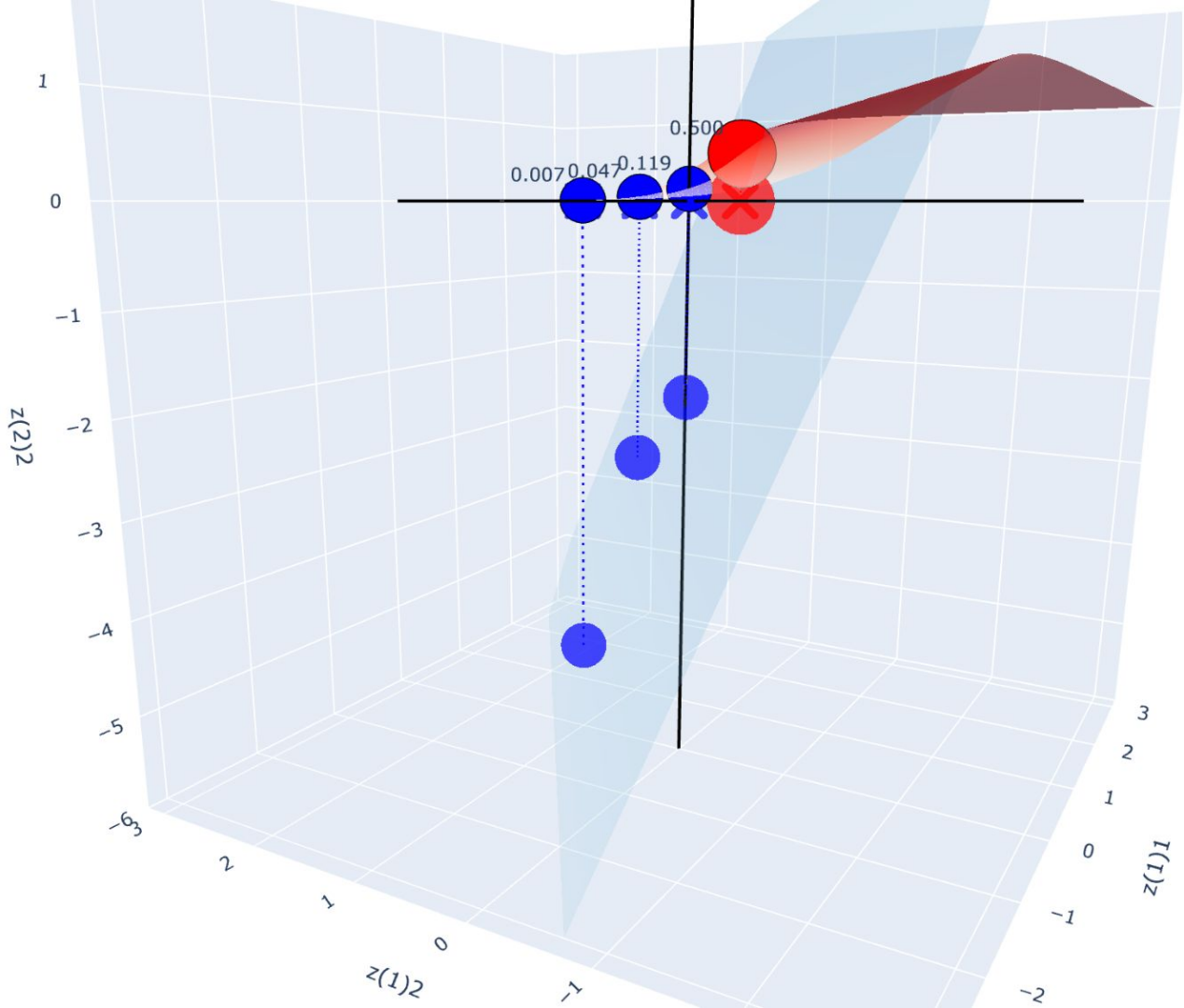
$w_{1,1}^{(1)} = -0.5,\ w_{1,2}^{(1)} = 2,\ w_{0,1}^{(1)} = 3$
$w_{2,1}^{(1)} = 0.5,\ w_{2,2}^{(1)} = -3,\ w_{0,2}^{(1)} = -2$
$w_{1}^{(2)} = 1.5,\ w_{2}^{(2)} = 2.5,\ w_{0}^{(2)} = 0$



$$z_2^{(2)} = \sigma\left(y_2^{(1)}\right)$$

$y_2^{(1)} = 0$

$z_2^{(2)} = 0.5$


0.500
0.119
0.047
0.007
z(2)2
z(1)2
z(1)1

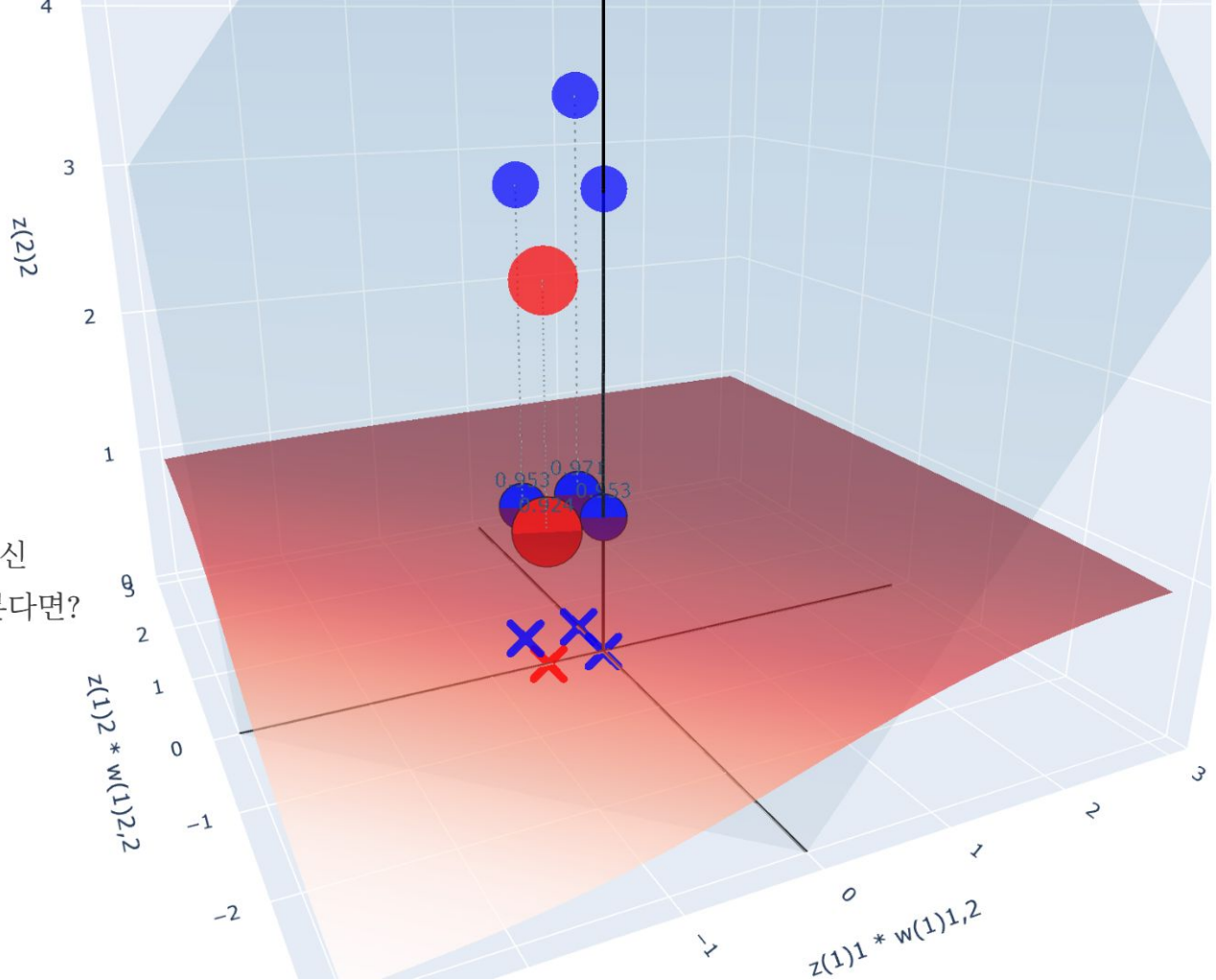
$$z_2^{(2)} = \sigma^{(1)}\left(y_2^{(1)}\right),$$

$$y_2^{(1)} = w_{0,2}^{(1)} + w_{1,2}^{(1)} z_1^{(1)} + w_{2,2}^{(1)} z_2^{(1)}$$

$$\rightarrow z_2^{(2)} = \sigma^{(1)}\left(w_{0,2}^{(1)} + w_{1,2}^{(1)} z_1^{(1)} + w_{2,2}^{(1)} z_2^{(1)}\right)$$

x축과 y축에 $input$으로 들어오는$z_1^{(1)}, z_2^{(1)}$ 대신

$w_{1,2}^{(1)} * z_1^{(1)}$, $w_{2,2}^{(1)} * z_2^{(1)}$ 이 들어온 상황을 생각해본다면?



순전파

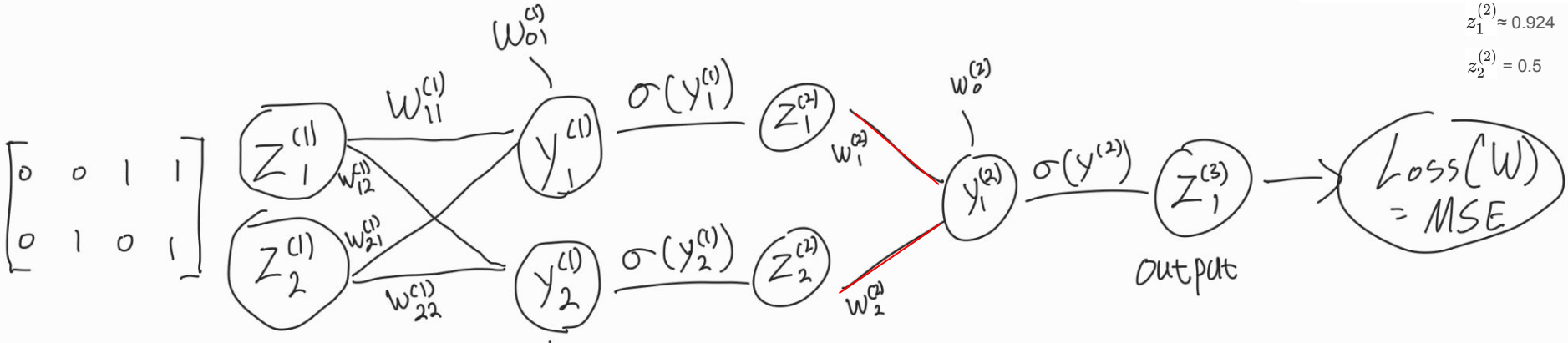
$w_{1,1}^{(1)} = -0.5,\ w_{1,2}^{(1)} = 2,\ w_{0,1}^{(1)} = 3$

$w_{2,1}^{(1)} = 0.5,\ w_{2,2}^{(1)} = -3,\ w_{0,2}^{(1)} = -2$

$w_{1}^{(2)} = 1.5,\ w_{2}^{(2)} = 2.5,\ w_{0}^{(2)} = 0$

$z_1^{(2)} \approx 0.924$

$z_2^{(2)} = 0.5$



$$y_1^{(2)} = w_0^{(2)} + w_1^{(2)}\ z_1^{(2)} + w_2^{(2)}\ z_2^{(2)}$$

$$y_1^{(2)} \approx 2.64$$

$w_{1,1}^{(1)} = -0.5,\ w_{1,2}^{(1)} = 2,\ w_{0,1}^{(1)} = 3$

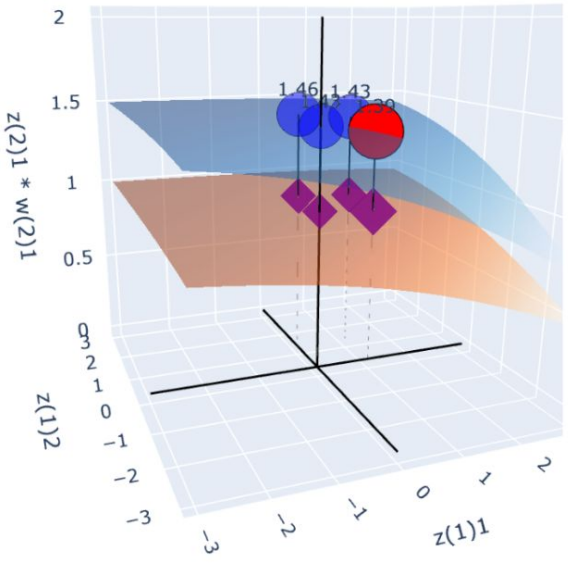
$w_{2,1}^{(1)} = 0.5,\ w_{2,2}^{(1)} = -3,\ w_{0,2}^{(1)} = -2$

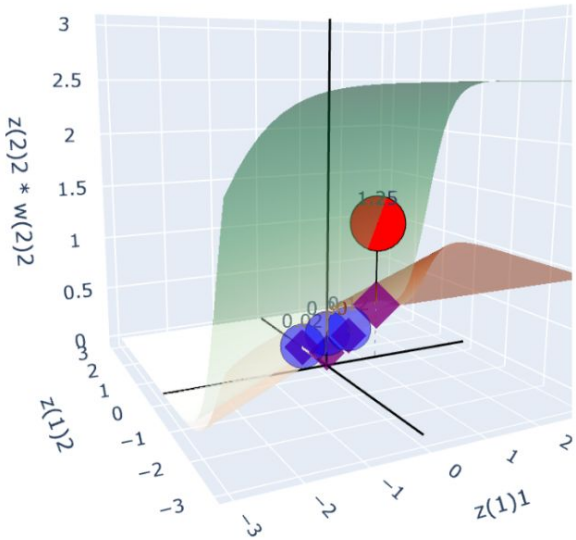
$w_1^{(2)} = 1.5,\ w_2^{(2)} = 2.5,\ w_0^{(2)} = 0$

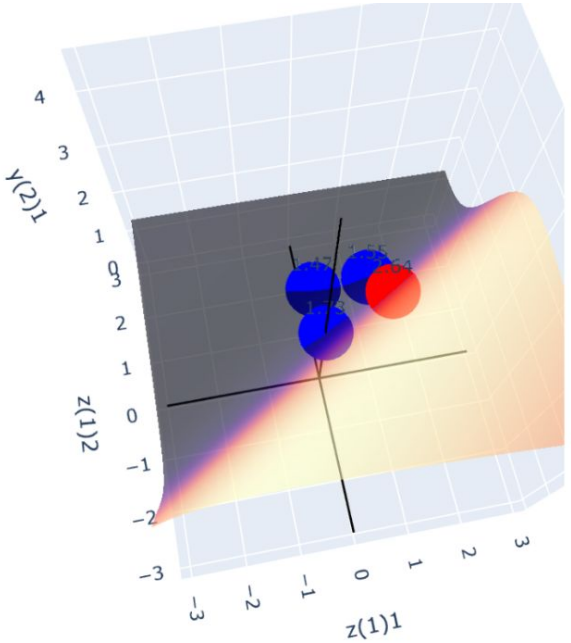
$z_1^{(2)} \approx 0.924$

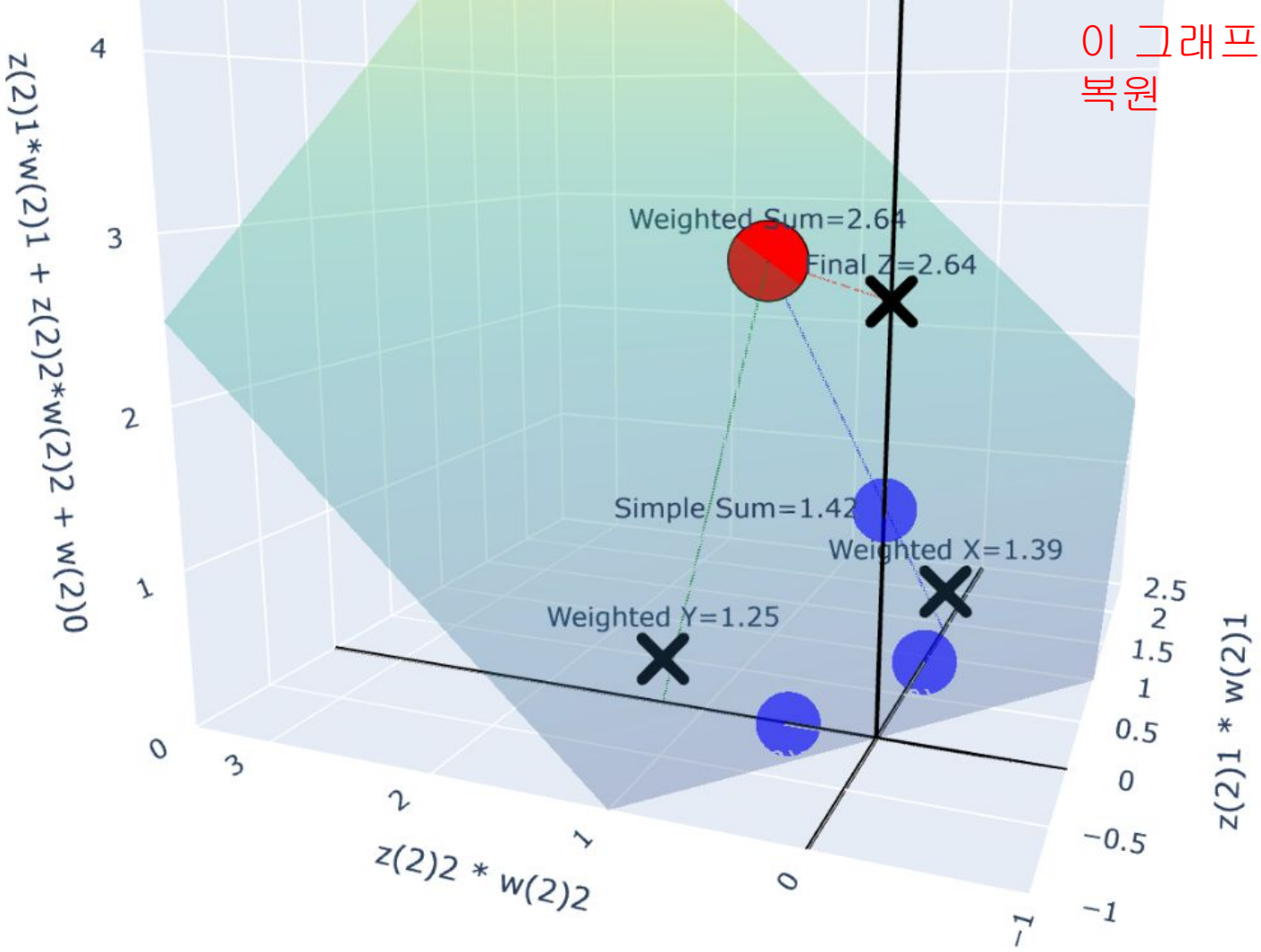
$z_2^{(2)} = 0.5$

$$y_1^{(2)} = w_0^{(2)} + w_1^{(2)}\ z_1^{(2)} + w_2^{(2)}\ z_2^{(2)}$$



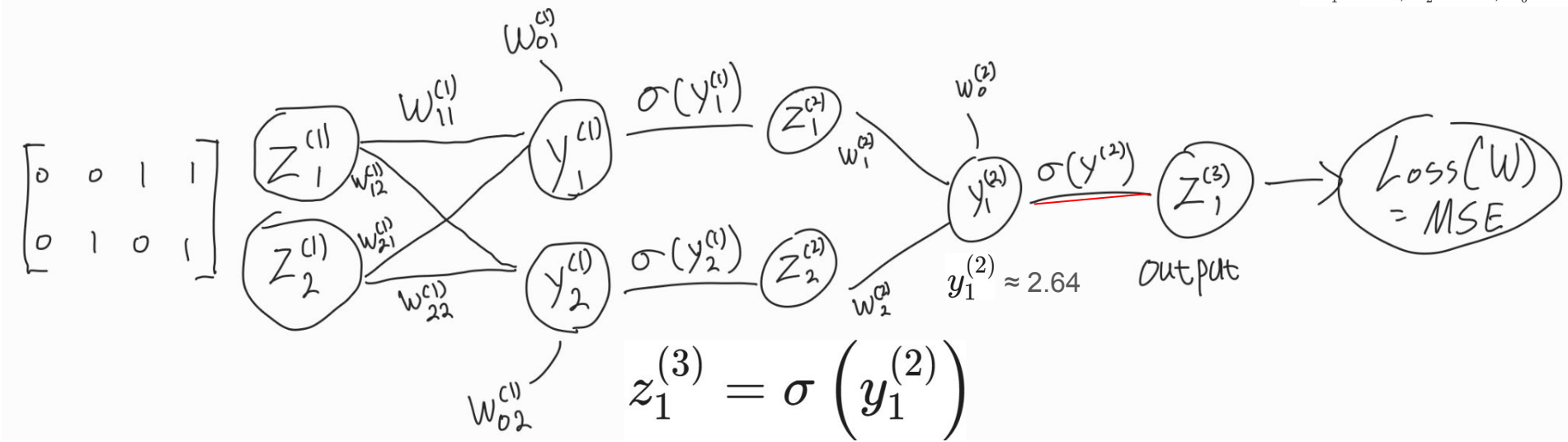





이 그래프 코드로 복원
Weighted Sum=2.64
Final Z=2.64
Simple Sum=1.42
Weighted X=1.39
Weighted Y=1.25
z(2)1*w(2)1 + z(2)2*w(2)2 + w(2)0
z(2)2 * w(2)2
z(2)1 * w(2)1

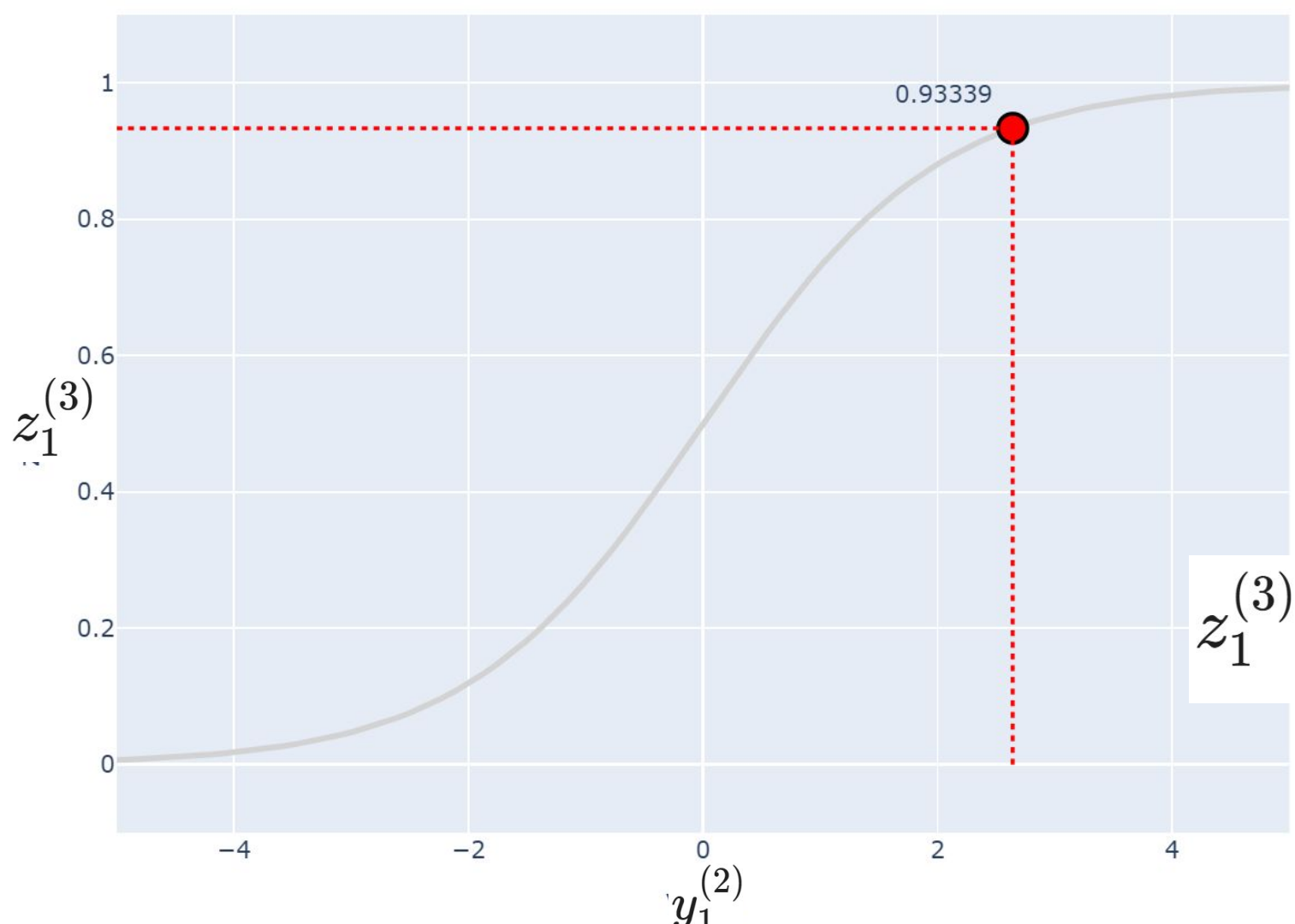
순전파

$$w_{1,1}^{(1)} = -0.5,\ w_{1,2}^{(1)} = 2,\ w_{0,1}^{(1)} = 3$$
$$w_{2,1}^{(1)} = 0.5,\ w_{2,2}^{(1)} = -3,\ w_{0,2}^{(1)} = -2$$
$$w_{1}^{(2)} = 1.5,\ w_{2}^{(2)} = 2.5,\ w_{0}^{(2)} = 0$$



$y_1^{(2)} \approx 2.64$

$$z_1^{(3)} = \sigma\left(y_1^{(2)}\right)$$

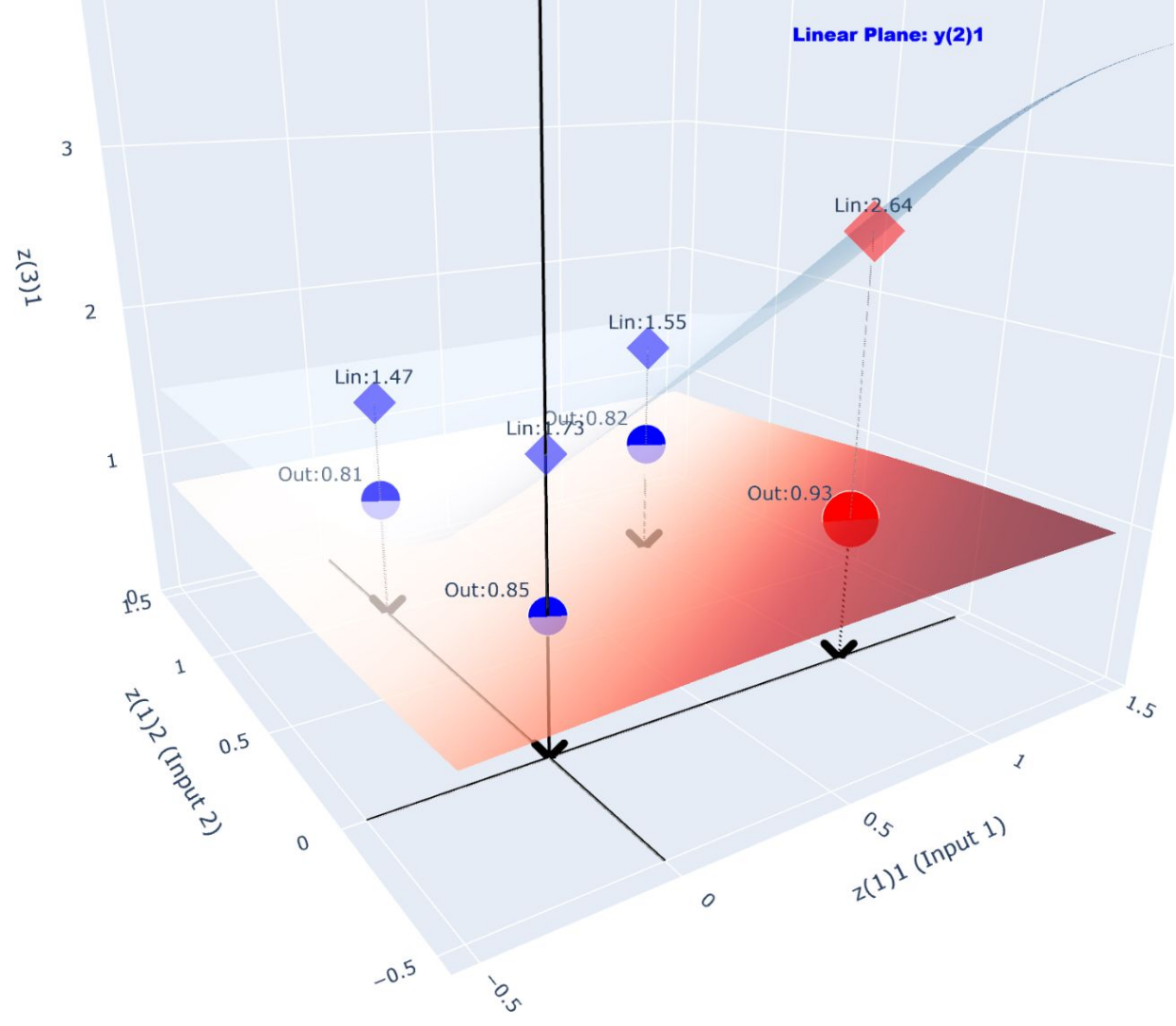


$$w_{1,1}^{(1)} = -0.5,\ w_{1,2}^{(1)} = 2,\ w_{0,1}^{(1)} = 3$$
$$w_{2,1}^{(1)} = 0.5,\ w_{2,2}^{(1)} = -3,\ w_{0,2}^{(1)} = -2$$
$$w_1^{(2)} = 1.5,\ w_2^{(2)} = 2.5,\ w_0^{(2)} = 0$$

$$z_1^{(3)} = \sigma\left(y_1^{(2)}\right)$$

$$y_1^{(2)} \approx 2.64$$

$$z_1^{(3)} \approx 0.93339$$



$$z_1^{(3)} = \sigma\left(y_1^{(2)}\right)$$

$$y_1^{(2)} \approx 2.64$$

$$z_1^{(3)} \approx 0.93339$$

Loss

입력		출력
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

w 파라미터들을 무작위로 설정했을 때

$$w_{1,1}^{(1)} = -0.5,\ w_{1,2}^{(1)} = 2,\ w_{0,1}^{(1)} = 3$$
$$w_{2,1}^{(1)} = 0.5,\ w_{2,2}^{(1)} = -3,\ w_{0,2}^{(1)} = -2$$
$$w_{1}^{(2)} = 1.5,\ w_{2}^{(2)} = 2.5,\ w_{0}^{(2)} = 0$$

input (1,0) 의 예측 y값 = $z_1^{(3)}$ ≈ 0.93339

실제 정답 (1) 과의 차이 ≈ 0.07

```
입력(Input)       | 정답(Target)   | 예측값(Pred)       | 손실(Loss)
------------------------------------------------------------------
[0 0]             | 0              | 0.84901          | 1.89055
[0 1]             | 1              | 0.81348          | 0.20644
[1 0]             | 1              | 0.93316          | 0.06918
[1 1]             | 0              | 0.82454          | 1.74035
------------------------------------------------------------------
전체 평균 손실 (Mean Loss): 0.97663
```

하지만 다른 값들은 정답 - 예측 간의 차이가 심하게 발생

입력 A, B에 대해서 Y를 잘 예측하게 하는 W 파라미터들을

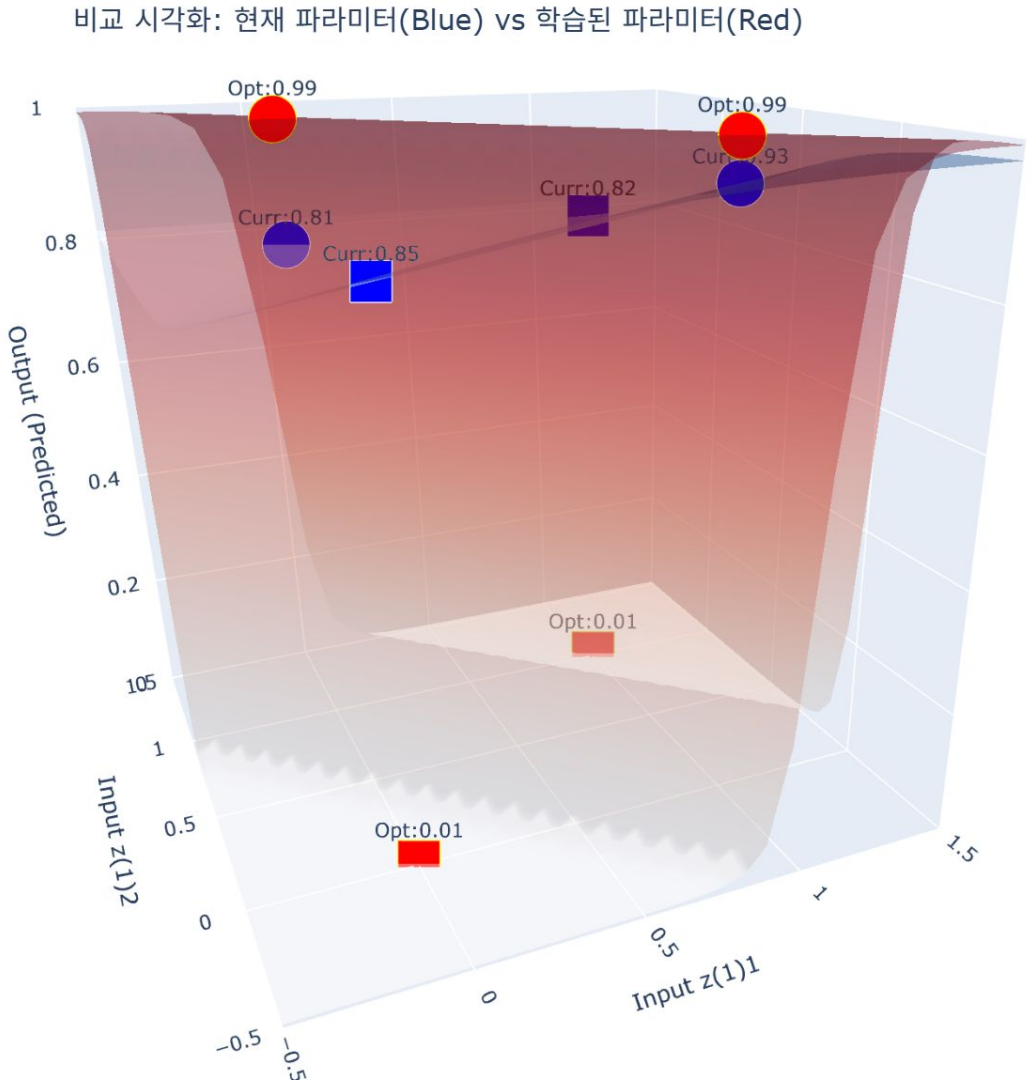
어떻게 수정해야 하는지 알아봅시다

Loss

이 오차를 어떻게 줄여야 할까요?

학습 대상인 w파라미터들을 수정

얼마나? 양수로? 음수로?



입력(Input)	정답(Target)	예측값(Pred)	손실(Loss)
[0 0]	0	0.84901	1.89055
[0 1]	1	0.81348	0.20644
[1 0]	1	0.93316	0.06918
[1 1]	0	0.82454	1.74035

전체 평균 손실 (Mean Loss): 0.97663

Loss

유의미한 연관성이 보이시나요?

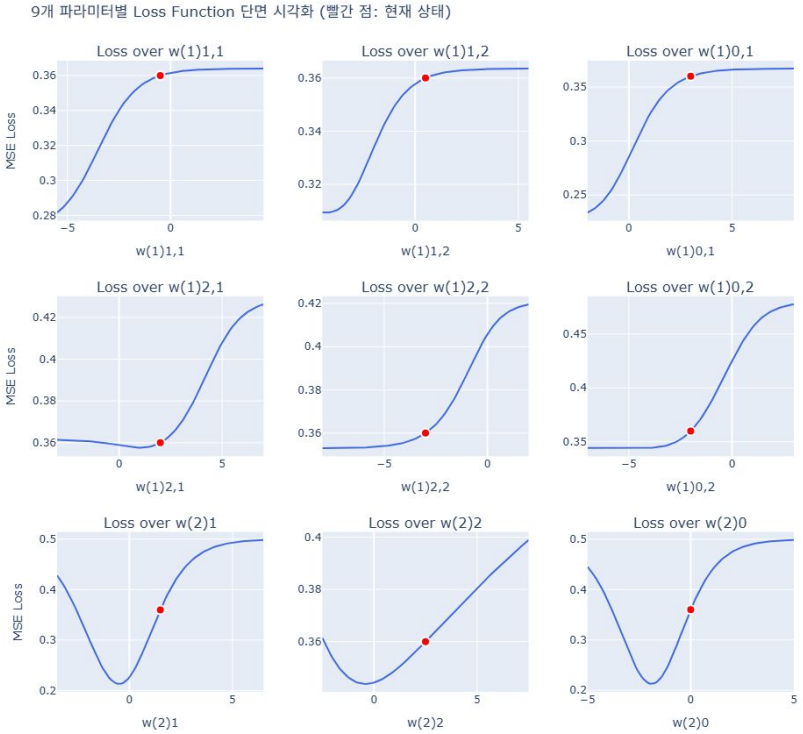
동일한 y노드로 가는 w들이 비슷한 경향

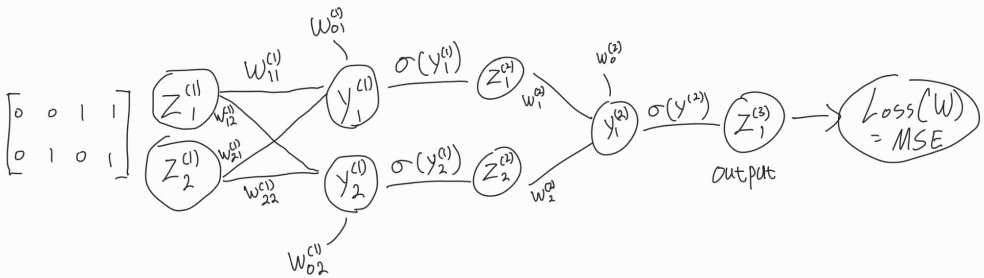
편향 w는 고정하고, 선형결합되는 y에 대한

$w_{0,1}^{(1)}, \quad w_{0,2}^{(1)}, \quad w_0^{(2)}$ $\qquad$ $y_1^{(1)}, \quad y_2^{(1)}, \quad y_1^{(2)}$

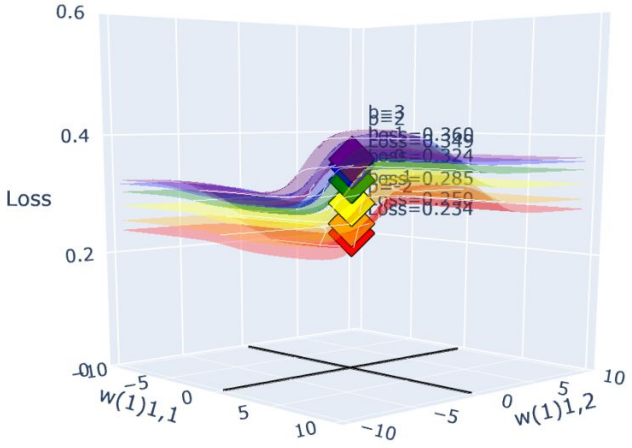
w들을 x, y로 갖는 3차원 Loss함수를 시각화

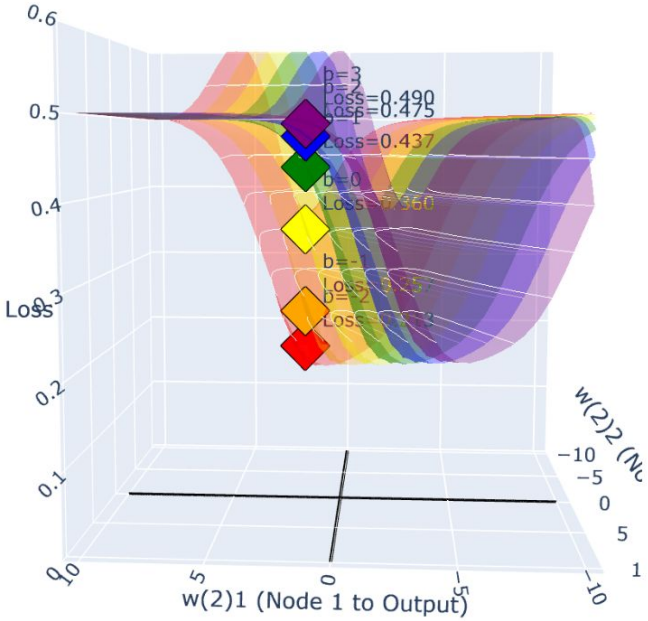
$$\begin{bmatrix} w_{1,1}^{(1)}, & w_{1,2}^{(1)} \end{bmatrix}$$
$$\begin{bmatrix} w_{2,1}^{(1)}, & w_{2,2}^{(1)} \end{bmatrix}$$
$$\begin{bmatrix} w_1^{(2)}, & w_2^{(2)} \end{bmatrix}$$

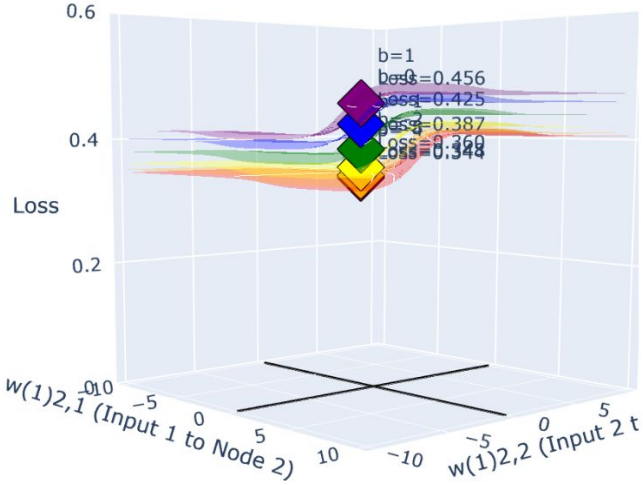




Loss


0.6
0.4
0.2
Loss
b=3
b=2
b=1
b=0
Loss=0.360
Loss=0.324
Loss=0.285
Loss=0.234
w(1)1,1
w(1)1,2


0.6
0.5
0.4
0.3
0.2
0.1
Loss
b=3
b=2
Loss=0.490
Loss=0.475
b=1
Loss=0.437
b=0
w(2)1 (Node 1 to Output)
w(2)2 (No


0.6
0.4
0.2
Loss
b=1
b=0
Loss=0.456
Loss=0.425
Loss=0.387
w(1)2,1 (Input 1 to Node 2)
w(1)2,2 (Input 2 t

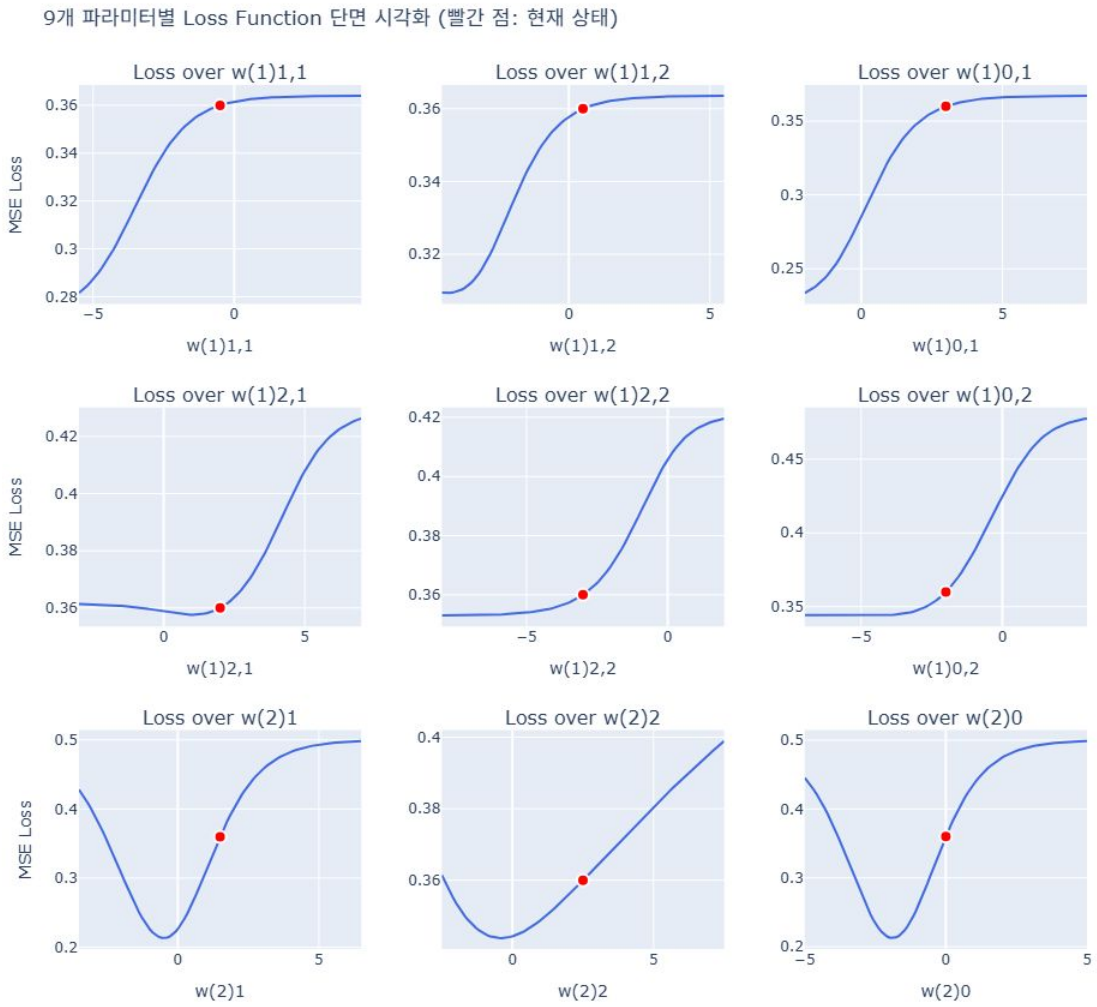
Loss

w별 gradient 확인

각 w에 대한 Loss를 측정

$$w_{new} = w_{old} - \eta \cdot \frac{\partial L}{\partial w}$$

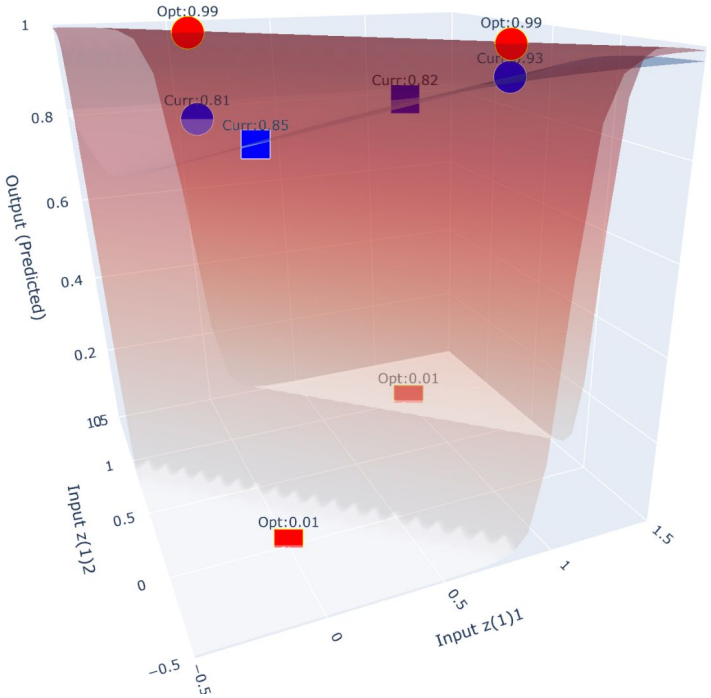
$$w_{1,1}^{(1)} = -0.5,\ w_{1,2}^{(1)} = 2,\ w_{0,1}^{(1)} = 3$$
$$w_{2,1}^{(1)} = 0.5,\ w_{2,2}^{(1)} = -3,\ w_{0,2}^{(1)} = -2$$
$$w_{1}^{(2)} = 1.5,\ w_{2}^{(2)} = 2.5,\ w_{0}^{(2)} = 0$$

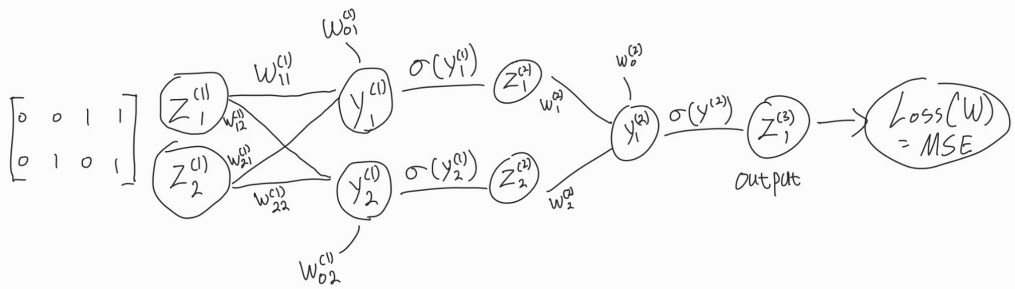


역전파

$w_{1,1}^{(1)}$ 을 타겟으로 설정하고 역전파

비교 시각화: 현재 파라미터(Blue) vs 학습된 파라미터(Red)





$$\frac{\partial Loss}{\partial w_{1,1}^{(1)}} = \frac{\partial Loss}{\partial z_1^{(3)}} * \frac{\partial z_1^{(3)}}{\partial y_1^{(2)}} * \frac{\partial y_1^{(2)}}{\partial z_1^{(2)}} * \frac{\partial z_1^{(2)}}{\partial y_1^{(1)}} * \frac{\partial \mathbf{y_1^{(1)}}}{\partial \mathbf{w_{1,1}^{(1)}}}$$

입력(Input)	정답(Target)	예측값(Pred)	손실(Loss)
[0 0]	0	0.84901	1.89055
[0 1]	1	0.81348	0.20644
[1 0]	1	0.93316	0.06918
[1 1]	0	0.82454	1.74035

전체 평균 손실 (Mean Loss): 0.97663

역전파

(0.9331)*(1-0.9331)≈0.0624

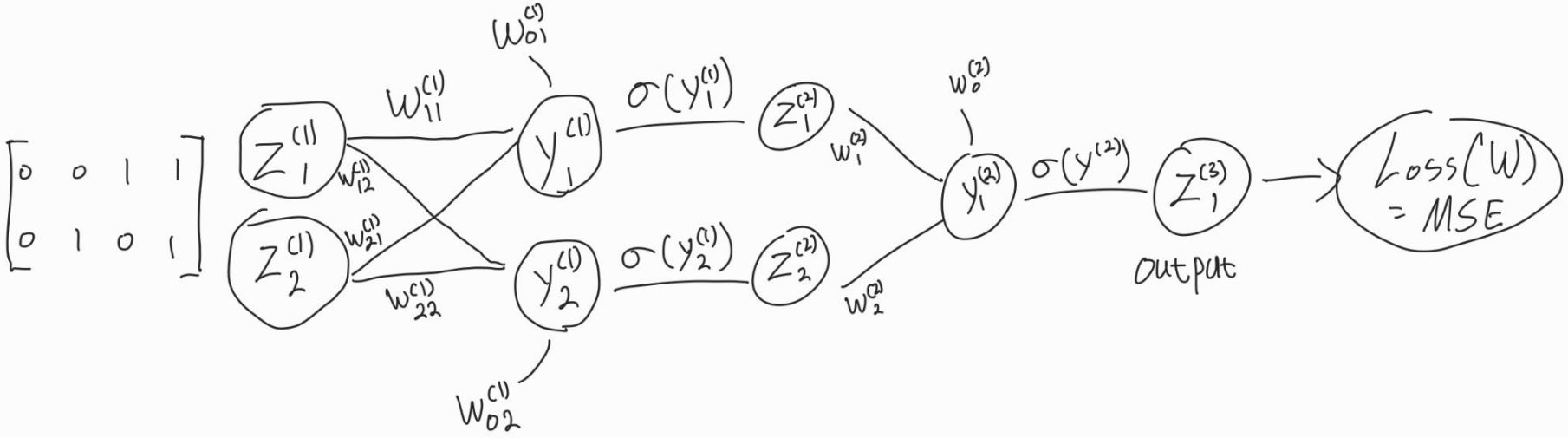
(0.9241)*(1-0.9241)≈0.0701

$$\frac{\partial Loss}{\partial w_{1,1}^{(1)}} = \frac{\partial Loss}{\partial z_1^{(3)}} * \frac{\partial z_1^{(3)}}{\partial y_1^{(2)}} * \frac{\partial y_1^{(2)}}{\partial z_1^{(2)}} * \frac{\partial z_1^{(2)}}{\partial y_1^{(1)}} * \frac{\partial \mathbf{y}_1^{(1)}}{\partial \mathbf{w}_{1,1}^{(1)}}$$

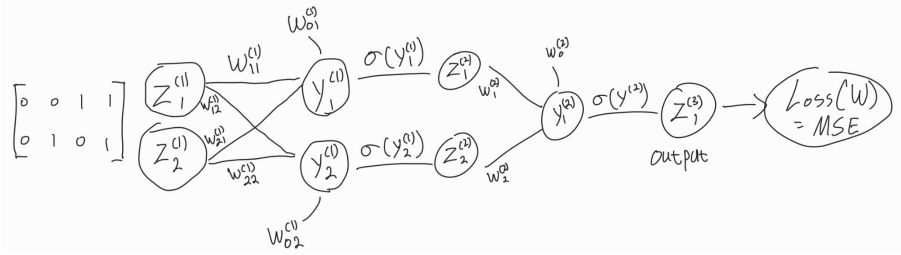
-1(1-0.9331) = -0.0669

$w_1^{(2)} = 1.5$

$z_1^{(1)} = 1$



역전파

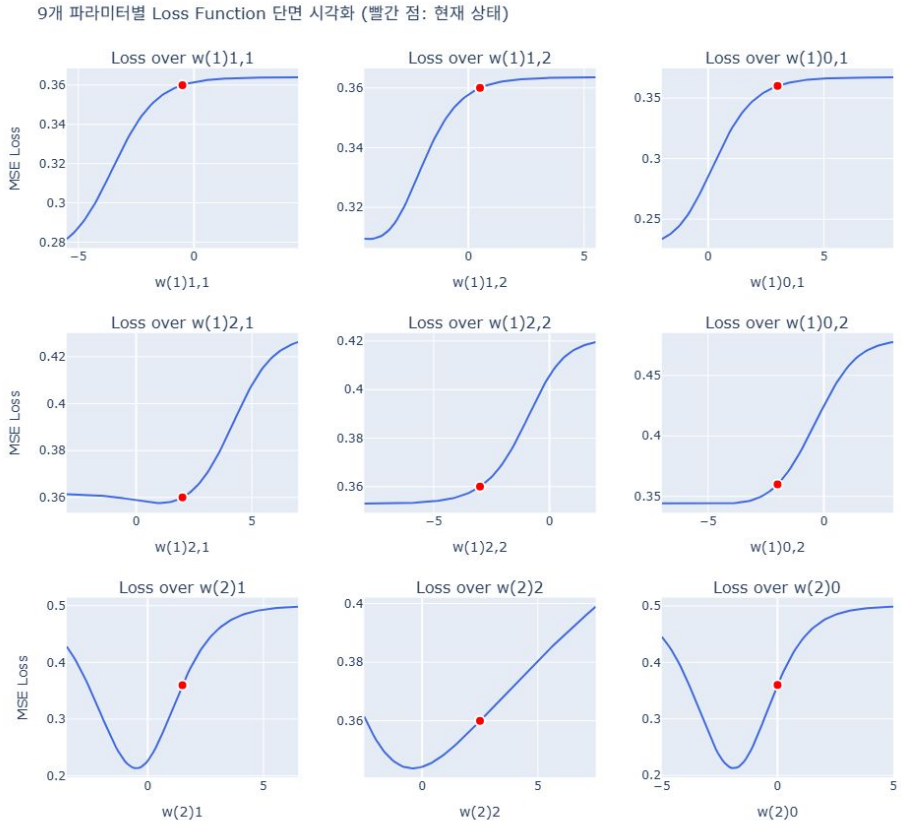


$$
\begin{aligned}
\text{Gradient} &= (-0.0669) \times (0.0624) \times (1.5) \times (0.0701) \times (1) \\
&= (-0.00417) \times (1.5) \times (0.0701) \\
&= -0.00625 \times 0.0701 \\
&\approx \mathbf{-0.000438}
\end{aligned}
$$

파라미터 업데이트

어느 방향으로 얼마나 수정해야 하는지 알 수 있는 **Loss**함수

각 **w**에 대해서 모두 역전파

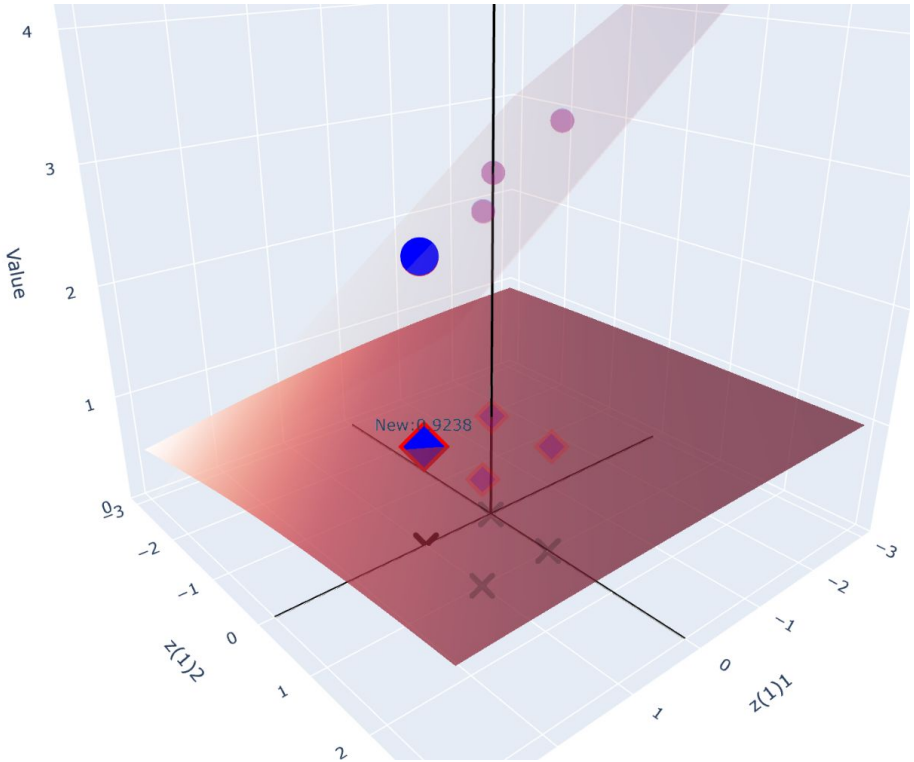


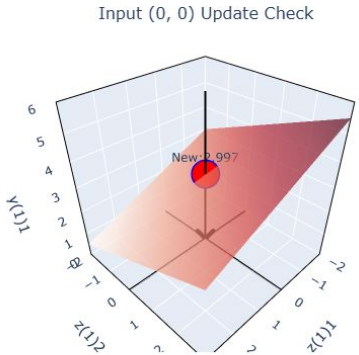
파라미터 업데이트

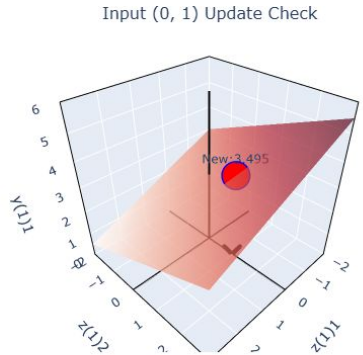
lr = 1.0

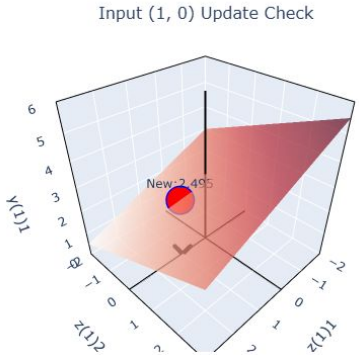
Parameter	Old Value	Gradient	New Value	Change
w11	-0.500000	0.001911	-0.501911	-0.001911
w21	0.500000	0.001719	0.498281	-0.001719
b1	3.000000	0.003453	2.996547	-0.003453
w12	2.000000	0.002717	1.997283	-0.002717
w22	-3.000000	0.003251	-3.003251	-0.003251
b2	-2.000000	0.009741	-2.009741	-0.009741
w_out1	1.500000	0.046495	1.453505	-0.046495
w_out2	2.500000	0.004089	2.495911	-0.004089
b_out	0.000000	0.048913	-0.048913	-0.048913

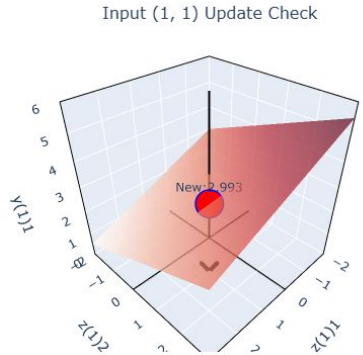
순전파


Value
New:0.9238
z(1)1
z(1)2


Input (0, 0) Update Check
New:2.997
y(1)1
z(1)1
z(1)2

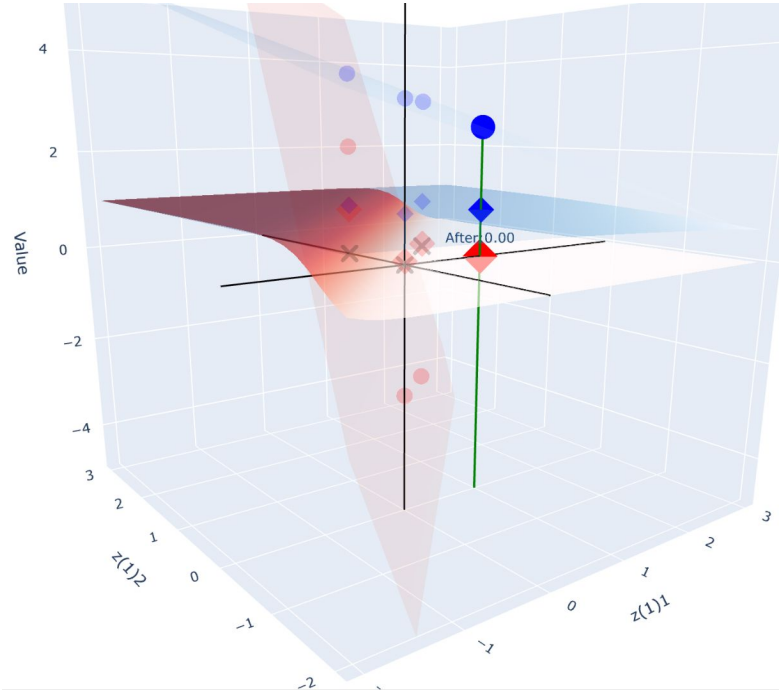

Input (0, 1) Update Check
New:3.495
y(1)1
z(1)1
z(1)2

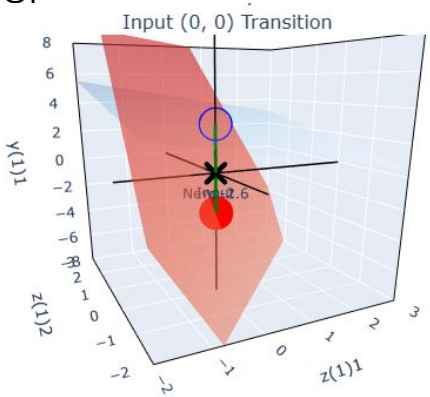

Input (1, 0) Update Check
New:2.495
y(1)1
z(1)1
z(1)2

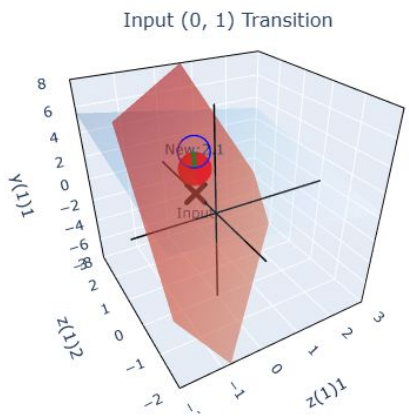

Input (1, 1) Update Check
New:2.993
y(1)1
z(1)1
z(1)2

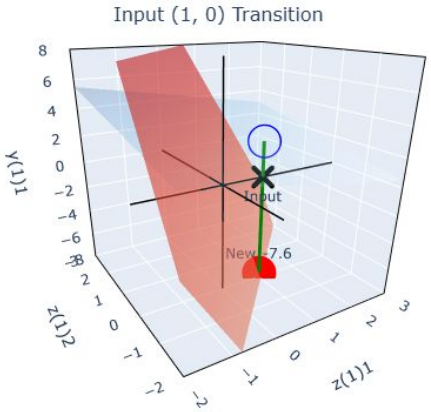
순전파

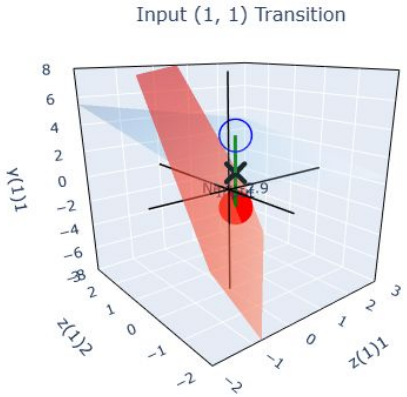
훈련을 잘 시킨 **w**파라미터로 순전파 시각화



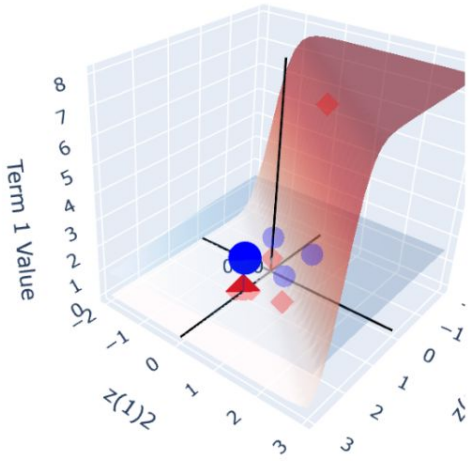


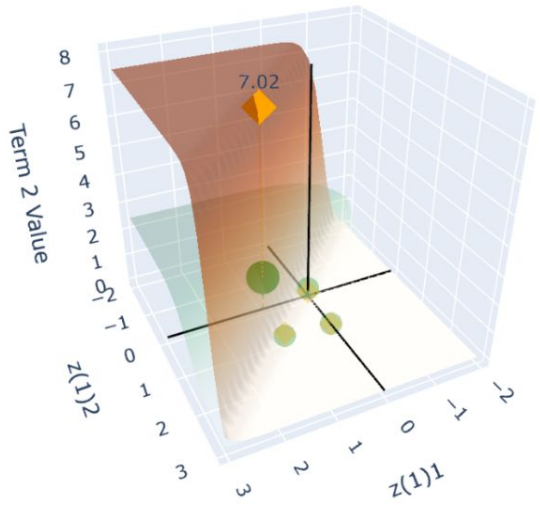


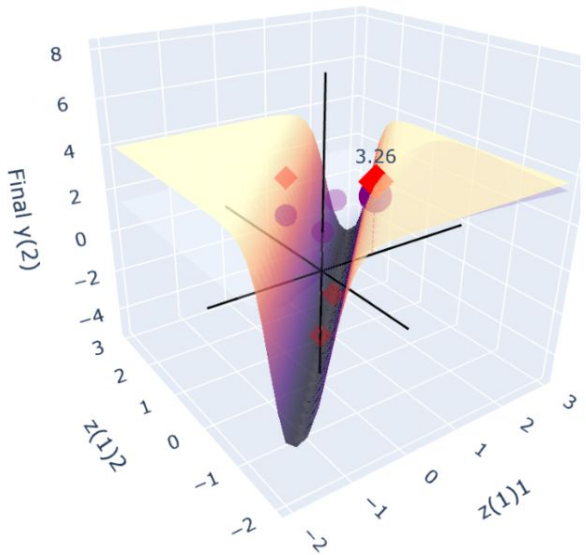




순전파


Term 1 Value
z(1)2
z(1)1


7.02
Term 2 Value
z(1)2
z(1)1


3.26
Final y(2)
z(1)2
z(1)1

코드로 이해

pytorch

PyTorch 예측 결과:

[[0.01603173]

[0.9808172]

[0.9807922]

[0.03328492]]

```python
import torch
import torch.nn as nn
import torch.optim as optim

X = torch.tensor([[0,0], [0,1], [1,0], [1,1]], dtype=torch.float32)
y = torch.tensor([[0],   [1],   [1],   [0]],   dtype=torch.float32)

class XORModel(nn.Module):
    def __init__(self):
        super().__init__()
        self.layer1 = nn.Linear(2, 2)

        # [수정됨] 활성화 함수: ReLU -> Sigmoid
        # 아까 수식에서 본 S자 곡선을 적용합니다.
        self.activation = nn.Sigmoid()

        self.layer2 = nn.Linear(2, 1)
        self.sigmoid = nn.Sigmoid()

    def forward(self, x):
        x = self.layer1(x)
        x = self.activation(x) # 여기서 부드럽게 휜 공간이 만들어집니다.
        x = self.layer2(x)
        return self.sigmoid(x)

model = XORModel()

criterion = nn.BCELoss()
optimizer = optim.Adam(model.parameters(), lr=0.01)

for epoch in range(2000):
    prediction = model(X)
    loss = criterion(prediction, y)
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()

print("PyTorch (Sigmoid) 예측 결과:\n", model(X).detach().numpy())
```

코드로 이해

Keras

Keras 예측 결과:

[[0.02846328]

[0.96662706]

[0.97150636]

[0.0256856]]

```python
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.optimizers import Adam

# 1. 데이터 준비
X = np.array([[0,0], [0,1], [1,0], [1,1]])
y = np.array([[0],   [1],   [1],   [0]])

model = Sequential()

# [중요] 은닉층 2개 (Sigmoid)
# 이론 설명대로 뉴런 2개를 유지합니다.
model.add(Dense(units=2, input_dim=2, activation='sigmoid'))

# 출력층 1개 (Sigmoid)
model.add(Dense(units=1, activation='sigmoid'))

optimizer = Adam(learning_rate=0.01)

model.compile(loss='binary_crossentropy', optimizer=optimizer, metrics=['accuracy'])

# 학습
model.fit(X, y, epochs=1000, verbose=0)

# 결과 출력 (소수점 3자리까지만 깔끔하게)
pred = model.predict(X)
print("Keras 예측 결과:\n", pred)
print(np.round(pred)) # 반올림해서 0과 1로 변환
```

QnA

질문 받겠습니다

감사합니다.

참고 링크 (출처)

개구리 사진 - https://velog.io/@tenyears/%EA%B6%81%EA%B8%88%ED%95%9C%EA%B2%83-Promise

신경망 사진 - Deep Convolutional Neural Networks: A Comprehensive Review[v1] | Preprints.org

게츠비 개구리 사진 - https://m.blog.naver.com/parkamsterdam/222527494434

and, or 게이트 사진 - http://www.ktword.co.kr/test/view/view.php?no=4557

nand 게이트 사진 - http://www.ktword.co.kr/test/view/view.php?no=4560

xor 3차원시각화 사진 - 딥러닝 수업자료 (고선우교수님) - 03_04 XOR 문제로 Deep Neural Network 작동원리 이해하기.ipynb

xor 게이트 사진 - https://velog.io/@citizenyves/Perceptron-%EB%8B%A4%EC%B8%B5-%ED%8D%BC%EC%85%89%ED%8A%B8%EB%A1%A0-XOR-%EA%B2%8C%EC%9D%B4%ED%8A%B8python-%EA%B5%AC%ED%98%84

비선형 근사화 사진 - 딥러닝 수업자료 (고선우교수님) - 03_01 Perceptron과 Fully Connected Neural Network.ipynb

xor 게이트 진리표 - https://m.blog.naver.com/cni1577/221619153912

할머니 사진 - https://kr.lovepik.com/image-507752562/an-old-man-who-has-questions-about-reading.html

기타 그래프사진, 수식 - 이연승