AI and Deep Learning

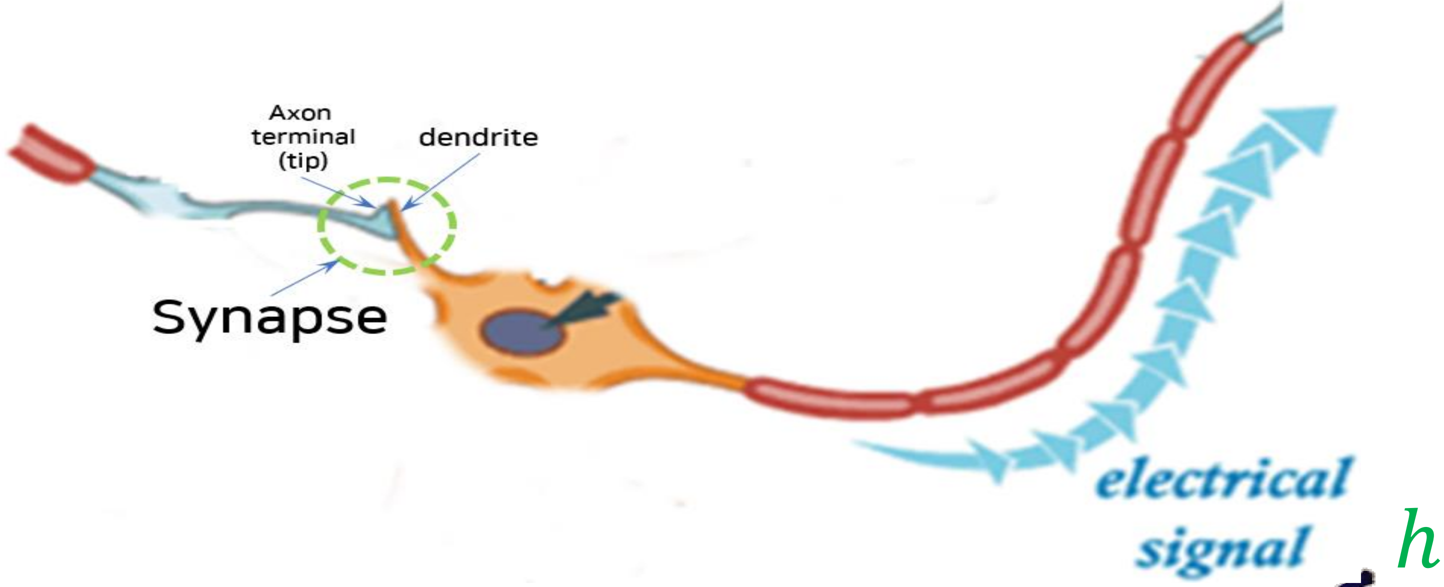
Logistic Regression & Classification

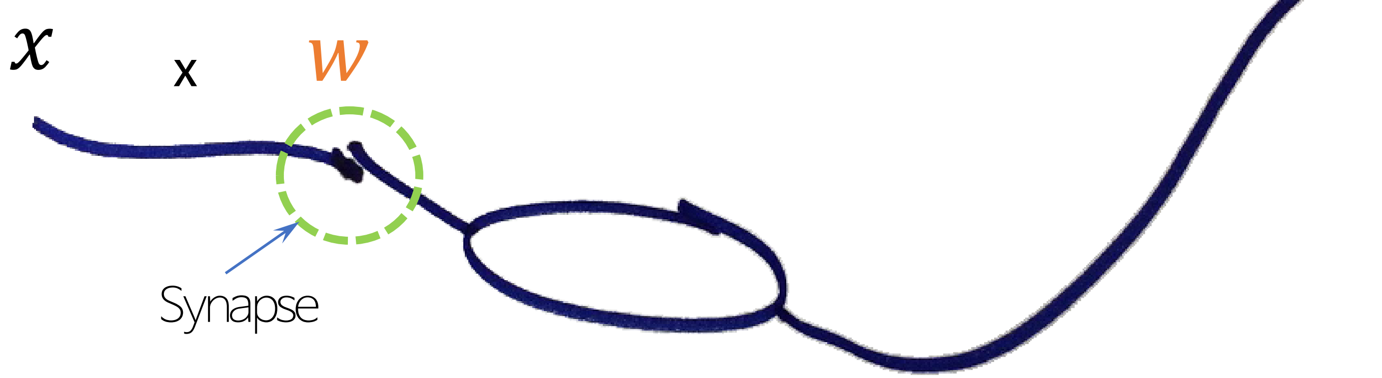
Jeju National University

Yungcheol Byun

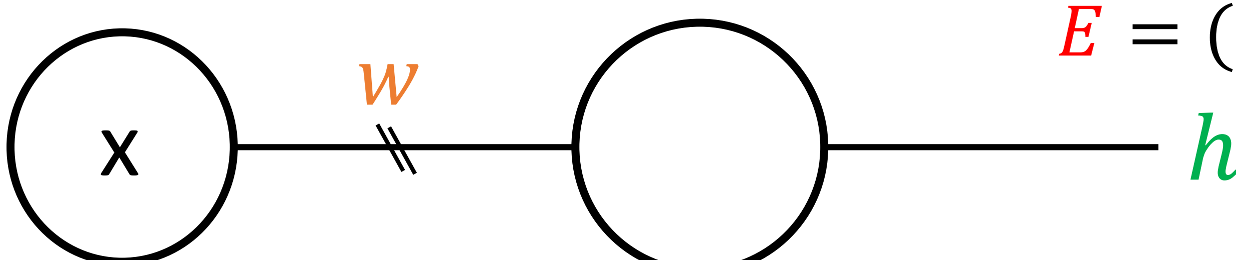
Agenda

- Logistic regression and classification
- New loss/cost function
- Decision boundary
- Implementation using TensorFlow
- Multiple-class problem


Axon terminal (tip)
dendrite
Synapse
electrical signal
h


x
x
w
Synapse

$$E = (h - y)^2$$


X
w
h

Logistic Regression

The shape of regression is not linear but logistic.

What does that mean?

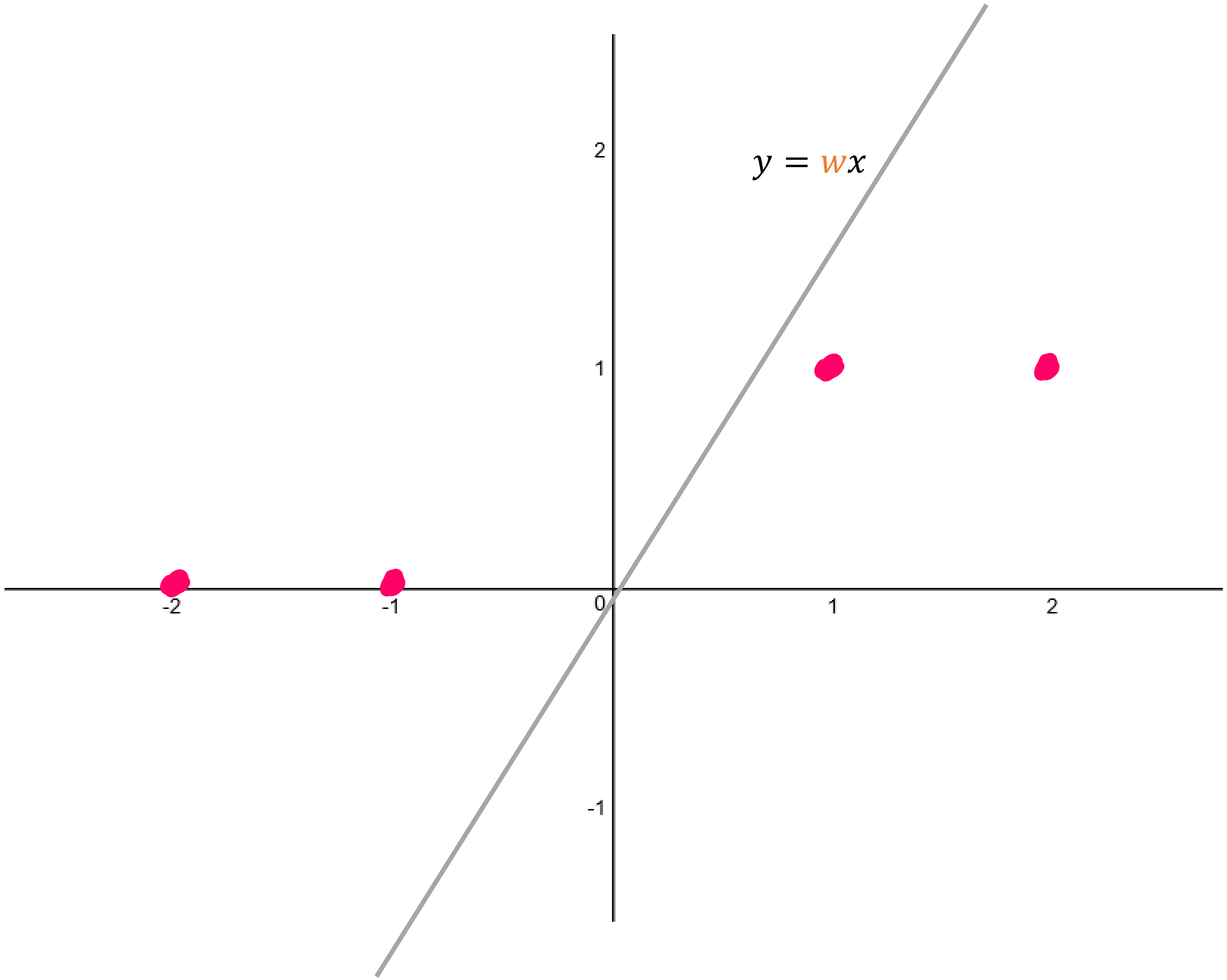
desmos

Draw $(-2, 0), (-1, 0), (1, 1), (2, 1)$.

$$y = wx$$

$$y = sigmoid(wx)$$

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

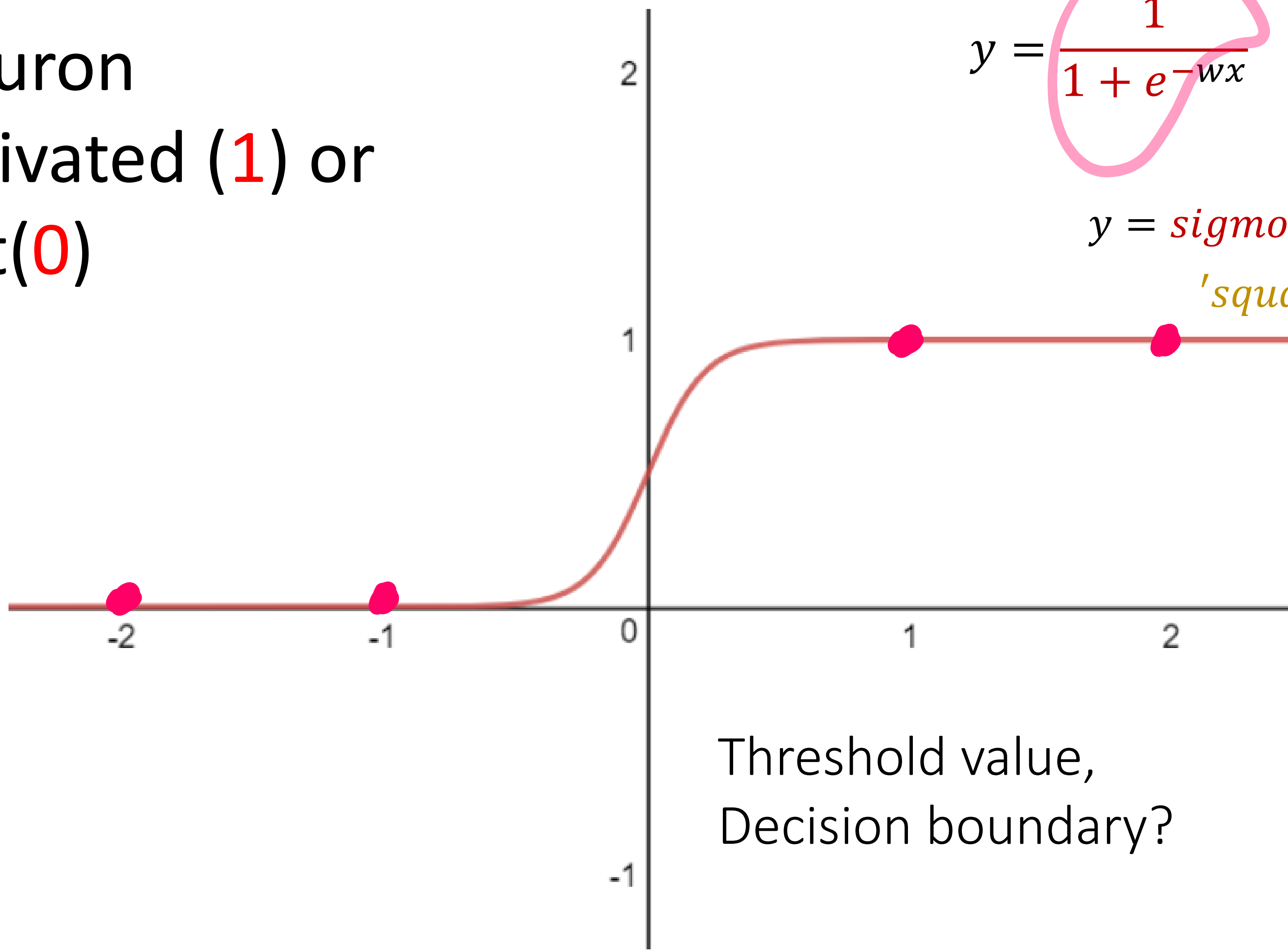

$y = wx$
2
1
-1
0
-2
-1
1
2

Neuron activated (1) or not(0)

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

$$y = sigmoid(wx)$$

$'squash'$



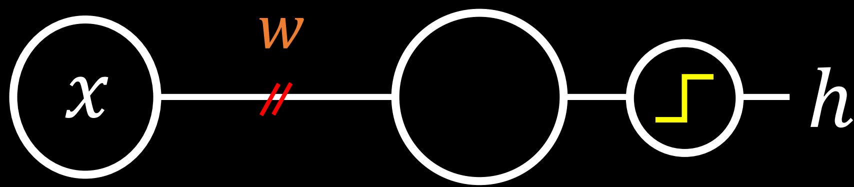
Threshold value,
Decision boundary?

Revisited

Real operation of a neuron

- signal ON if the weighted sum is greater than T
- otherwise signal OFF

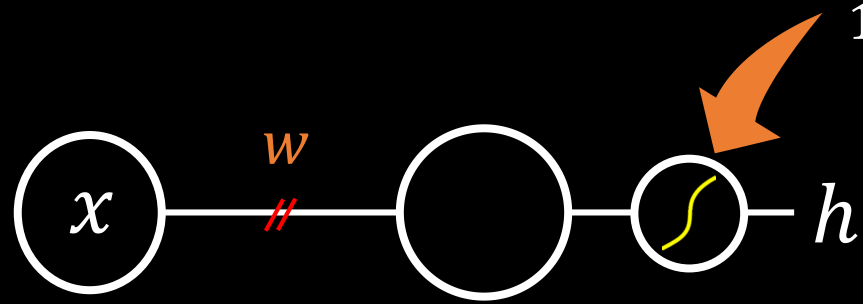
Revisited



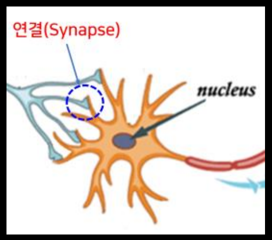
Thresholding

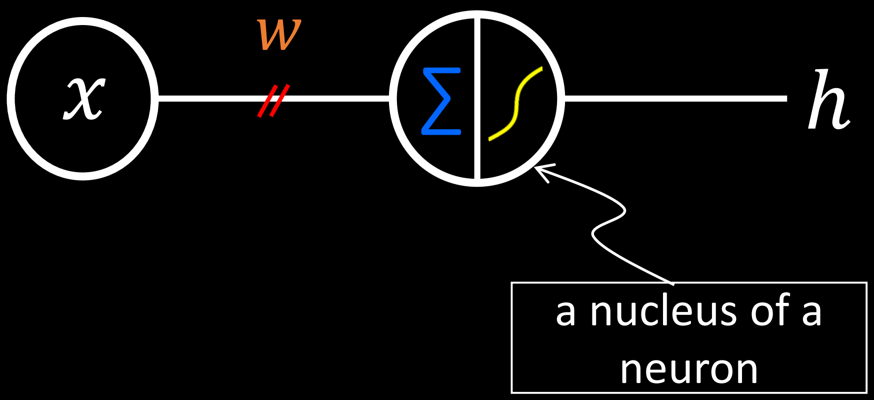
Revisited

sigmoid $\frac{1}{1+e^{-wx}}$



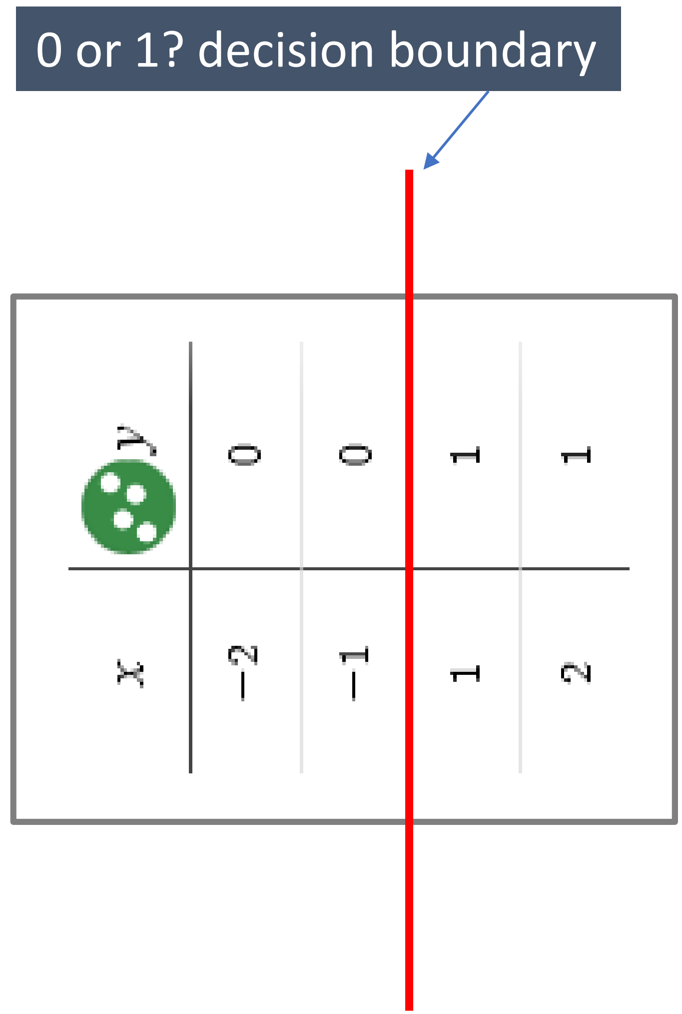
Thresholding





- Activated(1) or not(0) according to the input x
- Let’s guess the decision boundary to decide.

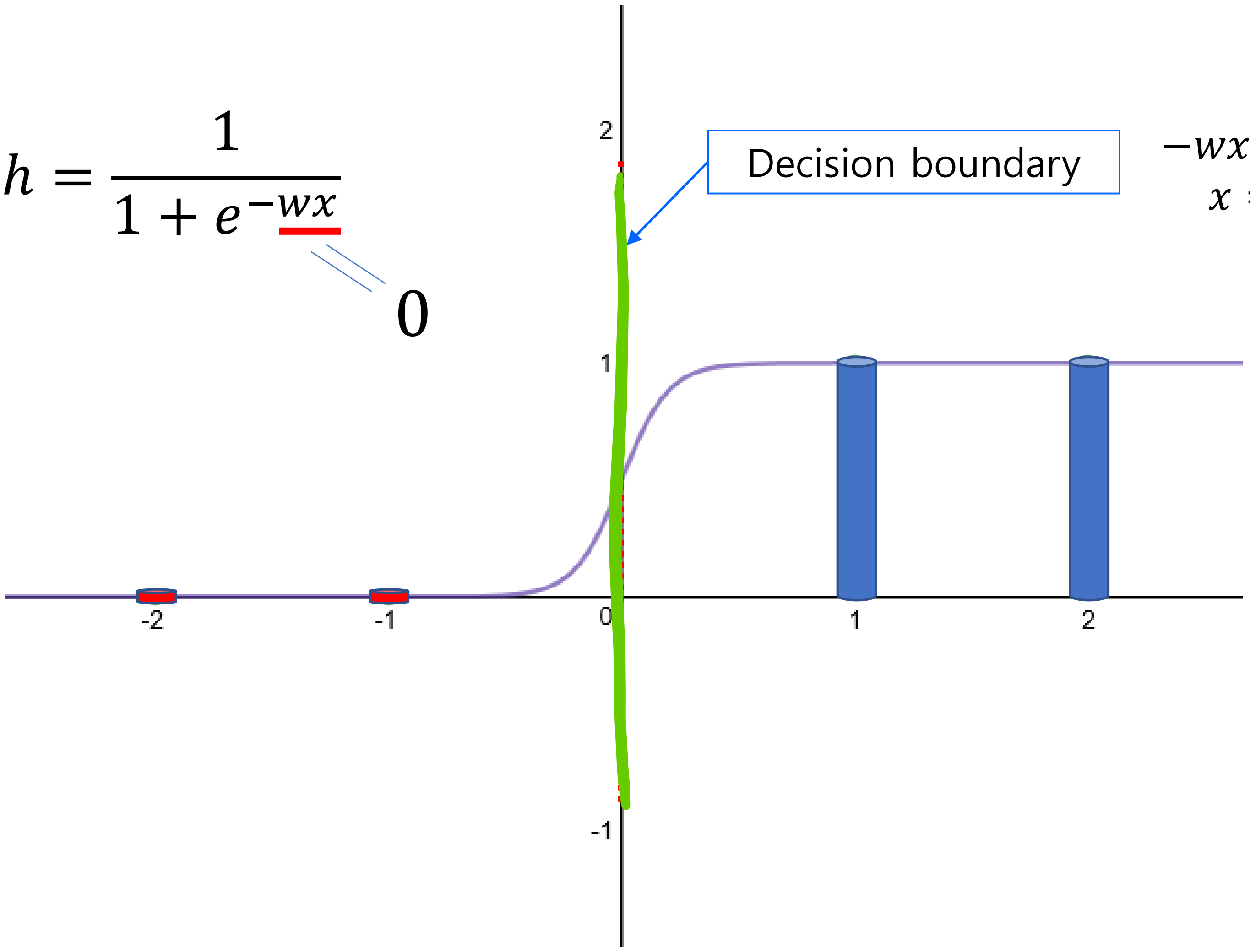
x	y
-2	0
-1	0
1	1
2	1



Decision boundary

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

0

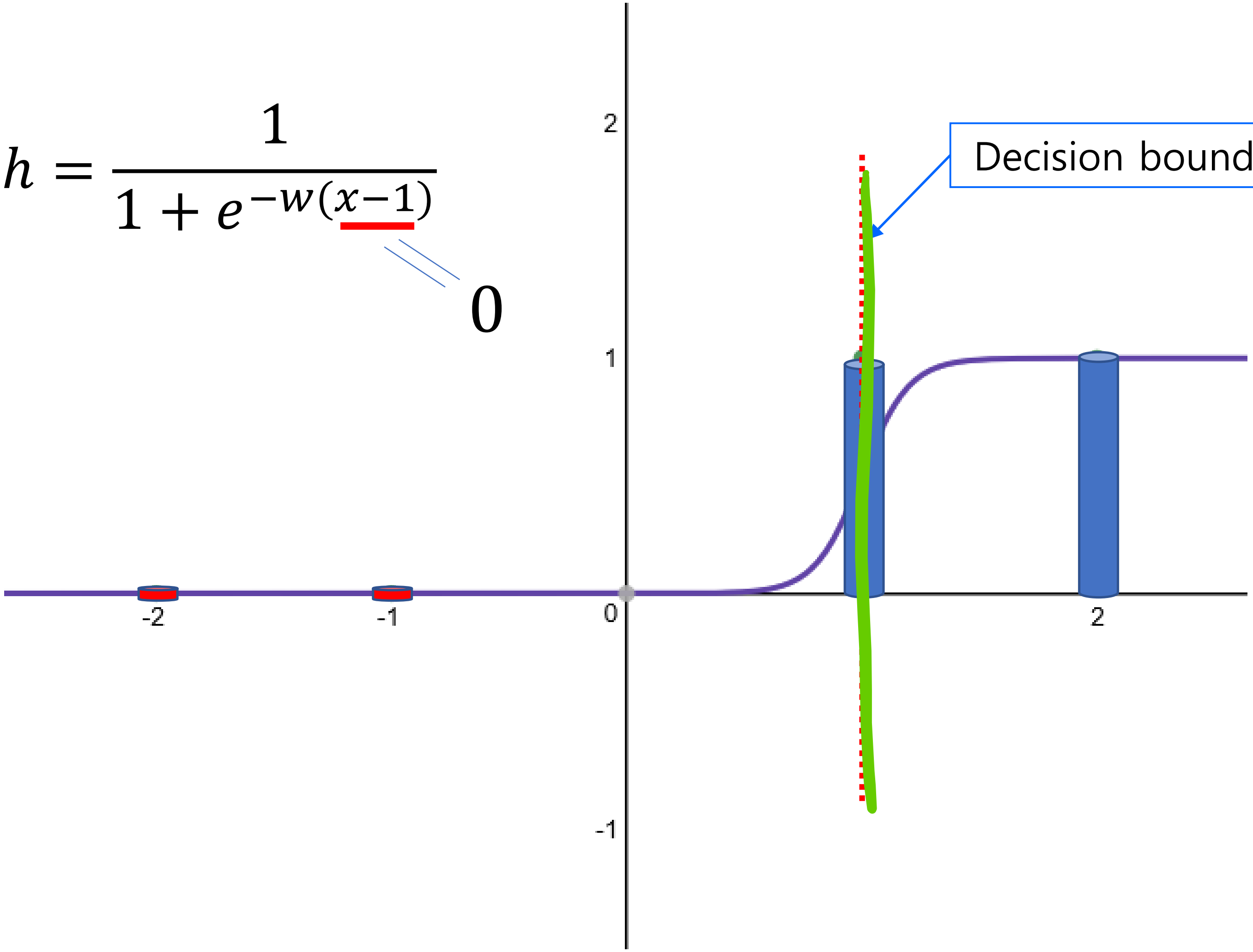


$$-wx = 0$$
$$x = 0$$

Decision boundary

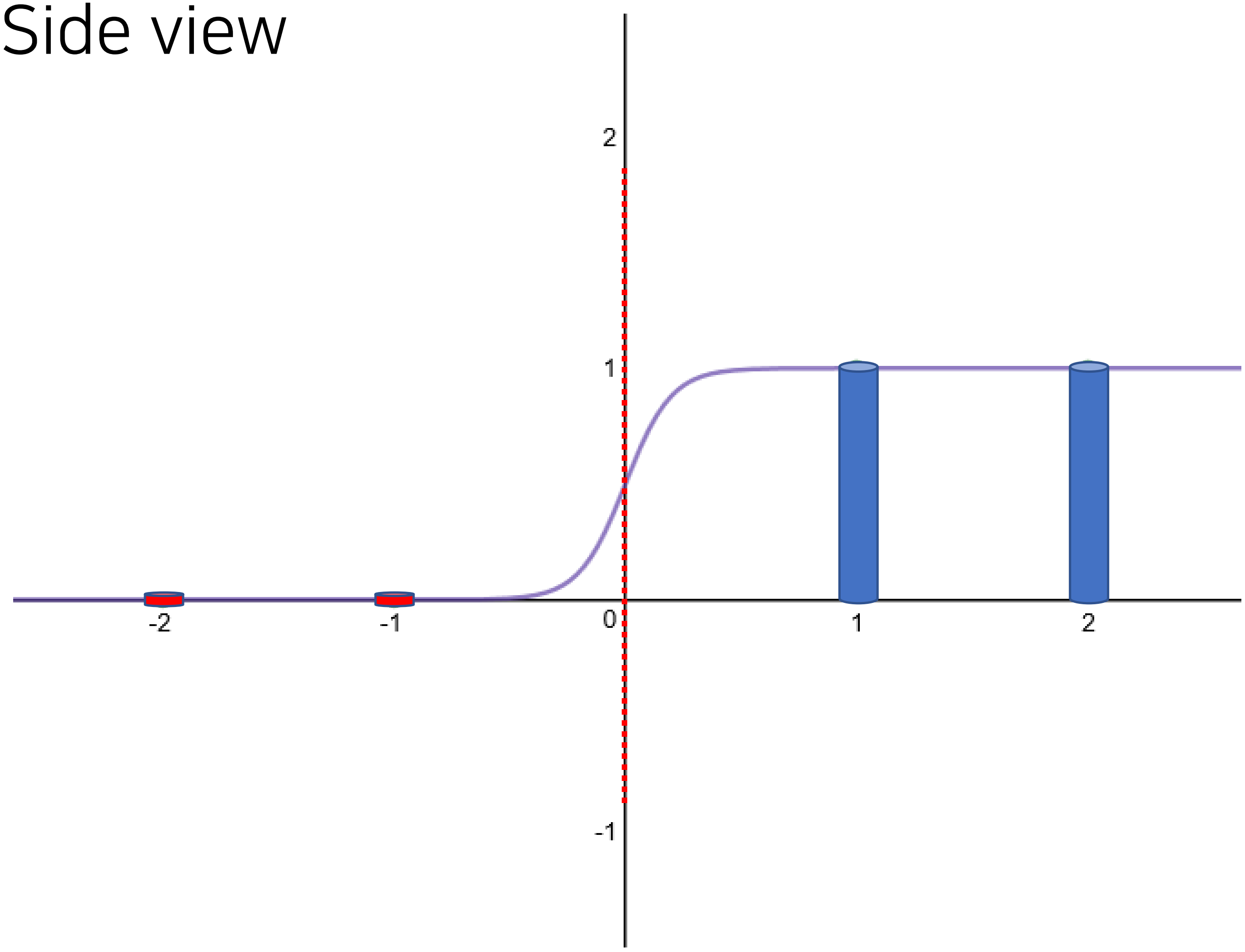
$$h = \frac{1}{1 + e^{-w(x-1)}}$$

0

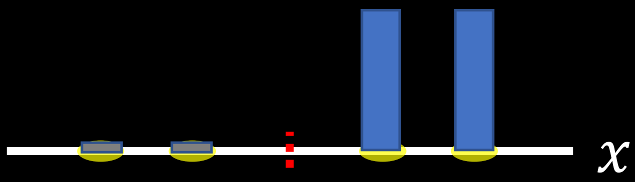


$x = 1$

Side view



Decision Boundary



{Side view}

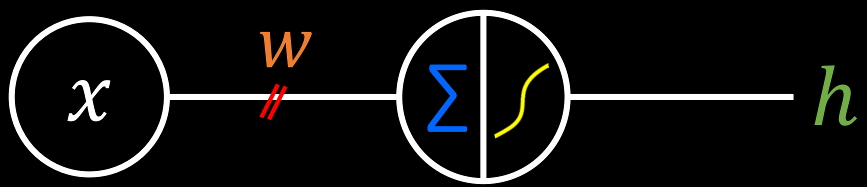
{View from above}

Classification

- Pass(1) or Fail(0)
- Spam(1) or Ham(0)
- Scam(fraud, 1) or not(0)
- Safe(1) or Dangerous(0)
- Intrusion/virus(1) or not(0)
- Cancer(1) or not(0)
- Binary classification -> Multiple classification

Guess the decision boundary from the below figure.

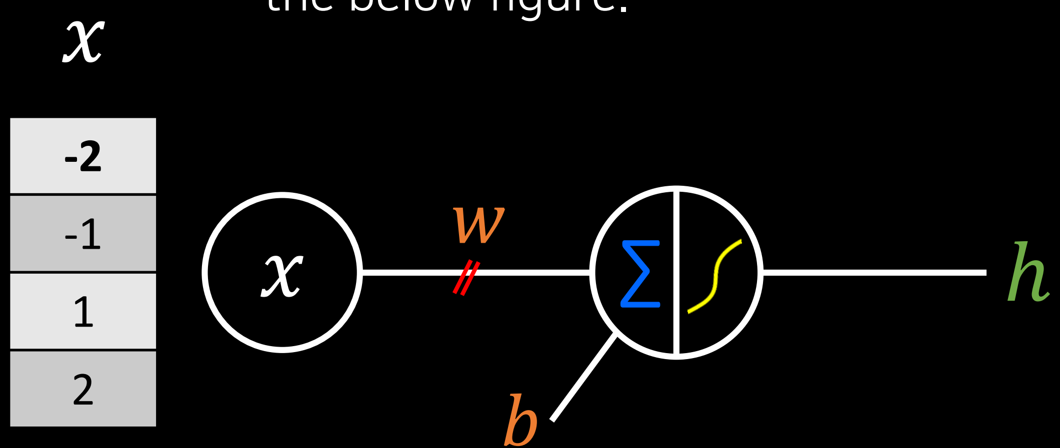
x
-2
-1
1
2



y
0
0
1
1

$$h = \begin{cases} 1 & if\ wx \geq 0 \\ 0 & otherwise \end{cases}$$

Guess the decision boundary from the below figure.



$$h = \begin{cases} 1 & if\ wx + b \geq 0 \\ 0 & otherwise \end{cases}$$

Hypothesis

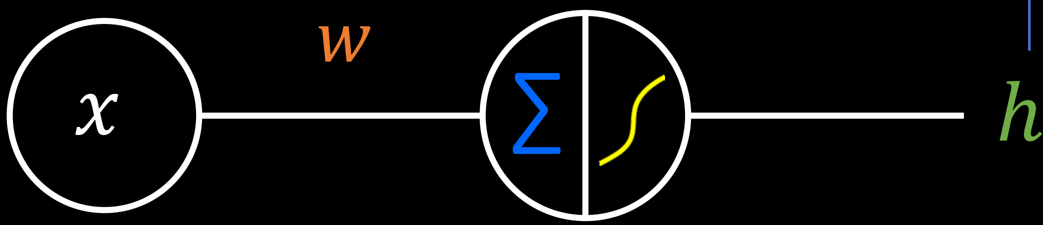
- What is hypothesis? Neuron's output
- Find decision boundary from the equation.

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

$$h = \frac{1}{1 + e^{-(wx+b)}}$$

Cost/Error Function

$$E = (h - y)^2$$



Does MSE work?

Draw $(-2, 0), (-1, 0), (1, 1), (2, 1)$.

$$h = wx$$

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

Draw $(1, 1)$ only.

$$E = \left(\frac{1}{1 + e^{-w \cdot 1}} - 1 \right)^2$$

$$(w, E)$$

desmos

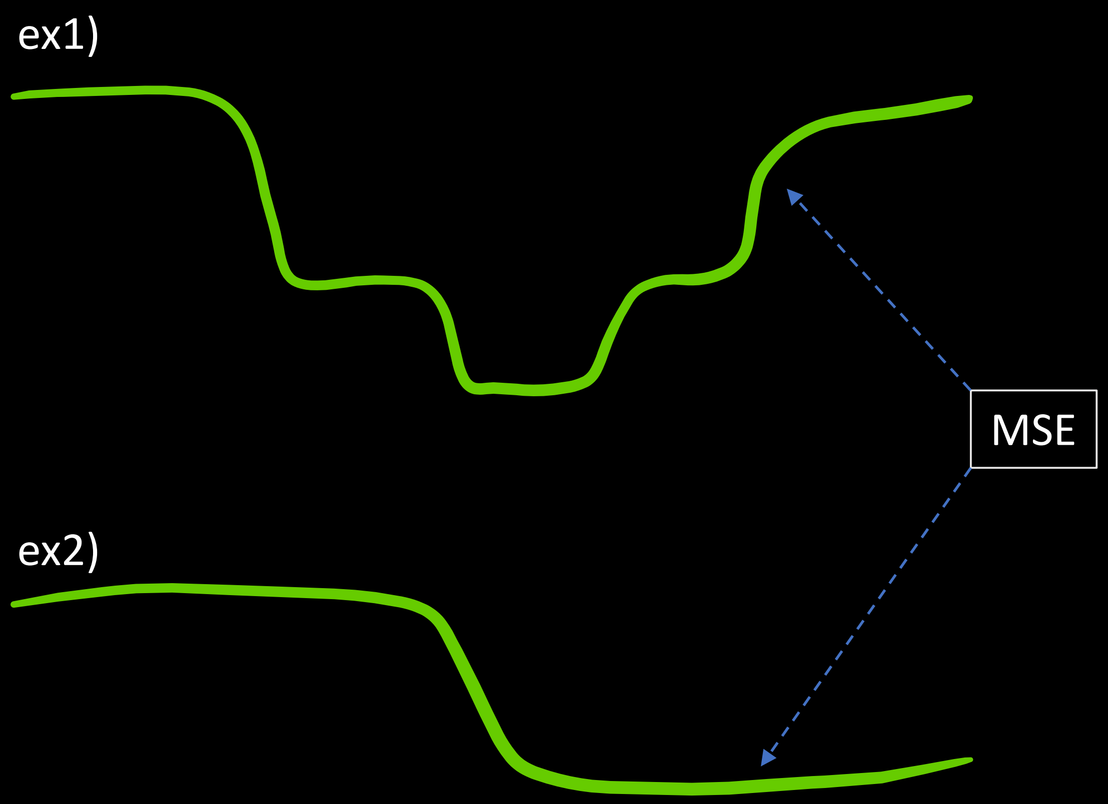
Draw two points: $(-1,0),(1,1),(-3,0),(3,1)$.

$$E=\left(\frac{1}{1+e^{-w(-1)}}-0\right)^2+\left(\frac{1}{1+e^{-w(1)}}-1\right)^2+\left(\frac{1}{1+e^{-w(-3)}}-0\right)^2+\left(\frac{1}{1+e^{-w(3)}}-1\right)^2$$

Add bias b.

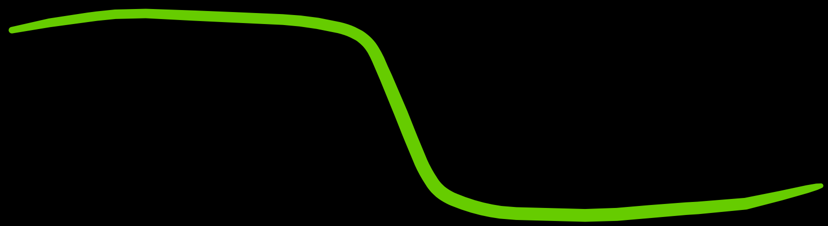
$$\left(w,\frac{E}{2}\right)$$

Cost/Error Function when we use MSE.



What problem in the cost/loss function?

No gradient decent in some cases



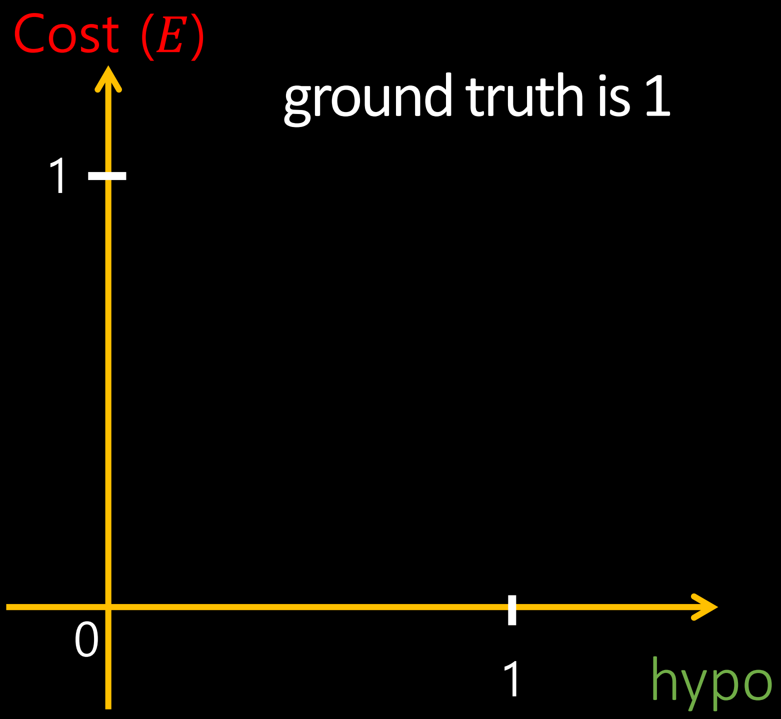
New loss function

- Check the output of a neuron (hypothesis)
- If equal to the ground truth(good), then error $\leftarrow$ 0.
- If opposite of the ground truth(bad), then error $\leftarrow \infty$

New loss function

When ground truth is 1

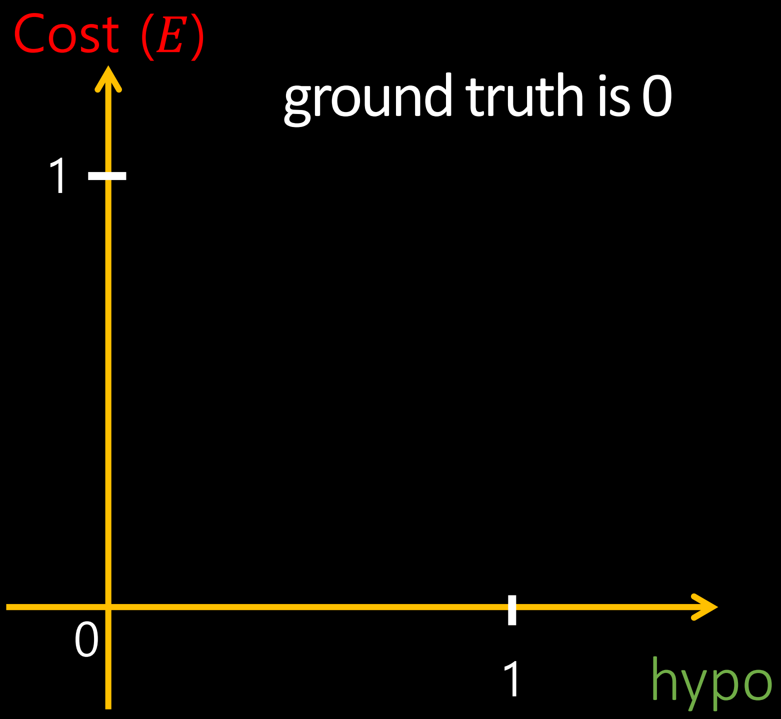
- if hypo is equal to 1, then error = 0
- if hypo is equal to 0 then error = ∞



New loss function

When ground truth is 0

- if hypo is equal to 0, then error = 0
- if hypo is equal to 1 then error = ∞



$$E = -\log(h)$$

$$E = -\log(1 - h)$$

$$E = -\log\left(\frac{1}{1 + e^{-wx}}\right)$$

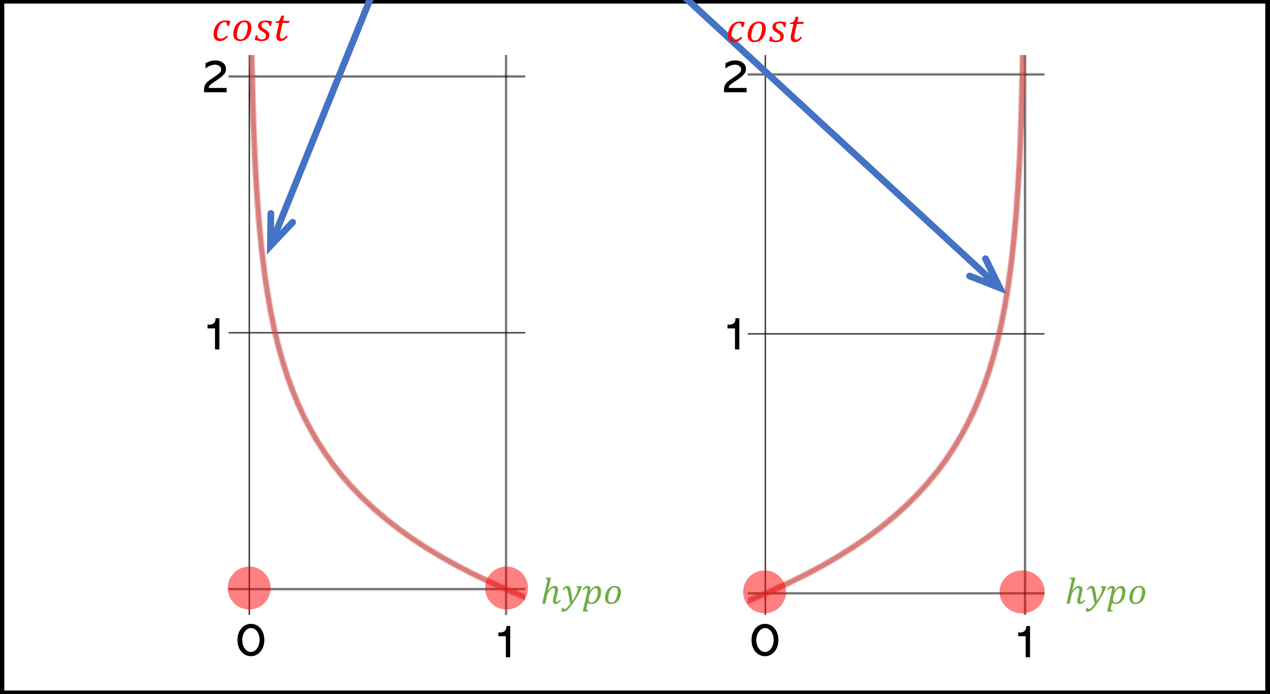
$$E = -\log\left(1 - \frac{1}{1 + e^{-wx}}\right)$$

New loss function

Prediction by a neuron

Correct answer

$$E = \begin{cases} -\log(h) & : y = 1 \\ -\log(1-h) & : y = 0 \end{cases}$$



New loss function

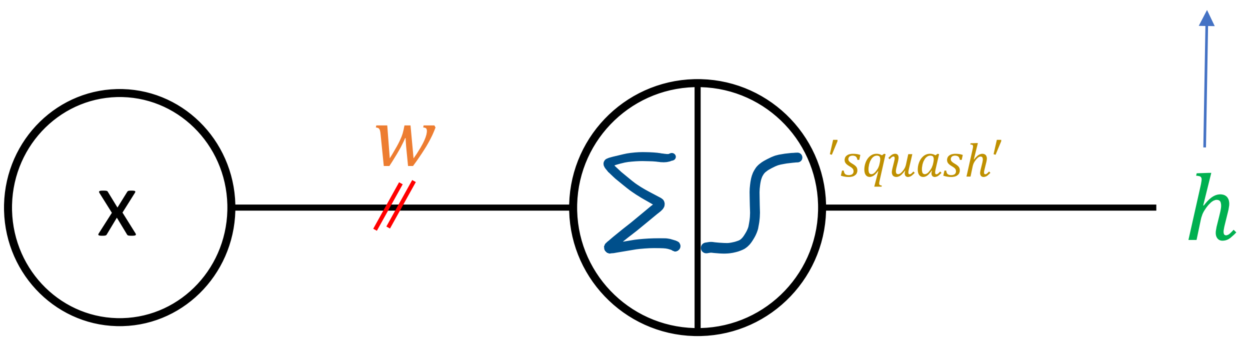
$$E = \begin{cases} -\log(wx) & : y = 1 \\ -\log(1 - wx) & : y = 0 \end{cases}$$

$$E = -y\log(wx) - (1 - \mathrm{y})\log(1 - wx)$$

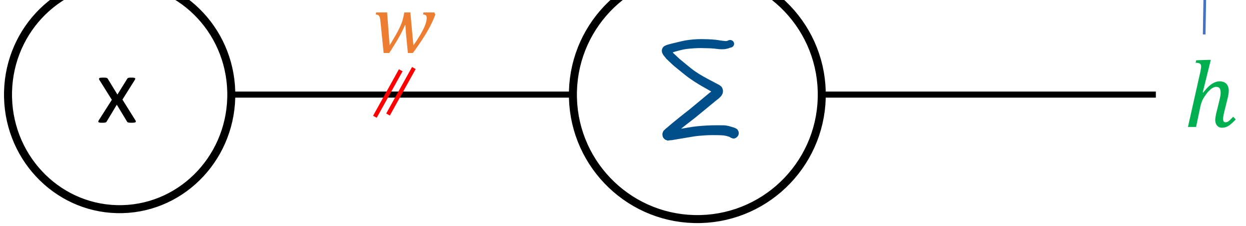
$$E = -(y\log(wx) + (1 - \mathrm{y})\log(1 - wx))$$

$$w = w - \alpha \cdot \frac{\partial E}{\partial w}$$

$$E = -(y\log(h) + (1-y)\log(1-h))$$



$$E = (h - y)^2$$



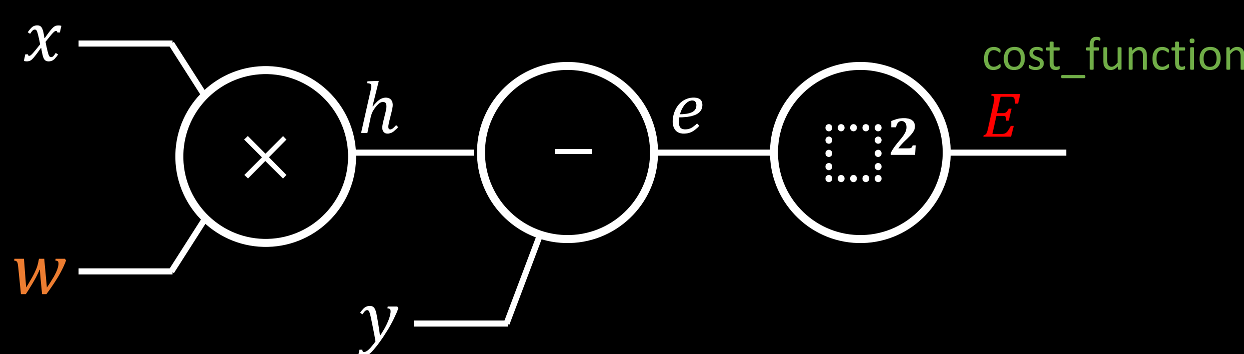
Computational graph

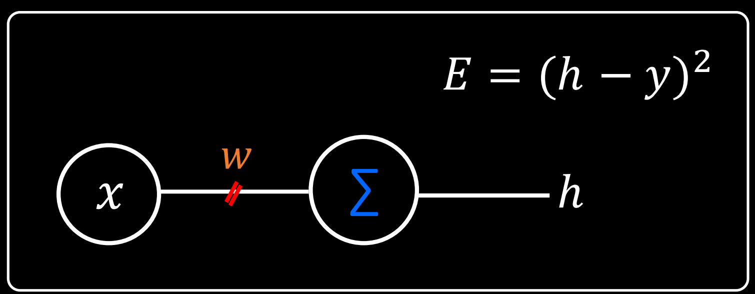
for the new cost function

Computational Graph

$$E = (wx - y)^2$$

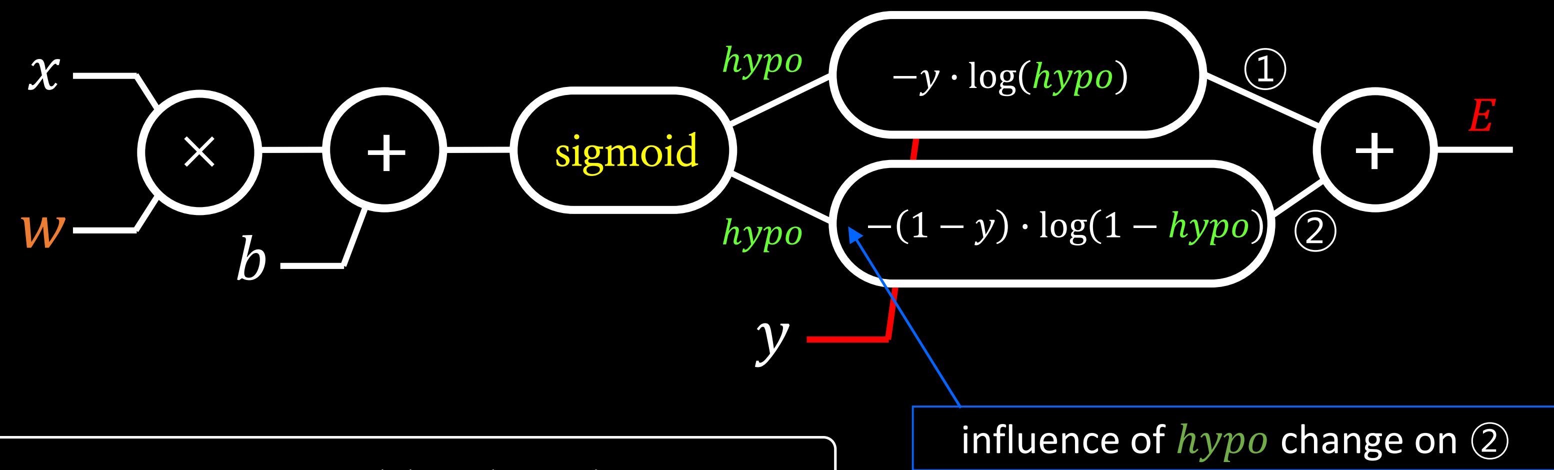
```
hypo = w * x
cost_function(E) = (hypo - y) ** 2)
```

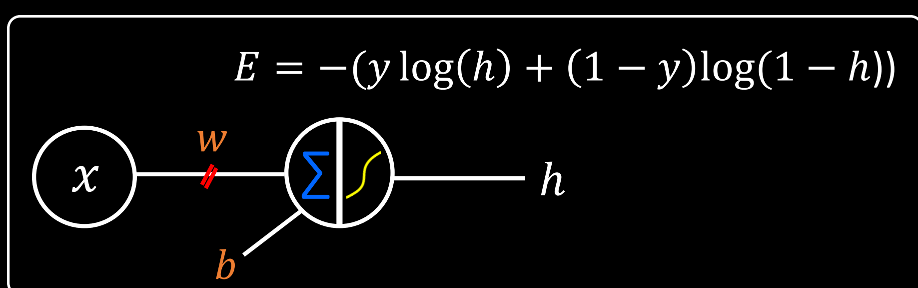




Computational Graph

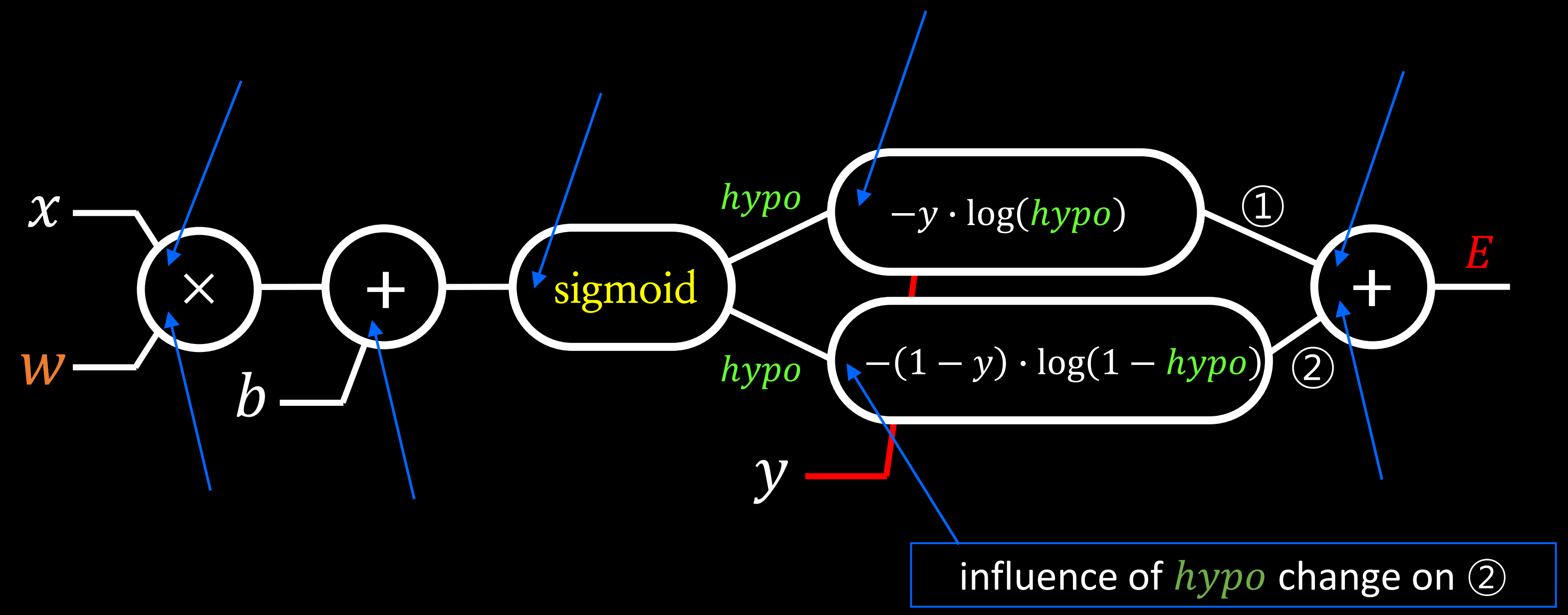
$$E = \overbrace{-y \cdot \log(hypo)}^{①} \overbrace{- (1-y) \cdot \log(1-hypo)}^{②}$$





$$\frac{\partial ②}{\partial h}$$

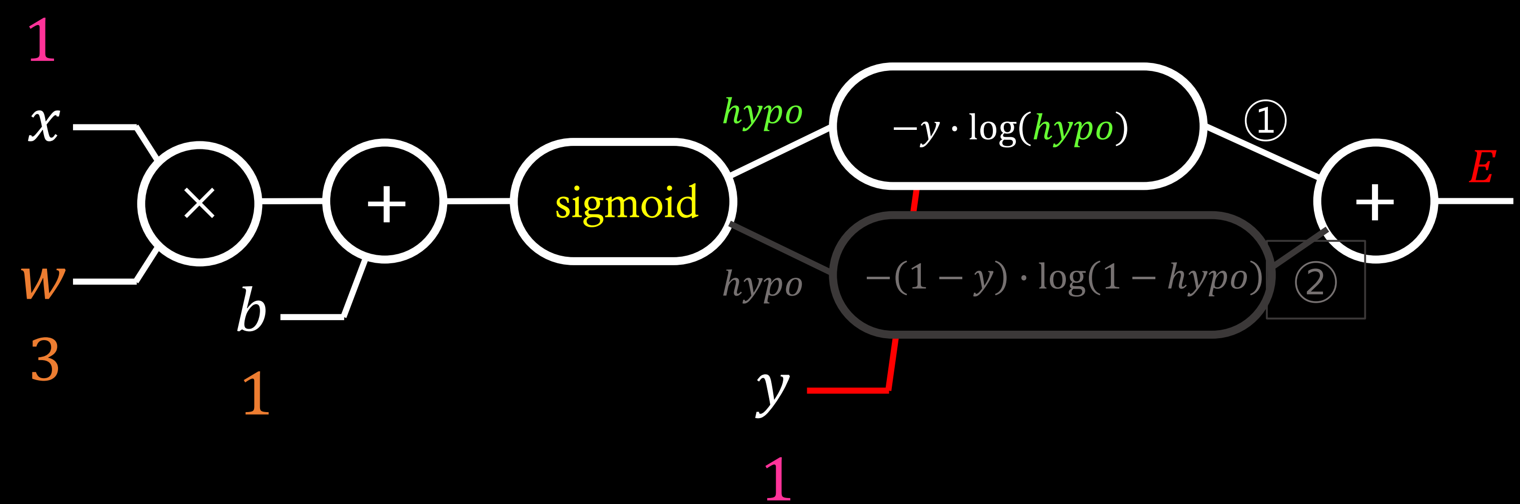
Local Gradients



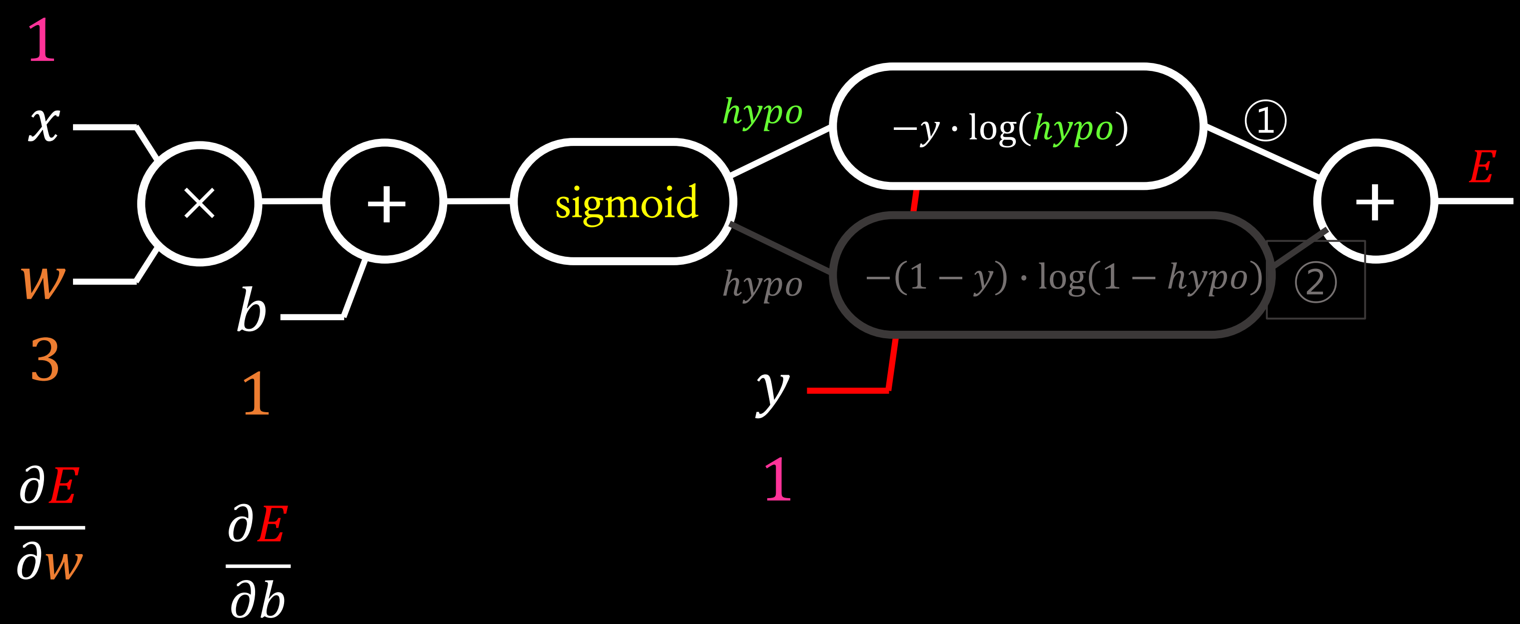
$$\frac{\partial ②}{\partial h}$$

Forward propagation

$$(x, y) \longrightarrow (1, 1)$$



Back-propagation



$$\boxed{w = w - \alpha \cdot \frac{\partial E}{\partial w}}$$

$$\boxed{b = b - \alpha \cdot \frac{\partial E}{\partial b}}$$

Parameters(w, b) tuning for what?

for better decision boundary

Lab 11.py

Classification of an input as 1 or 0

$$cost = -(y\log(H(X)) + (1-y)\log(1-H(X)))$$

```
x_data = [-2., -1, 1, 2]
y_data = [0., 0, 1, 1]
```

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

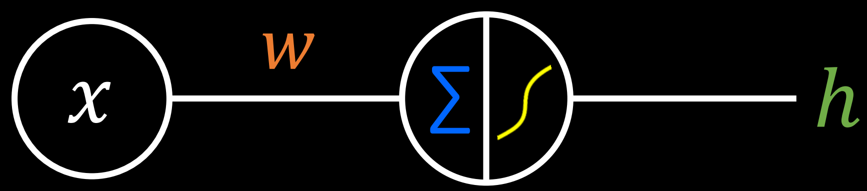
```
#----- learning
cost = -tf.reduce_mean(y_data * tf.log(hypo) +
        tf.subtract(1., y_data) * tf.log(tf.subtract(1., hypo)))

train = tf.train.GradientDescentOptimizer(learning_rate=0.01).minimize(cost)

sess = tf.Session()
sess.run(tf.global_variables_initializer())

for step in range(5001):
    sess.run(train)
```

```
#----- testing(classification)
predicted = tf.cast(hypo > 0.5, dtype=tf.float32)
p = sess.run(predicted)
print("Predicted: ", p)
```

Lab 12.py
With a bias

1-Input Neuron

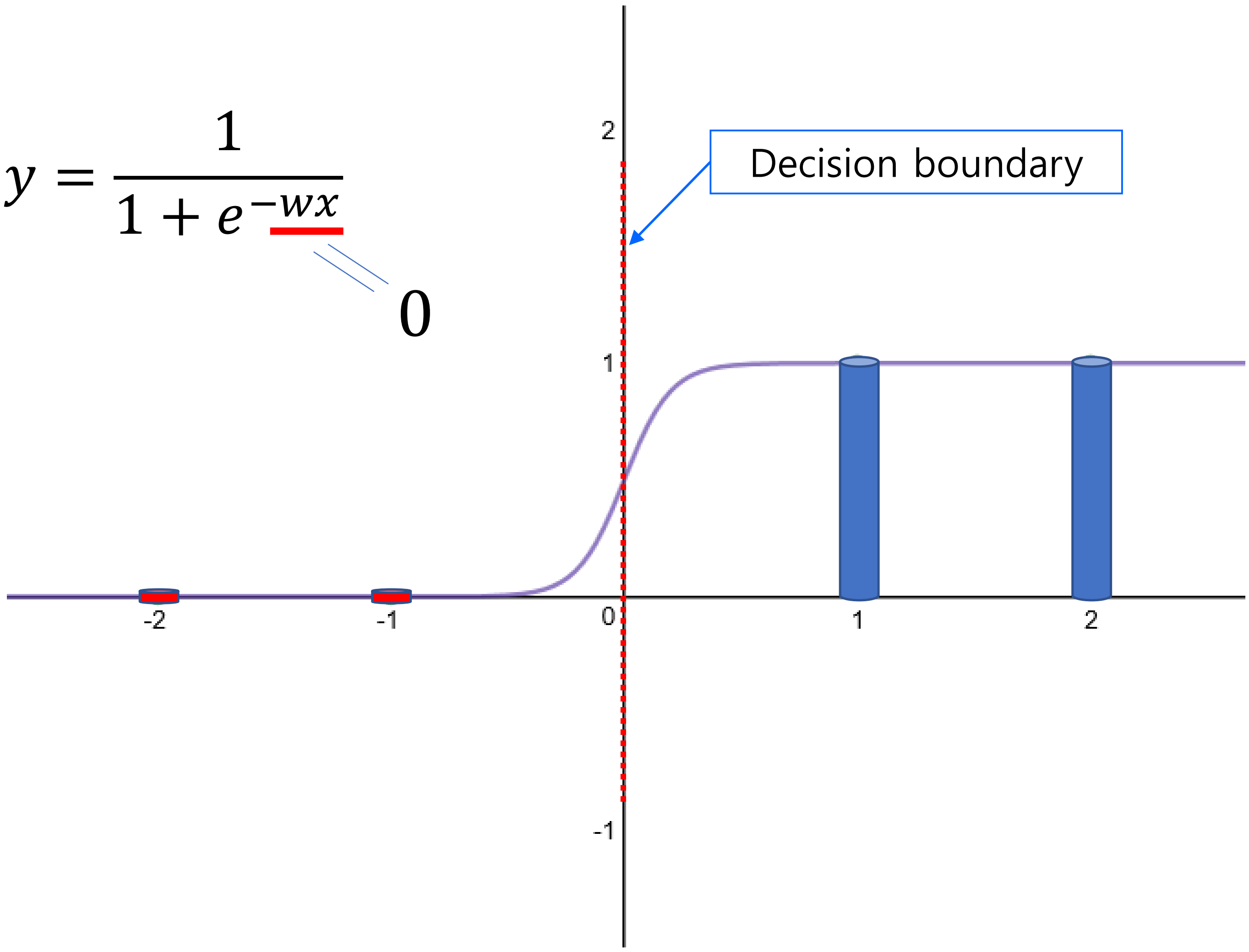
Guess a decision boundary.



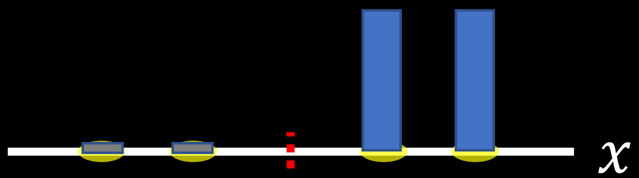
$$h = \frac{1}{1 + e^{-(wx)}}$$

1-Input Neuron

$$y = \frac{1}{1 + e^{\underbrace{-wx}_{=\,0}}}$$



1-Input Neuron

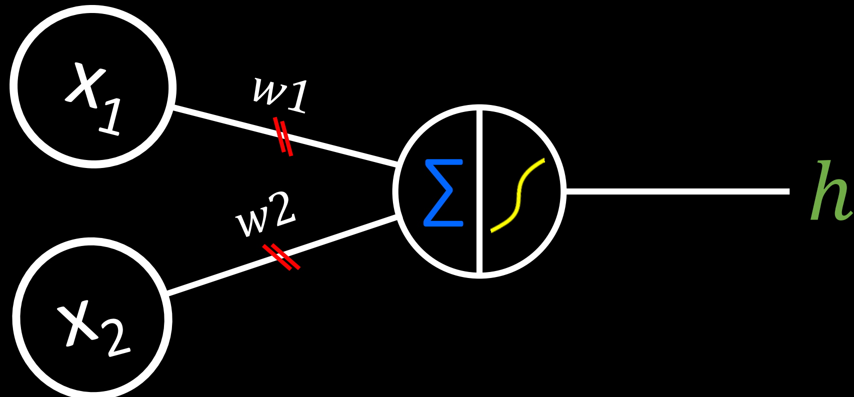


{Side view}

{View from above}

2-Input Neuron

Guess a decision boundary.



$$h = \frac{1}{1 + e^{-(w_1 x_1 + w_2 x_2)}}$$

2-Input Neuron

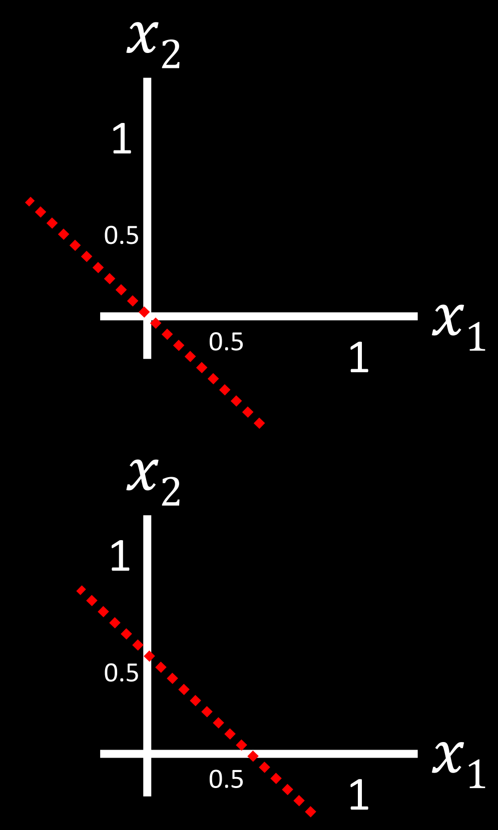
- Find decision boundary.

$$w_1 x_1 + w_2 x_2 = 0$$

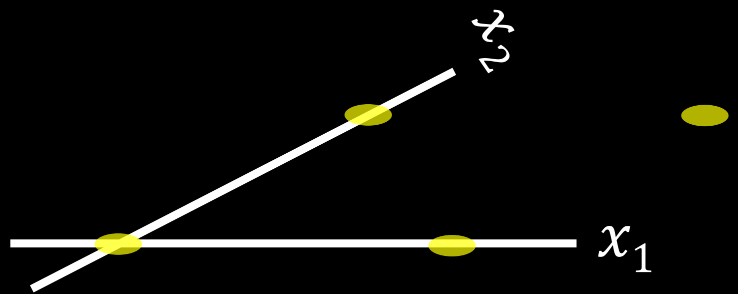
$$x_1 + x_2 = 0$$

$$x_1 + x_2 = 0.5$$

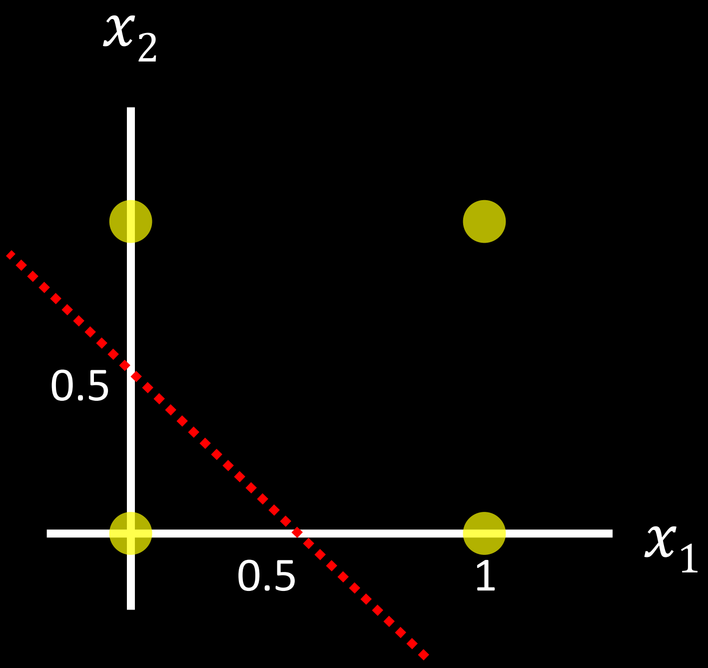
View from above



2-Input Neuron

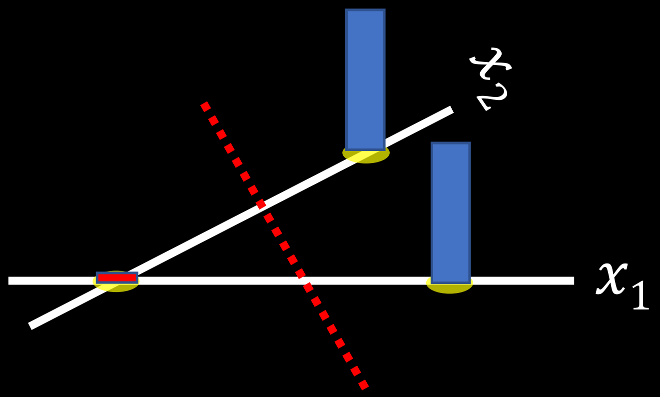


Side view

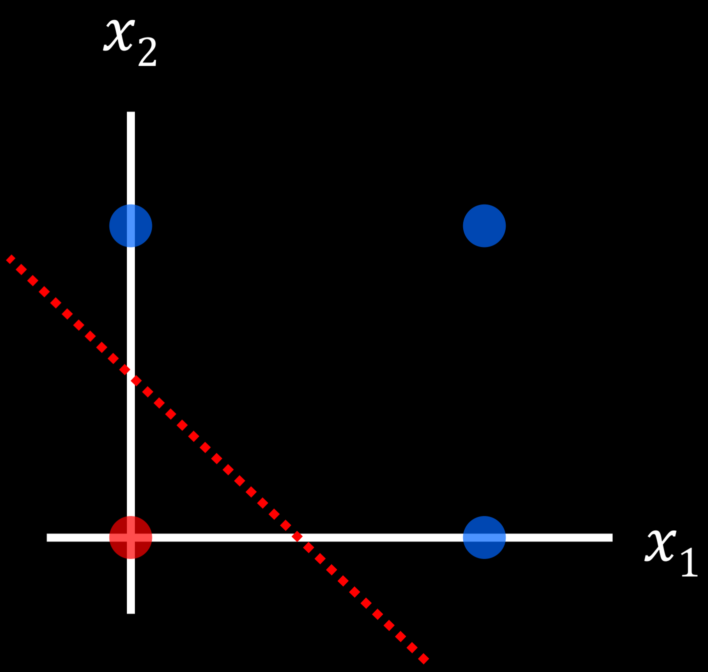


View from above

2-Input Neuron



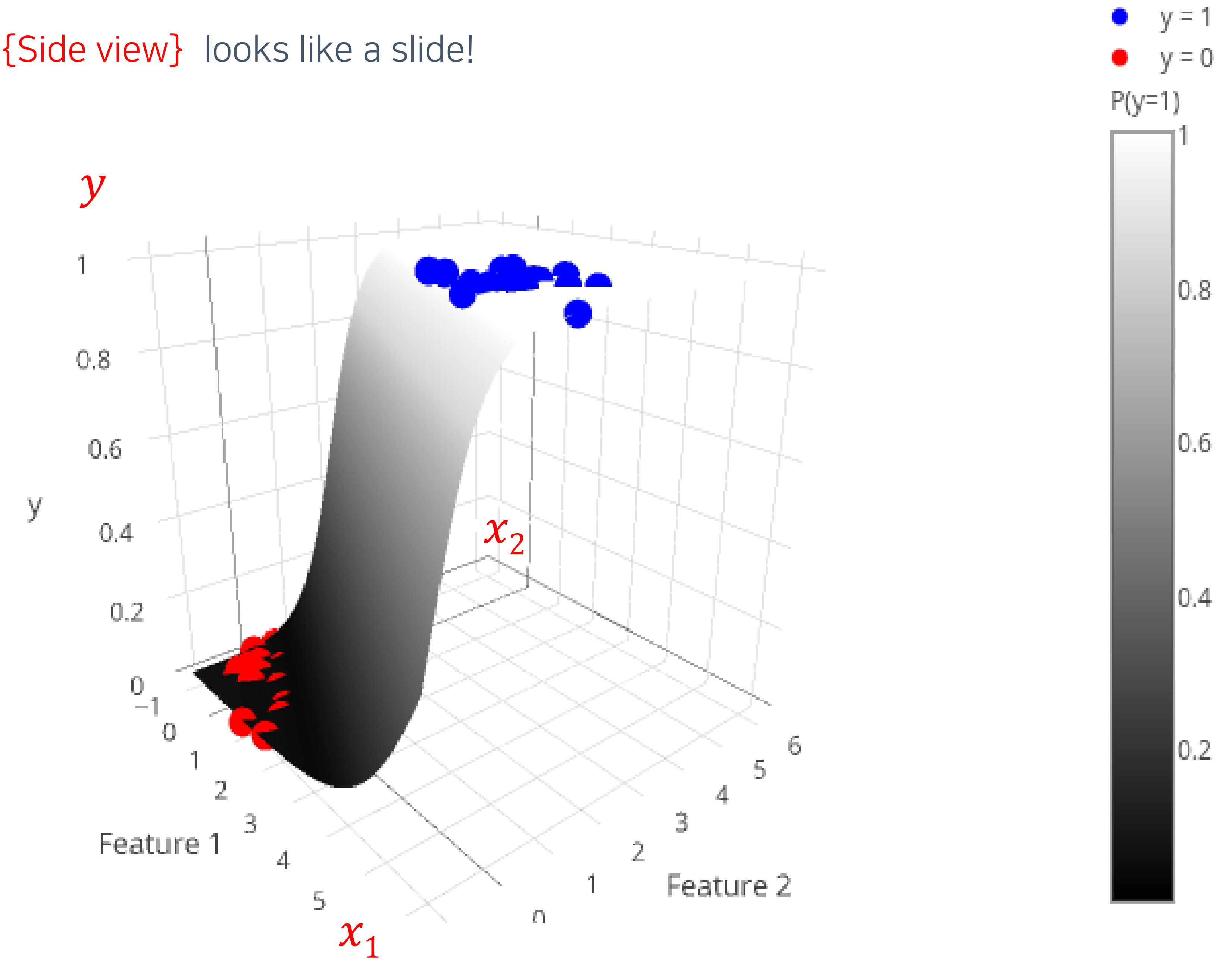
{Side view}



{View from above}

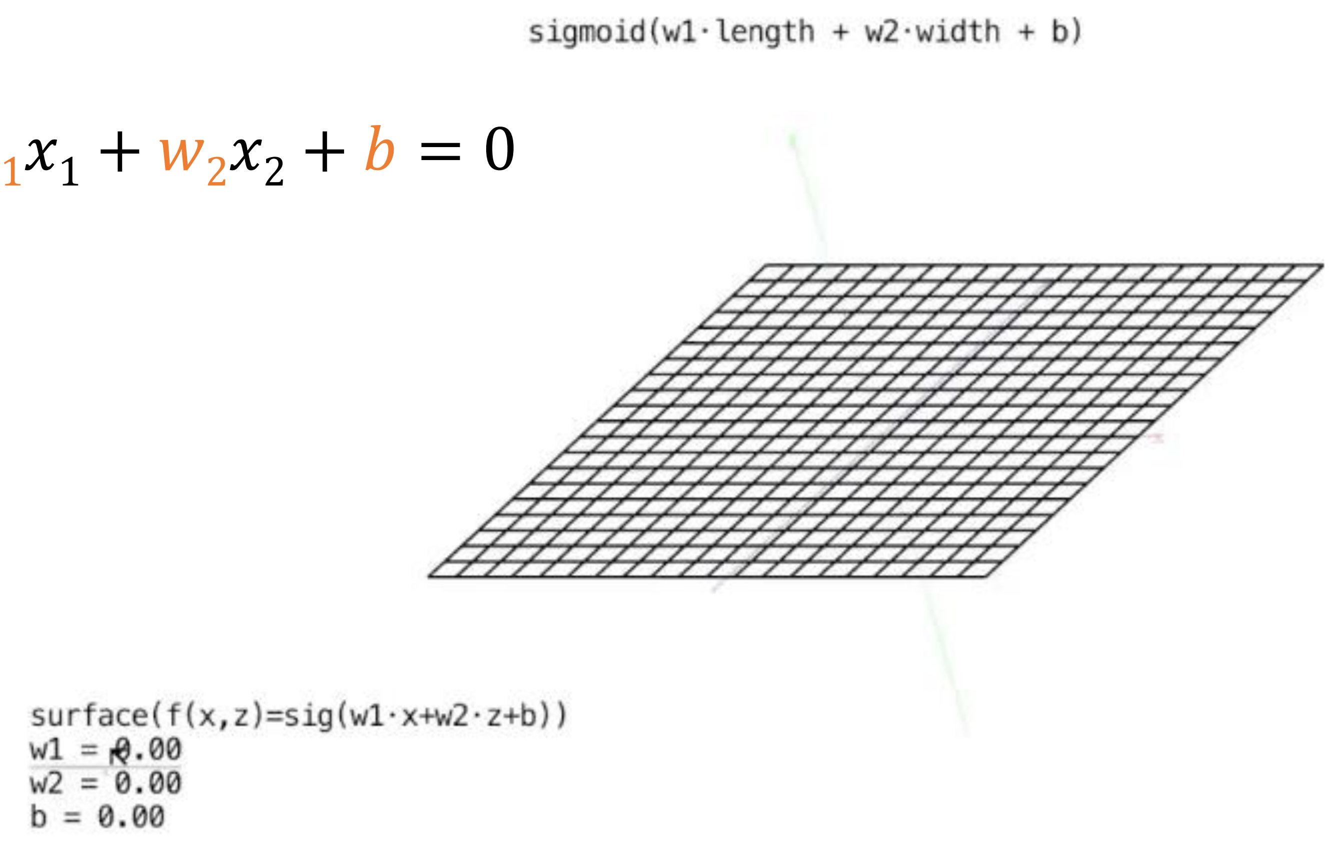
Logistic Regression: 2 Features (Inputs)

{Side view} looks like a slide!



2-Input Neuron

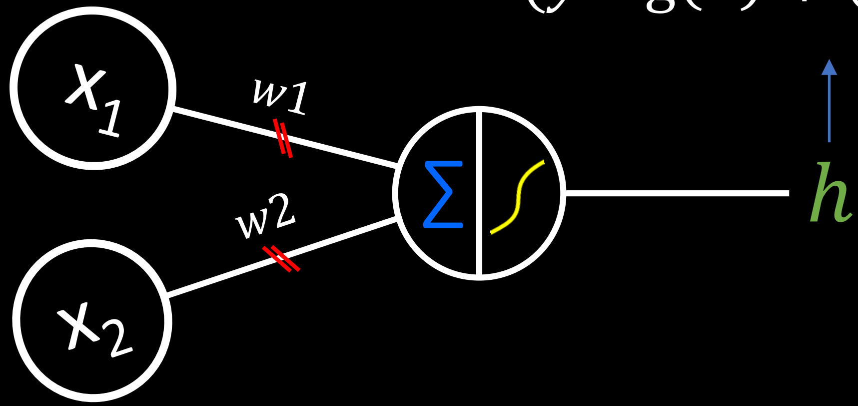
$$w_1 x_1 + w_2 x_2 + b = 0$$



Lab 13.py

Implementation of OR gate with a neuron(a decision boundary)

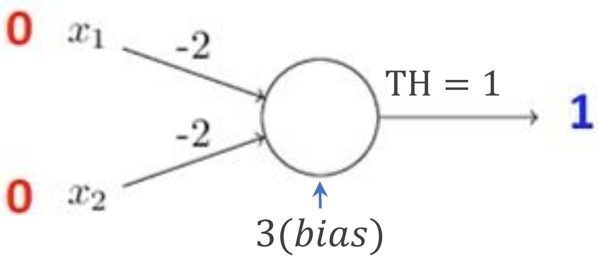
$$E = -(y\log(h) + (1-y)\log(1-h))$$



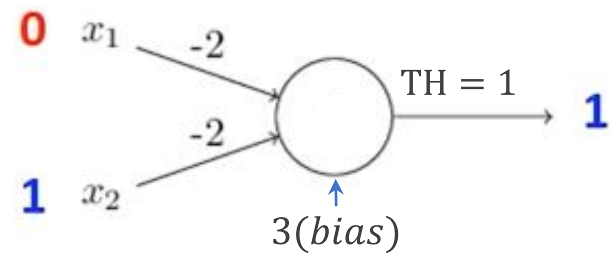
x_1	x_2	$AND(h)$
0	0	0
0	1	0
1	0	0
1	1	1

NAND

- NAND gates are functionally complete.
- We can build any logical functions out of them.



$(-2)*0 + (-2)*0 + 3 = 3$

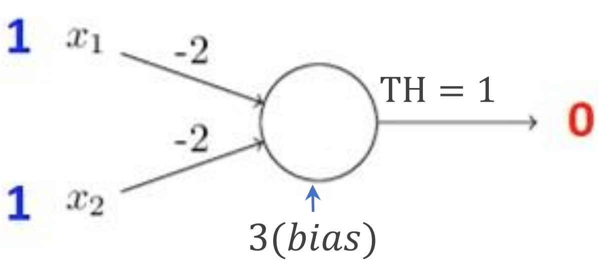


$(-2)*0 + (-2)*1 + 3 = 1$

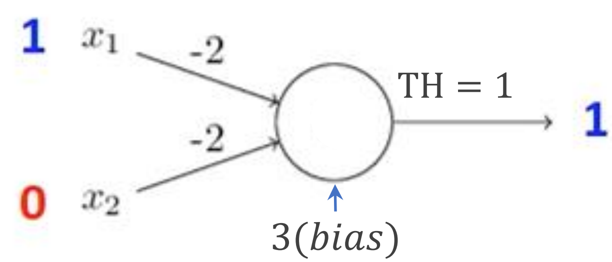
NAND

Truth Table

Input A	Input B		Output Q
0	0		1
0	1		1
1	0		1
1	1		0

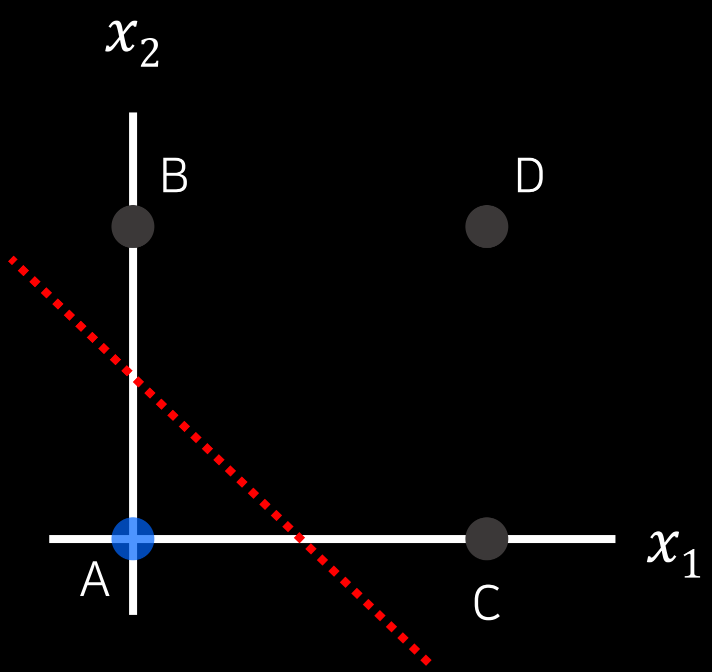


$(-2)*1 + (-2)*1 + 3 = -1$



$(-2)*1 + (-2)*0 + 3 = 1$

Decision boundary by a neuron



View from above

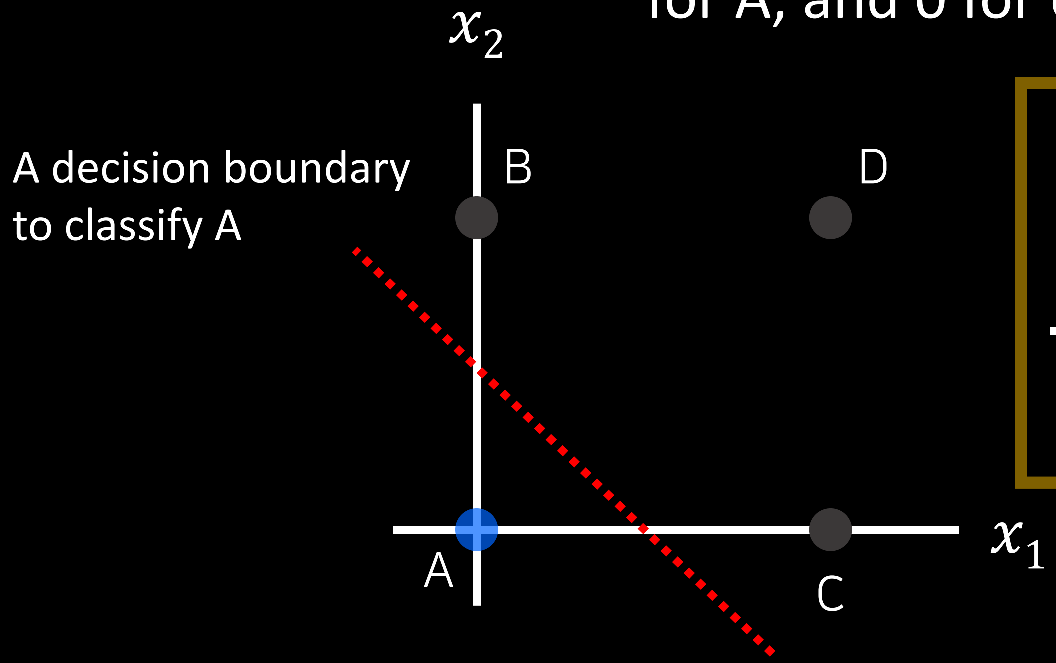
Decision boundary by a neuron

- A neuron, only 1 decision boundary
- A decision boundary yielding 2 classes (1 or 0)
- How to solve multiple classes more than 2

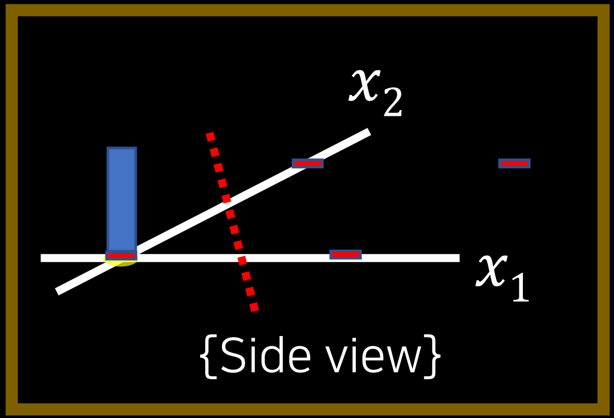
4-Class(A, B, C, D) Classification Problem

Neuron #1

The output of a neuron is 1 for A, and 0 for other cases.

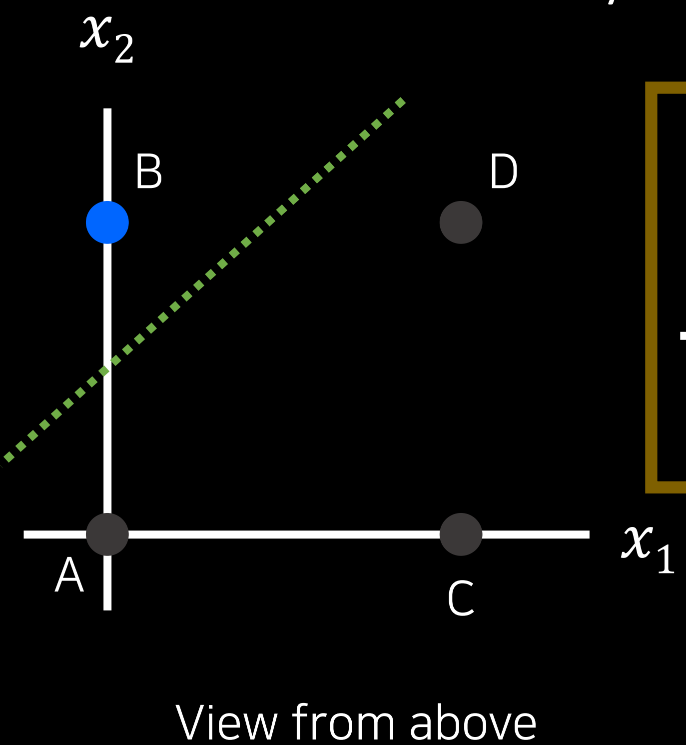


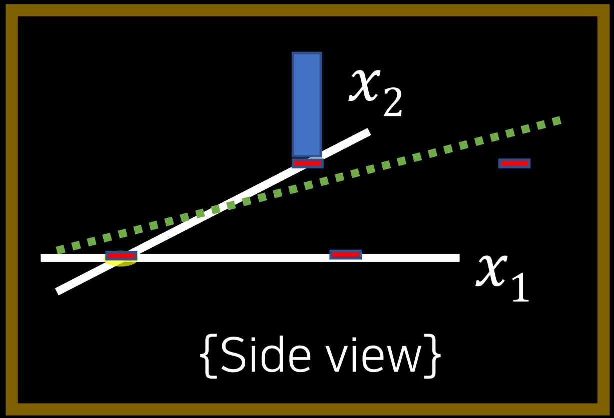
View from above



Neuron #2

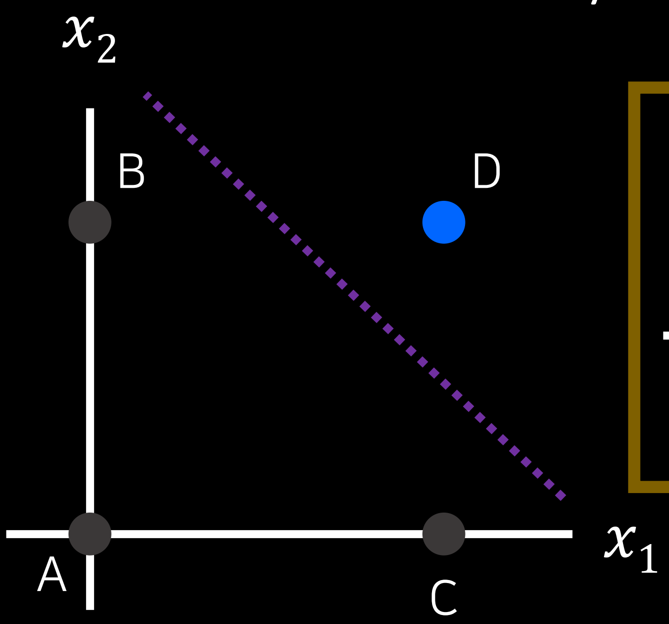
2[nd] neuron for 2[nd] decision boundary to classify B

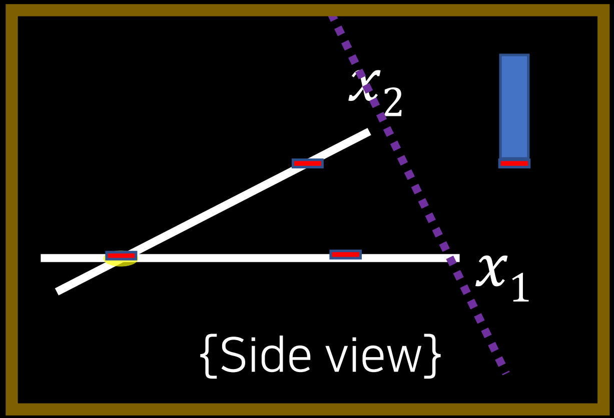




Neuron #3

3rd neuron for 3rd decision boundary to classify D

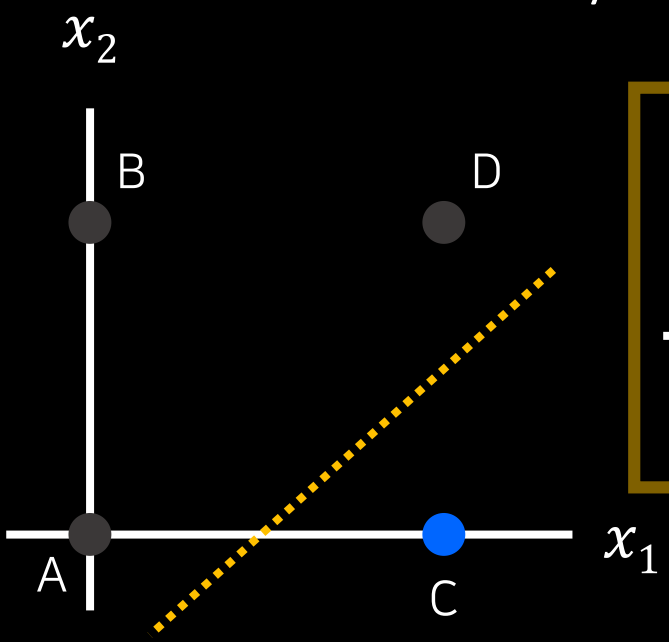




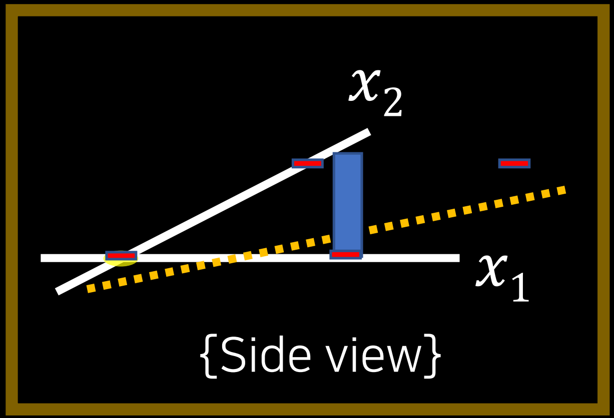
View from above

Neuron #4

4th neuron for 4th decision boundary to classify C



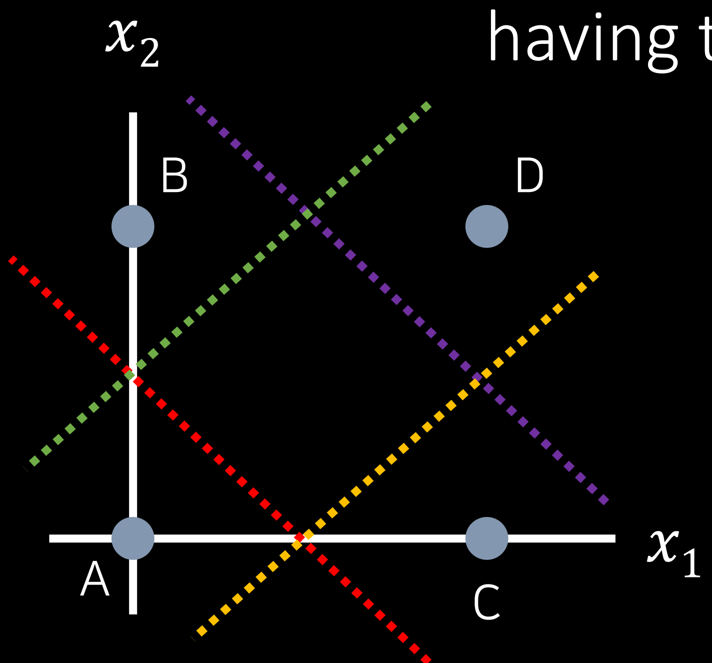
View from above



{Side view}

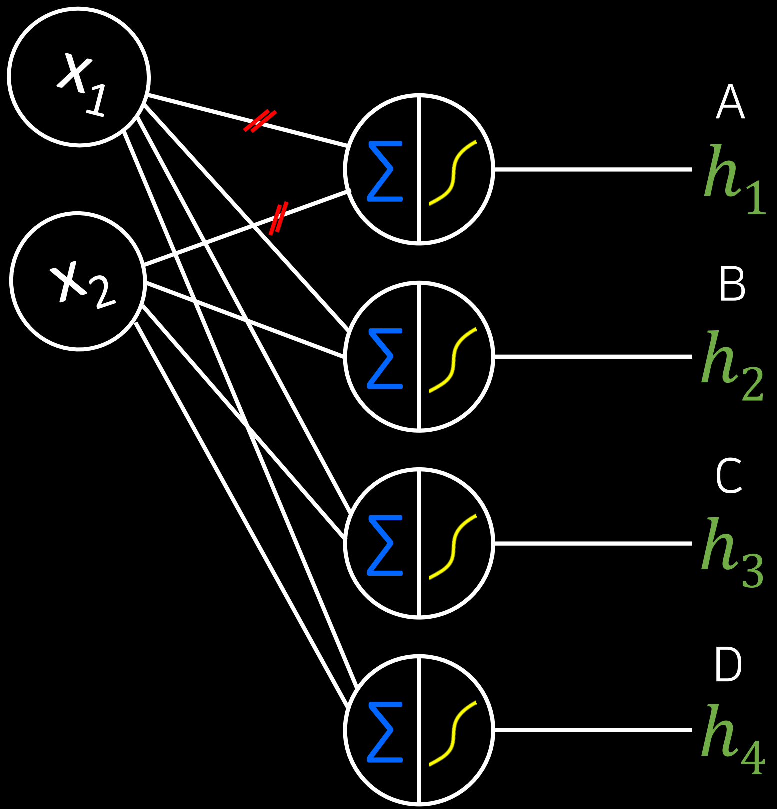
4 Neurons

4 neurons for
4 decision boundaries
having the same inputs

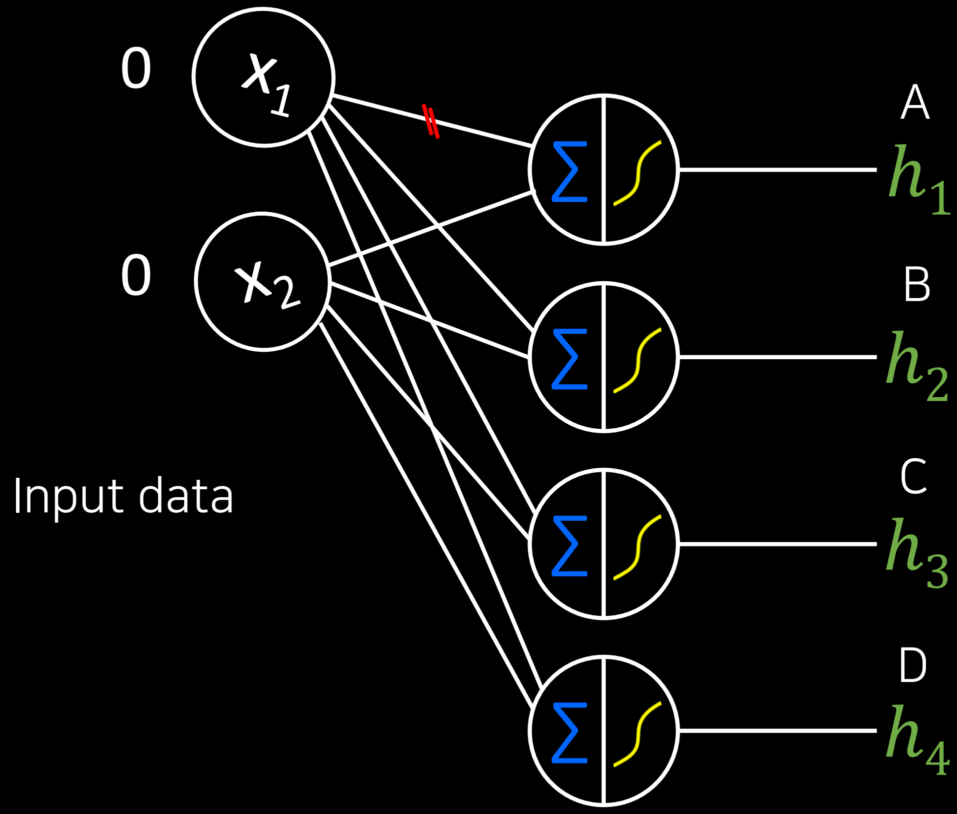


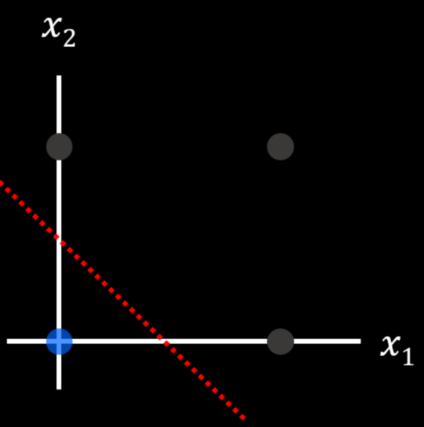
View from above

4 Neurons

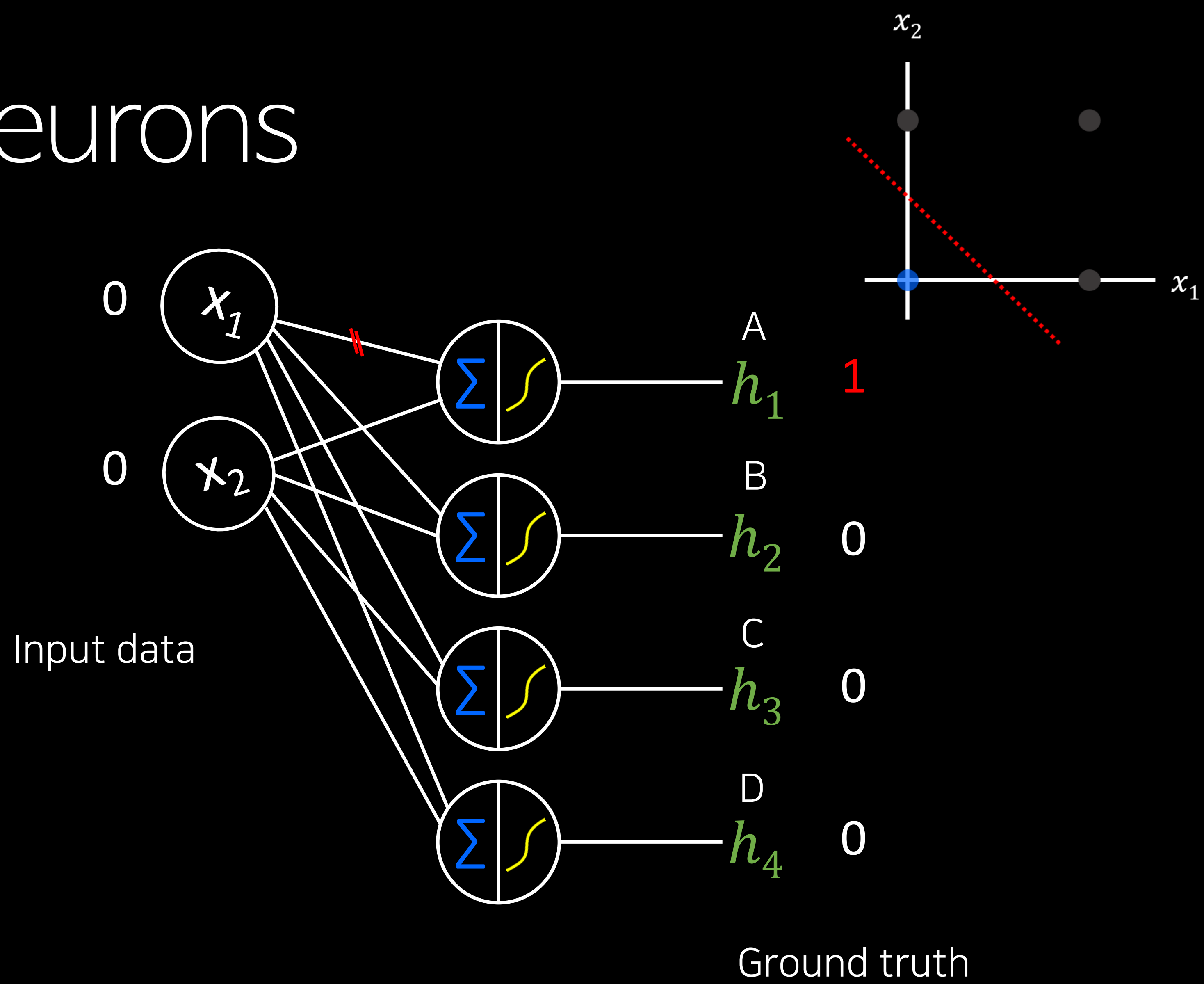


4 Neurons

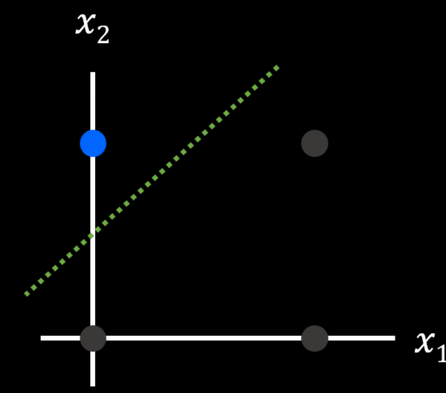


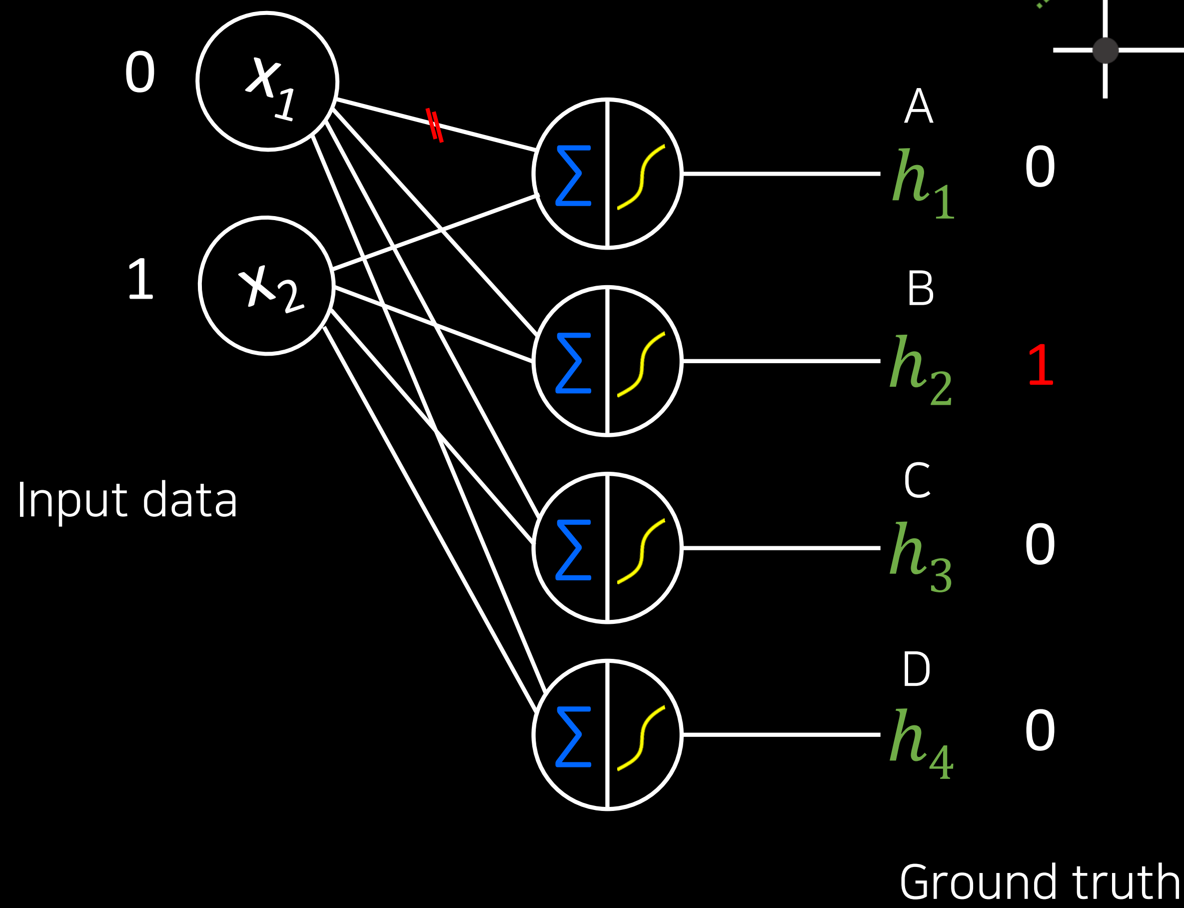


4 Neurons

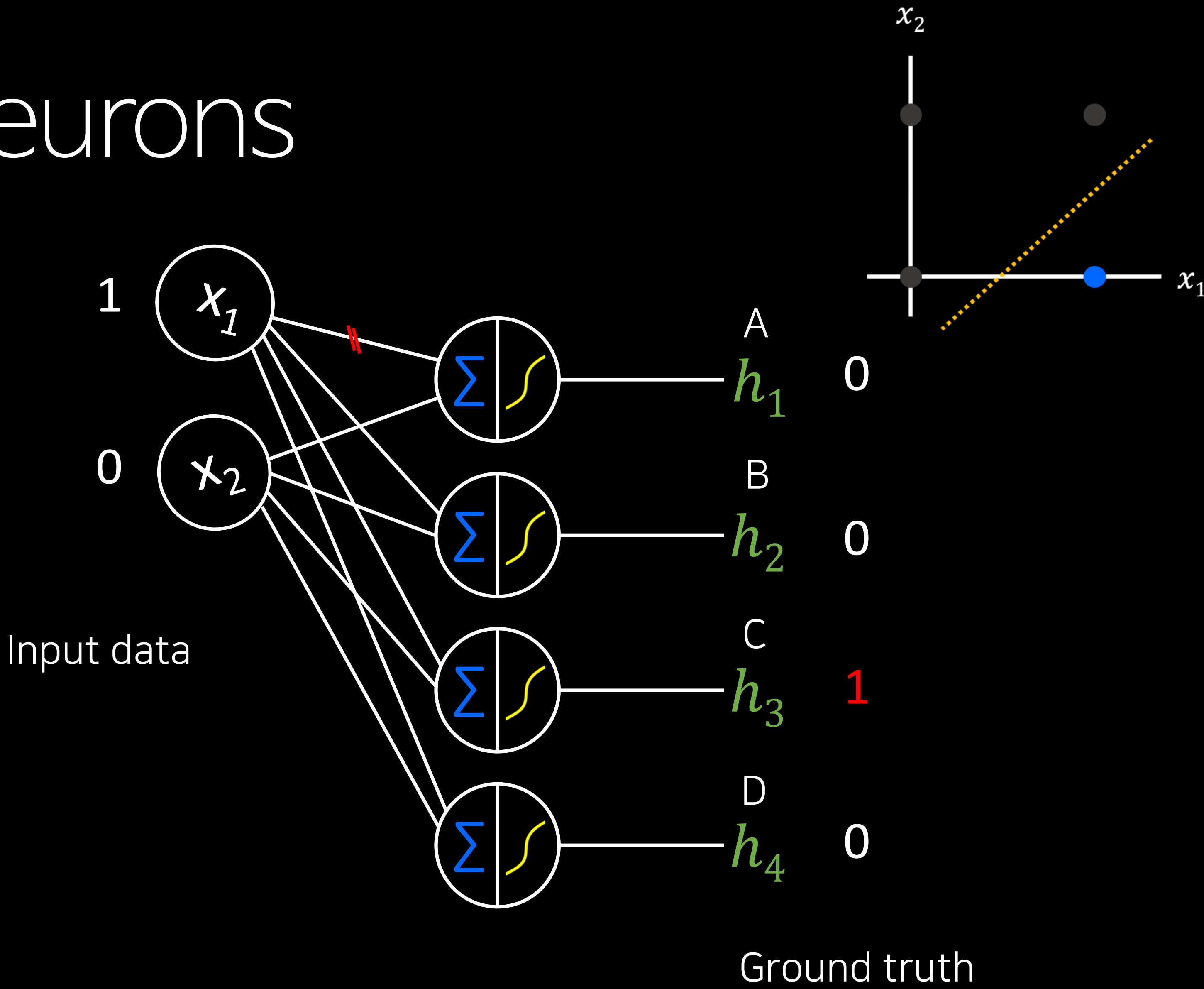


4 Neurons

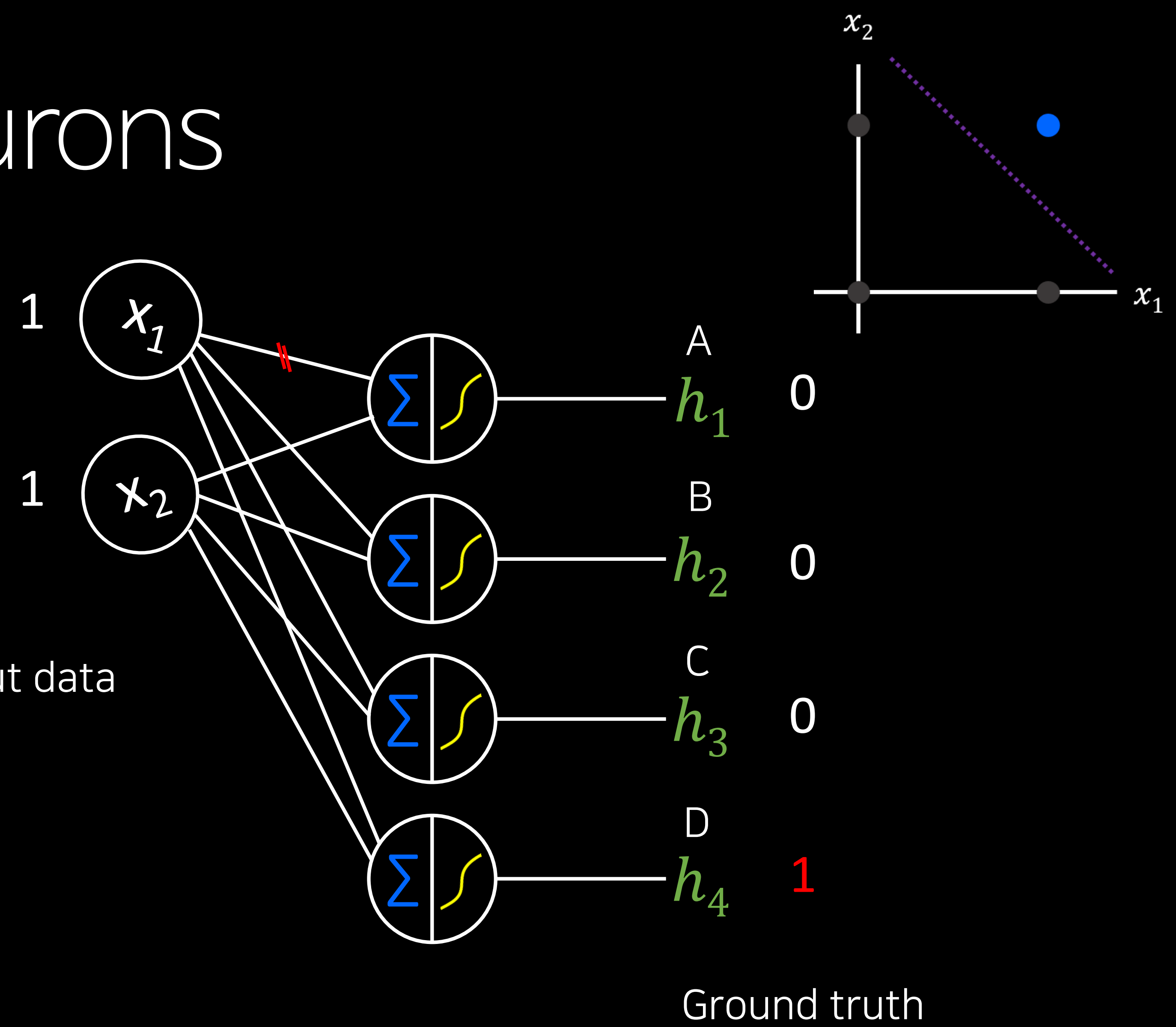




4 Neurons



4 Neurons



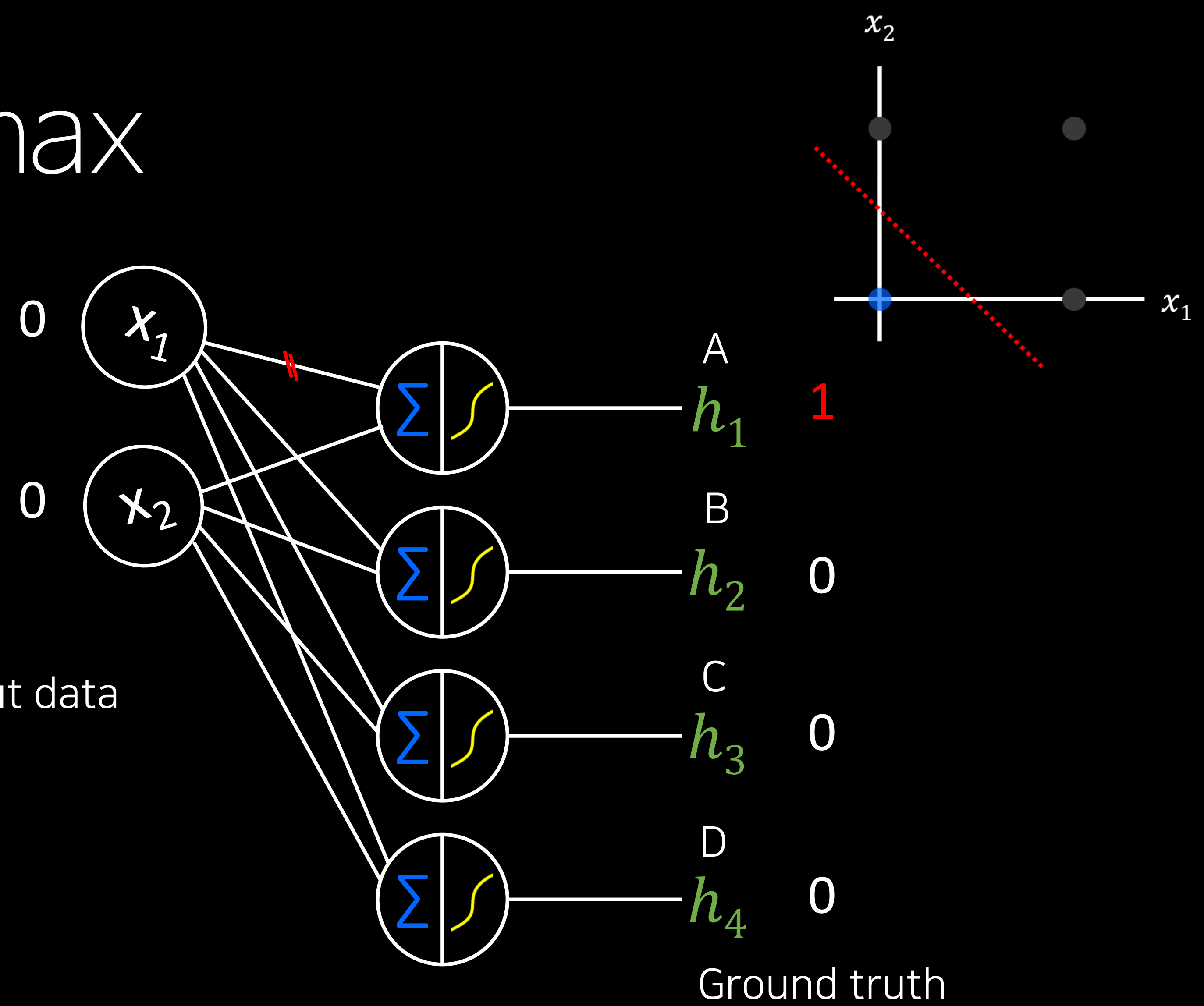
One-hot encoding

- Setting only one neuron's output as ON(1) and others as OFF(0)

Considerations

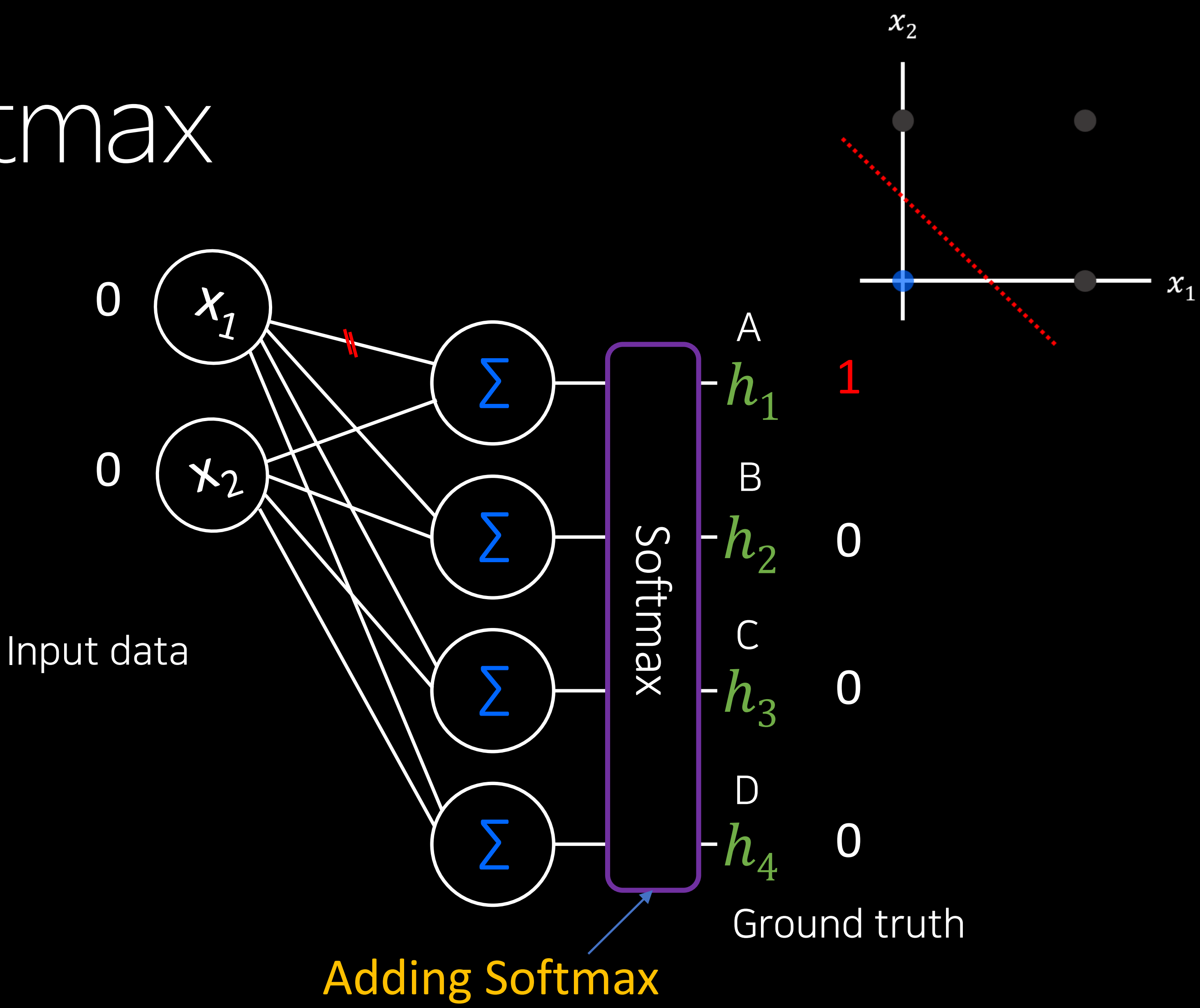
- If one output is 1, then the others must be 0.
- However, the four outputs are computed independently.
- No way to control the 4 outputs together
- A special function introduced → Softmax

Softmax



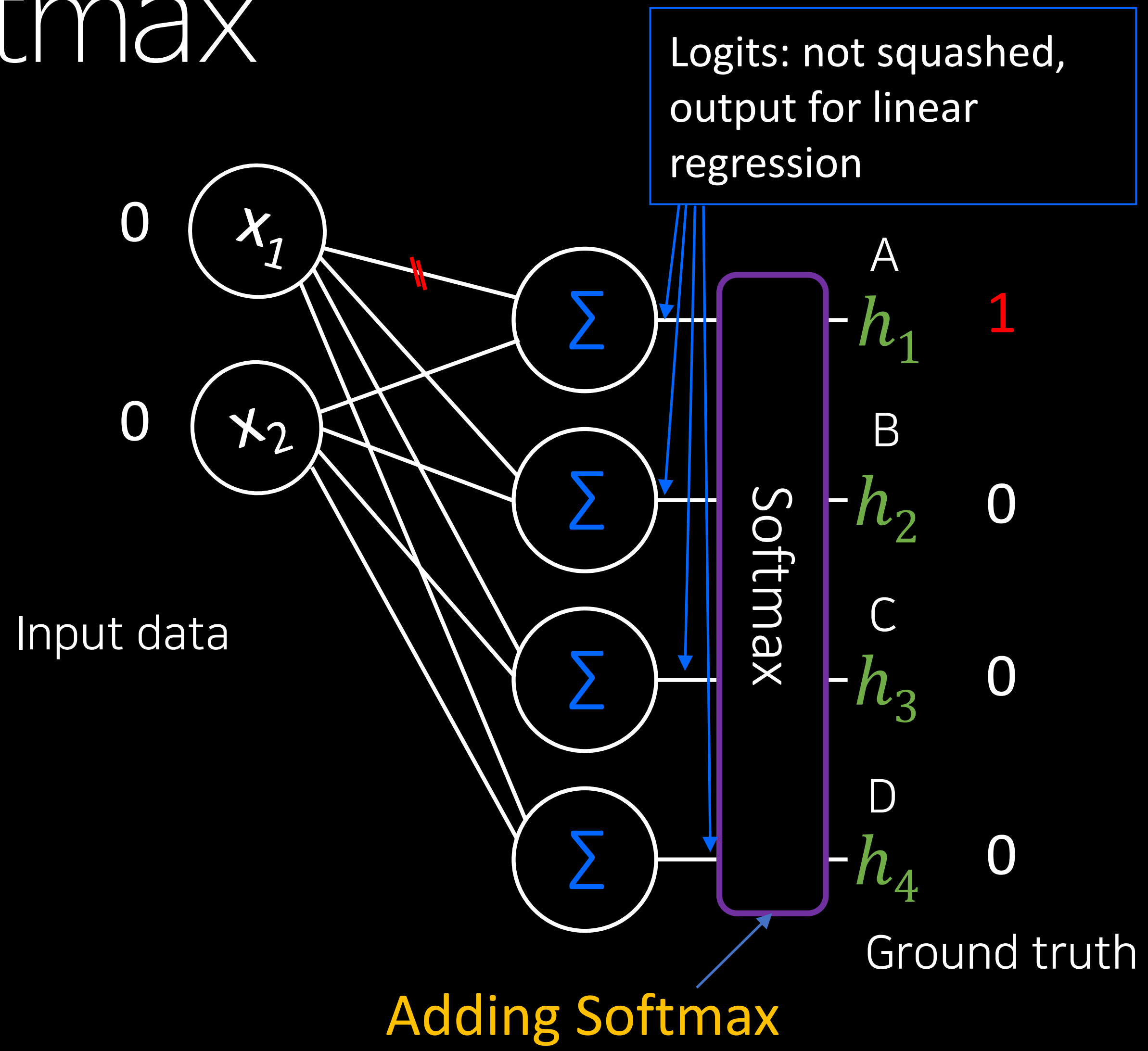
Initial architecture

Softmax



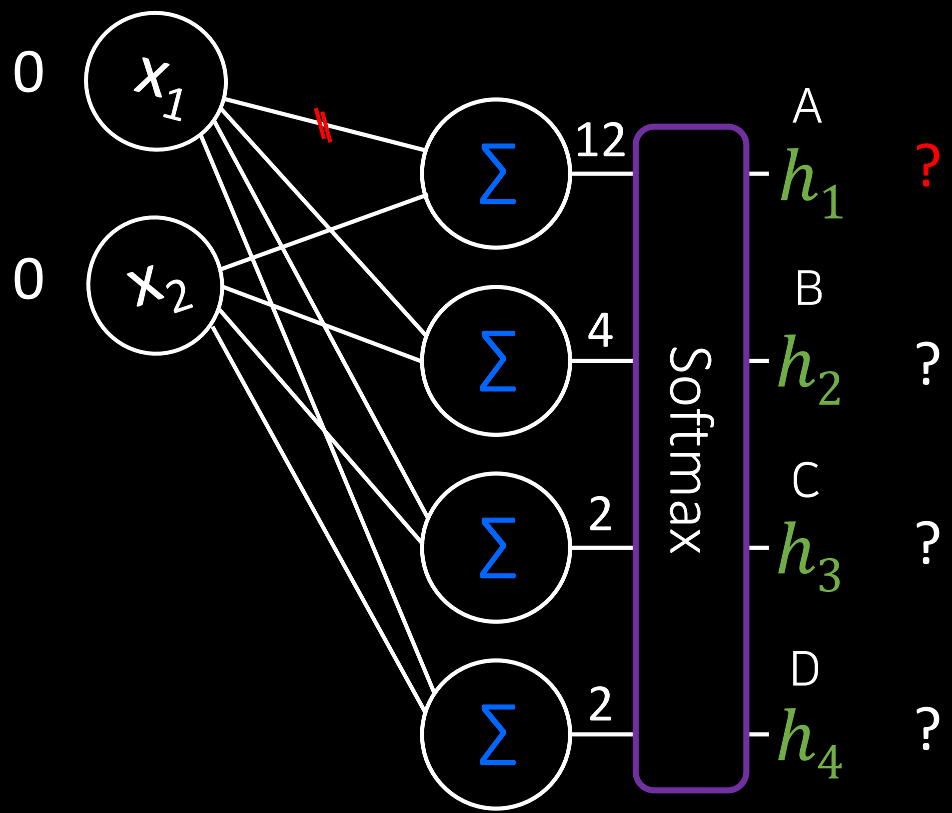
x_2
x_1
0
0
x_1
x_2
Input data
Softmax
A
h_1
1
B
h_2
0
C
h_3
0
D
h_4
0
Ground truth
Adding Softmax

Softmax



Logits: not squashed, output for linear regression
0
x_1
0
x_2
Input data
Σ
Σ
Σ
Σ
Softmax
A
h_1
1
B
h_2
0
C
h_3
0
D
h_4
0
Ground truth
Adding Softmax

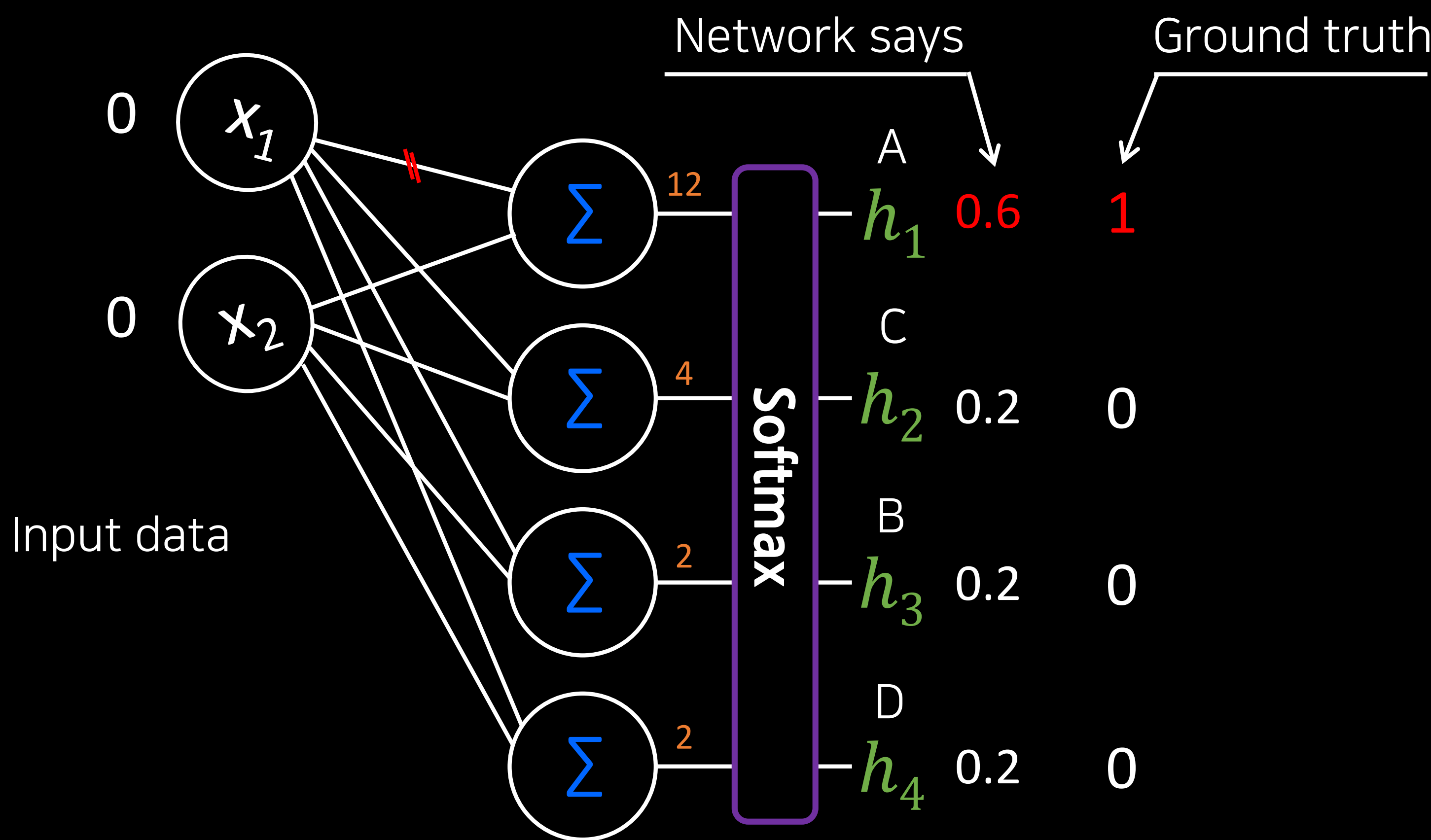
Softmax



Softmax

- For example, if the logits are 12, 4, 2, 2, then the Softmax function returns $\frac{12}{20}, \frac{4}{20}, \frac{2}{20}, \frac{2}{20}$ as results.
- Normalization of logits values
- Each value means the probability to be in the class.

Softmax



Network says
Ground truth
0
0
x_1
x_2
Input data
12
4
2
2
Softmax
A
C
B
D
h_1
h_2
h_3
h_4
0.6
0.2
0.2
0.2
1
0
0
0

New Loss/Error Function

- Distance between the output of a network(softmax) and correct answer (ground truth)
- If answer correctly, then the distance is 0,
- If not(incorrect), then the distance is ∞

New loss function

If answer correctly, then the distance is 0.

Softmax correct answer

$$\begin{bmatrix}1\\0\\0\\0\end{bmatrix} \odot \begin{bmatrix}1\\0\\0\\0\end{bmatrix} \rhd ?$$

$-\log$ (Softmax) Correct answer

$$\begin{bmatrix}0\\\infty\\\infty\\\infty\end{bmatrix} \odot \begin{bmatrix}1\\0\\0\\0\end{bmatrix} \rhd 0$$

$0 \to \infty$

$1 \to 0$

New loss function

If incorrect, then the distance is ∞.

Softmax, correct answer

$$\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \odot \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \triangleright ?$$

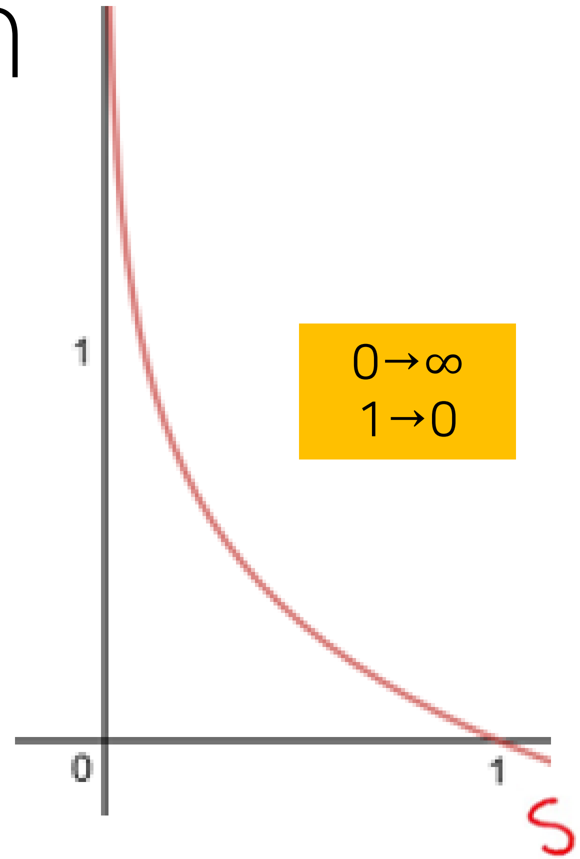
$0 \rightarrow \infty$
$1 \rightarrow 0$

$-\log$ (Softmax), Correct answer

$$\begin{bmatrix} \infty \\ 0 \\ \infty \\ \infty \end{bmatrix} \odot \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \triangleright \infty$$

$-\log$ function

$-\log(S)$



The output of softmax

$$-L \log(S)$$

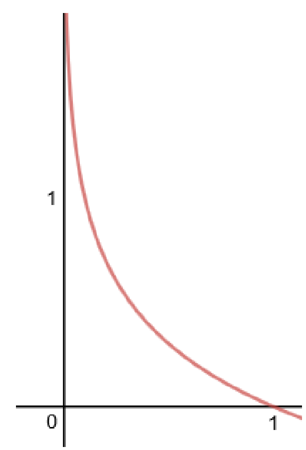
correct answer *L*

$$-\sum_i L_i \log(S_i)$$

New loss function

$$D(S, L) = -\sum_i L_i \log(S_i)$$

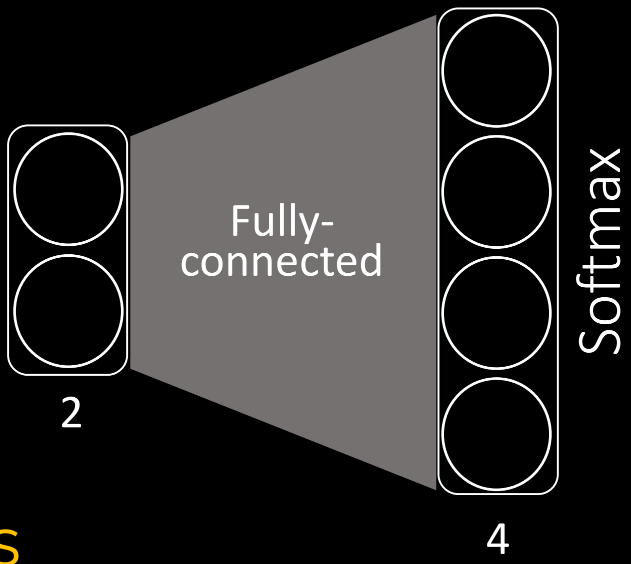
$$\underbrace{\begin{bmatrix}0.7\\0.2\\0.1\end{bmatrix}}_{S(Y)} \qquad \underbrace{\begin{bmatrix}1.0\\0.0\\0.0\end{bmatrix}}_{L}$$



softmax_cross_entropy_with_logits(logits, y_data)

The function returns 0 if the network answer correctly, and returns ∞ if not.

Lab 14.py



- Classification into one of four classes
- 4 neurons where each has 2-input
- A bias for each neuron