国際融合科学論/先端融合科学論

LECTURE 03
Machine Learning II: modern style machine learning

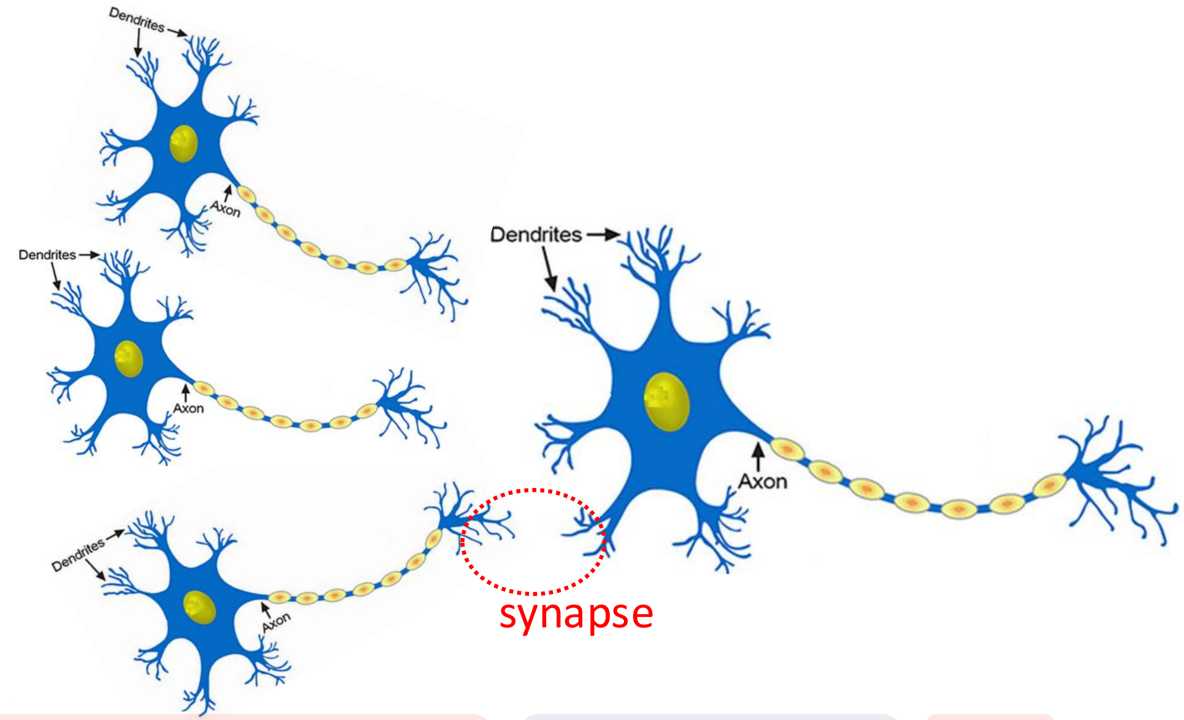
Dr. Suyong Eum

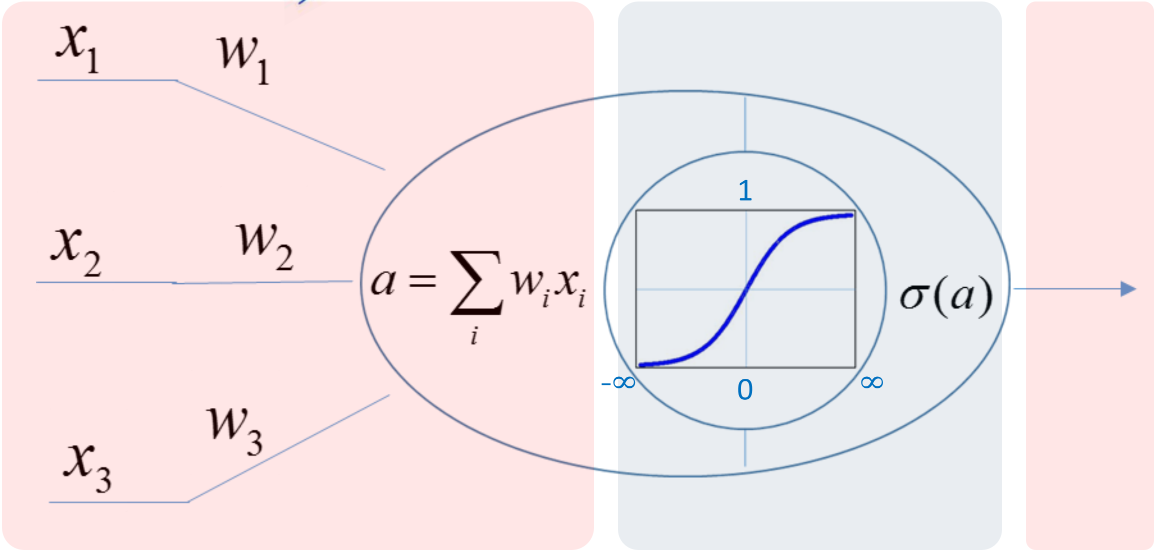
OSAKA UNIVERSITY

Lecture Outline

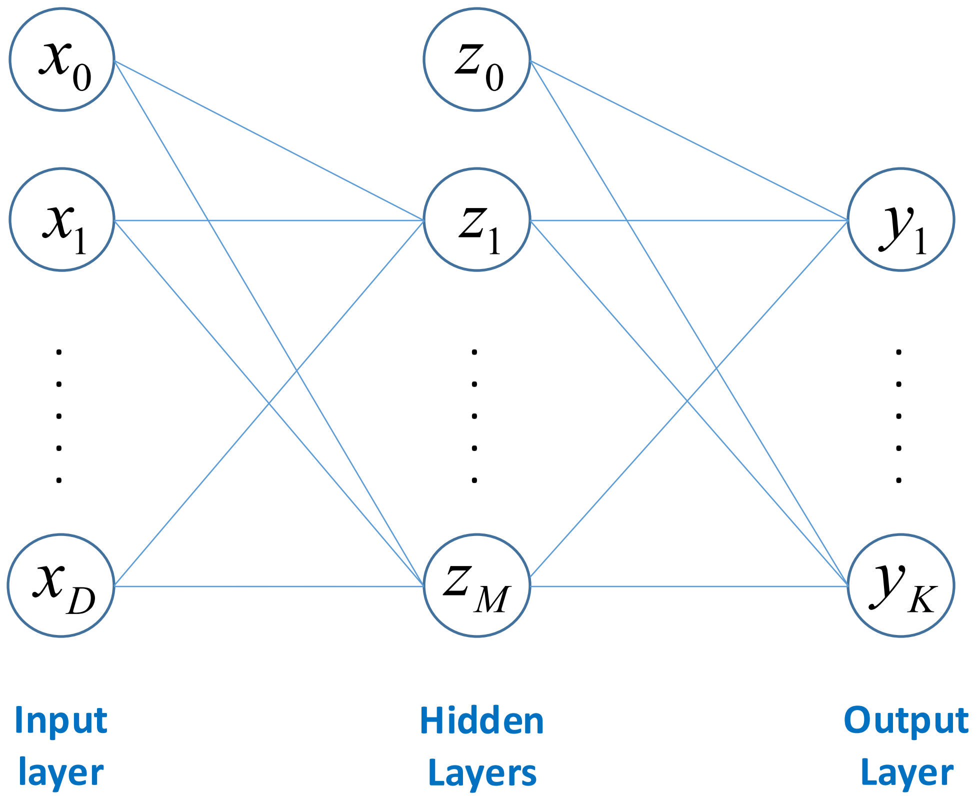
- ❑ Neural Networks (NNs)
  - Neural Network Architecture
  - Computation in NNs: Forward and Backpropagation
  - Activation Functions and Weight Initialization.

Neural networks: A bio-inspired approach



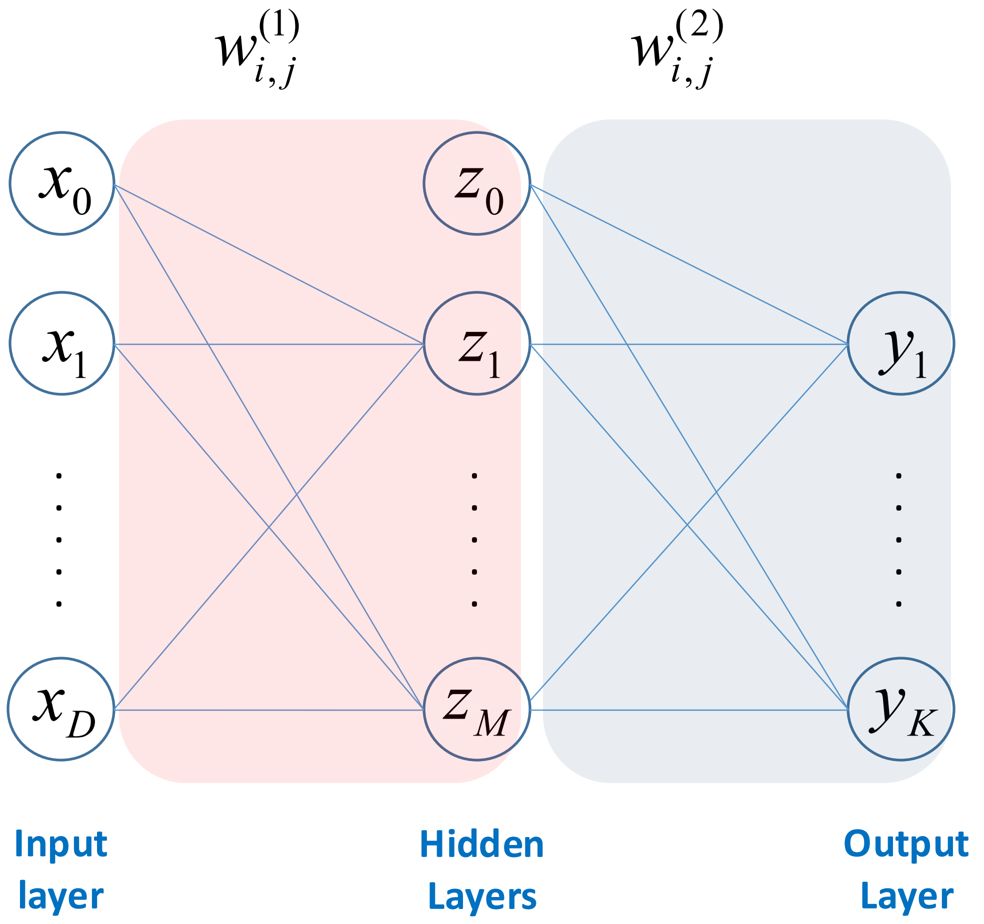


Neural networks: Terminology in neural networks



How many layers it has?

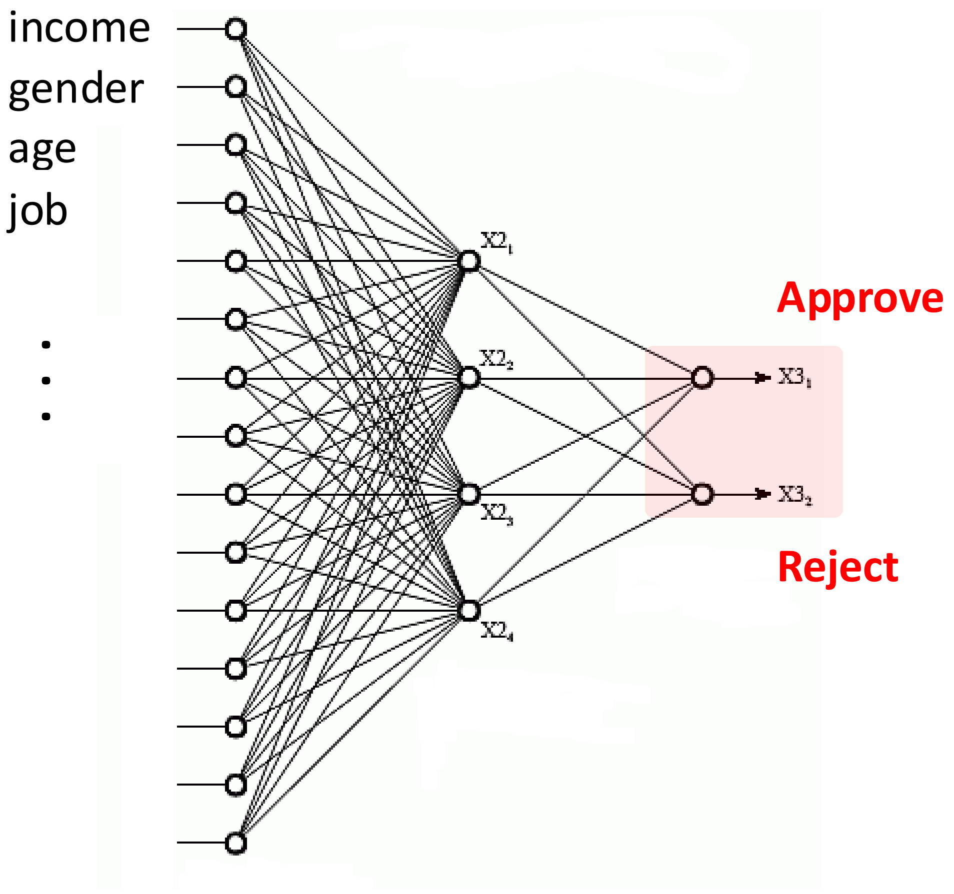
Neural networks: Terminology in neural networks



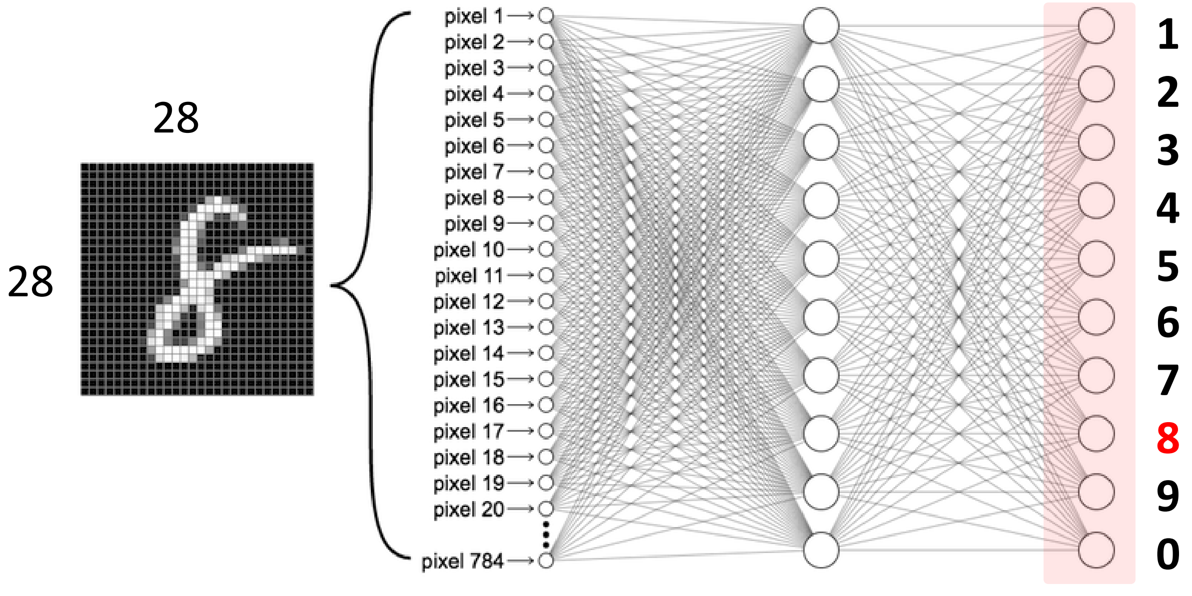
$w_{i,j}^{(\ell)}$: weight on a link at layer (ℓ) between node i and j

- In general, a standard L-layer neural network consists of
  - an input layer,
  - (L-1) hidden layers,
  - an output layer.

Neural networks: Example structures of neural networks

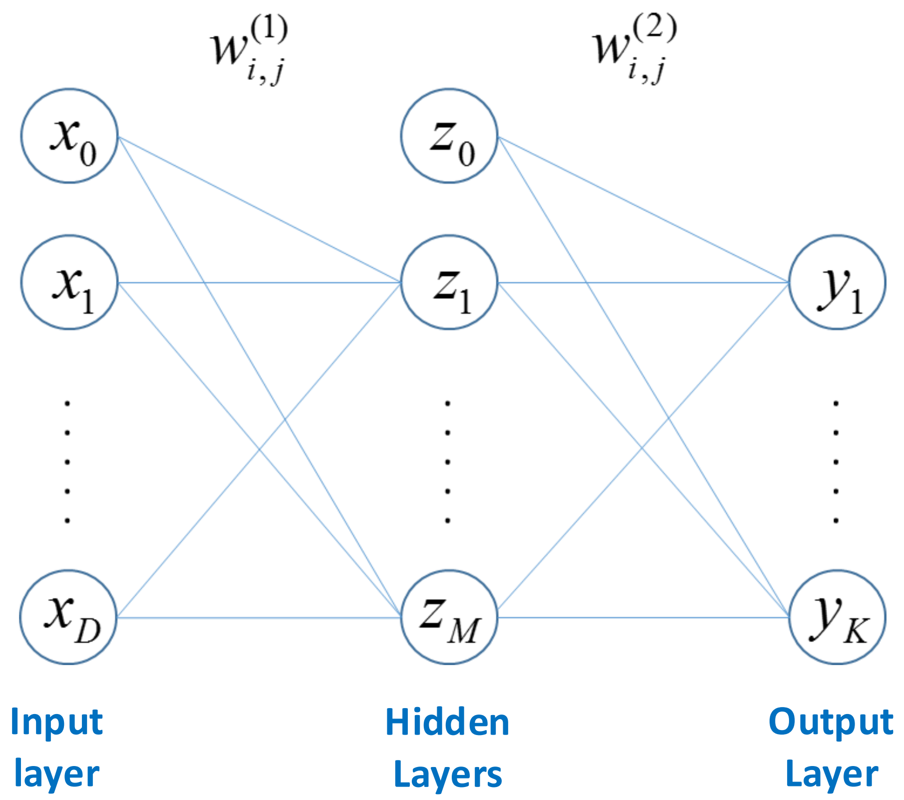


Credit card approval

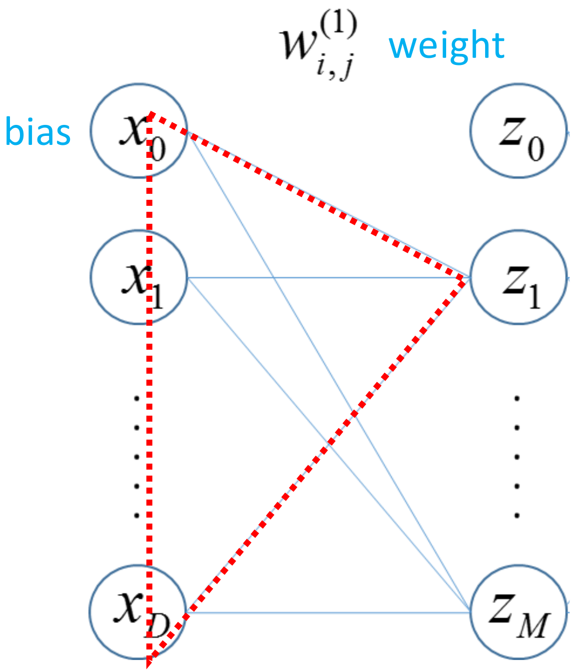


Digit recognition

Neural networks: Model



Neural networks: Model - Bias and weight



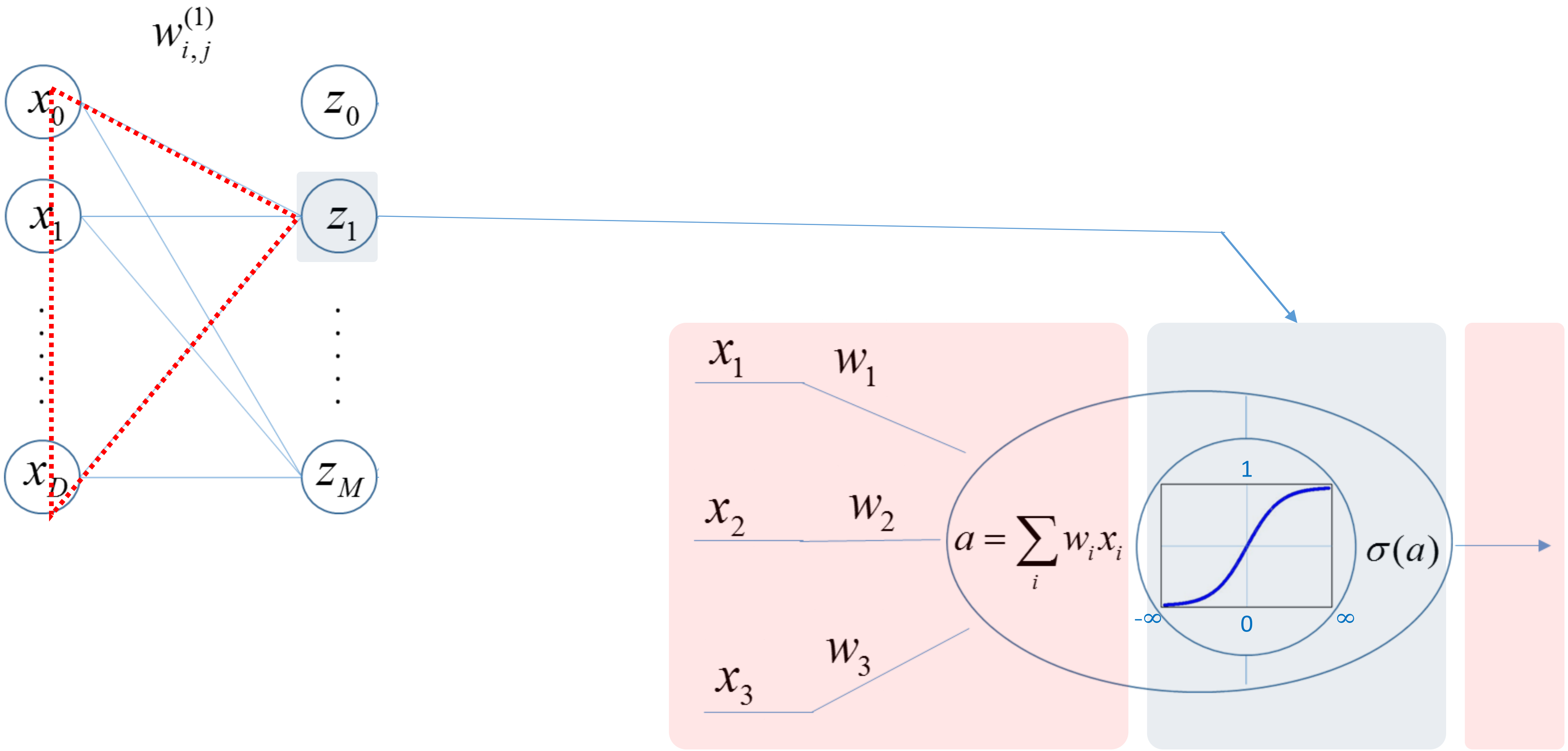
$$\boxed{w_{1,1}^{(1)}x_1 + w_{2,1}^{(1)}x_2 + w_{0,1}^{(1)}x_0}$$

bias

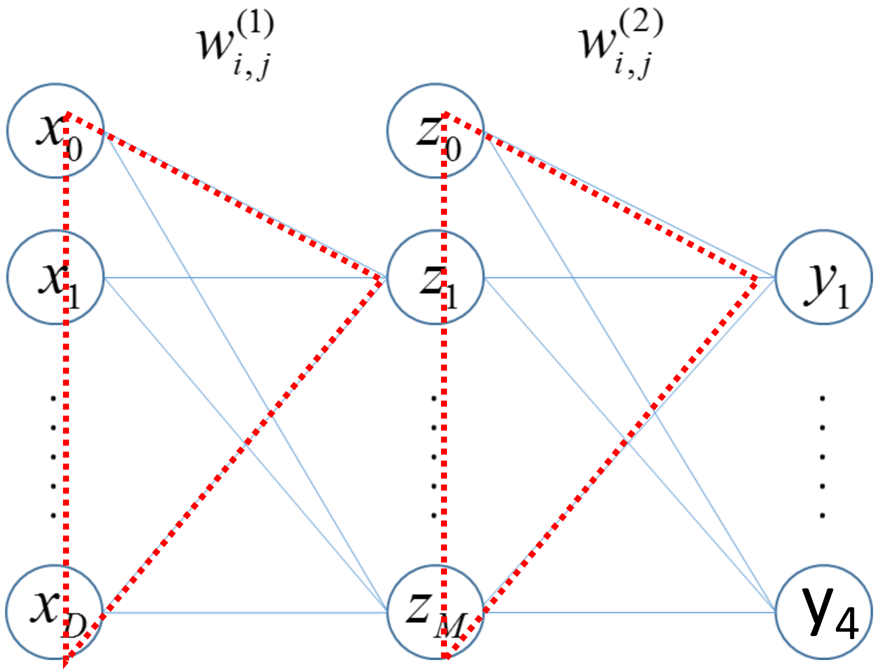
$$x_0 = 1$$

- ❑ Fully connected vs Partially connected

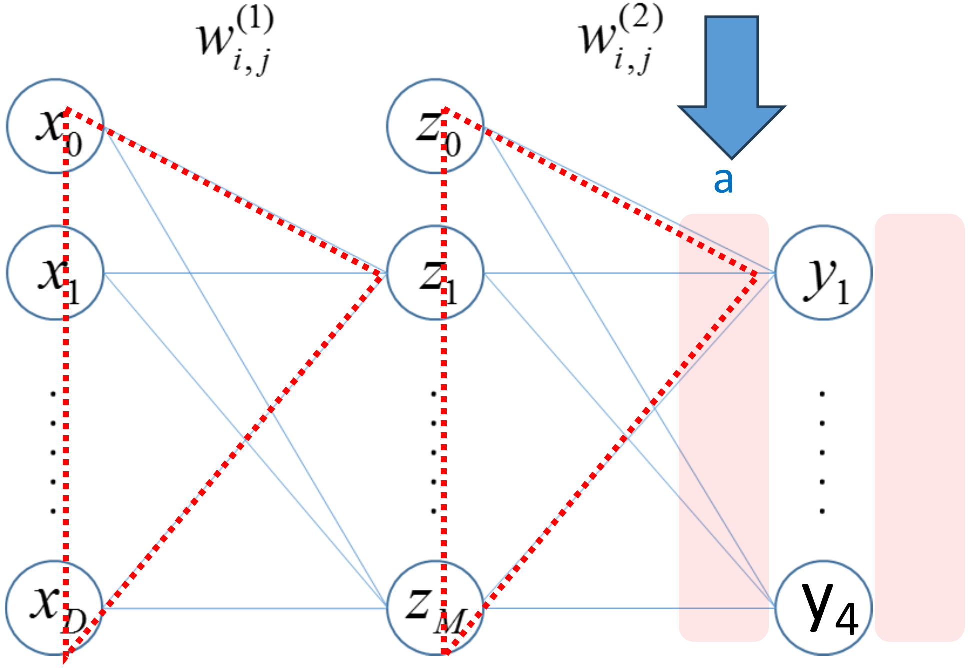
Neural networks: Model - Activation function

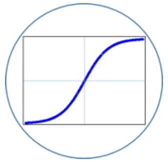


Neural networks: Model



Neural networks: Model



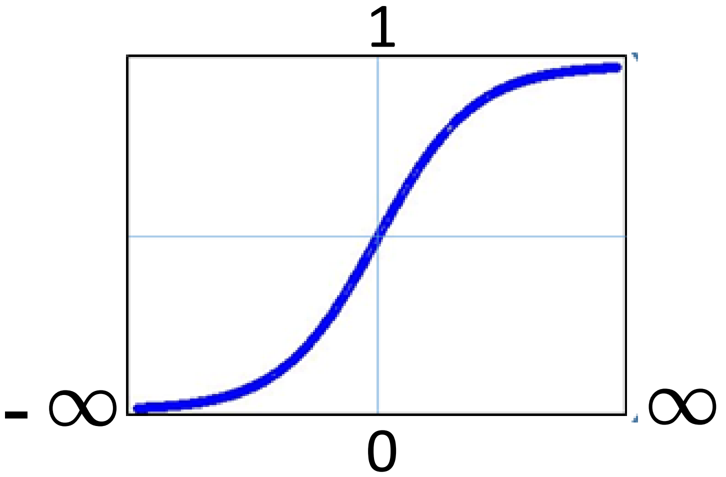


a

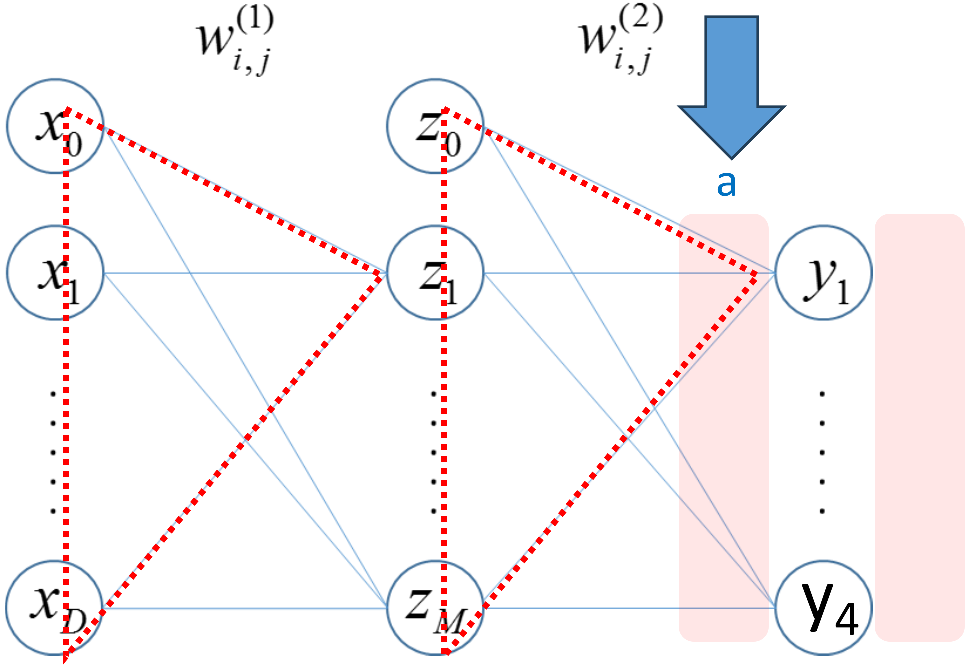
output		Sigmoid
-5	frog	0.00669
-1	bird	0.26894
1	dog	0.73106
5	cat	0.99331

$$\sigma_1(a) = \frac{1}{1+e^{-a}}$$

Sigmoid function

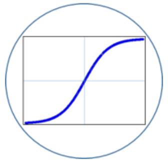


Neural networks: Softmax vs Normalization



a

output		Sigmoid	Normalization	Softmax
-5	frog	0.00669	0.00334	0.00004
-1	bird	0.26894	0.13447	0.00243
1	dog	0.73106	0.36553	0.01794
5	cat	0.99331	0.49666	0.97959

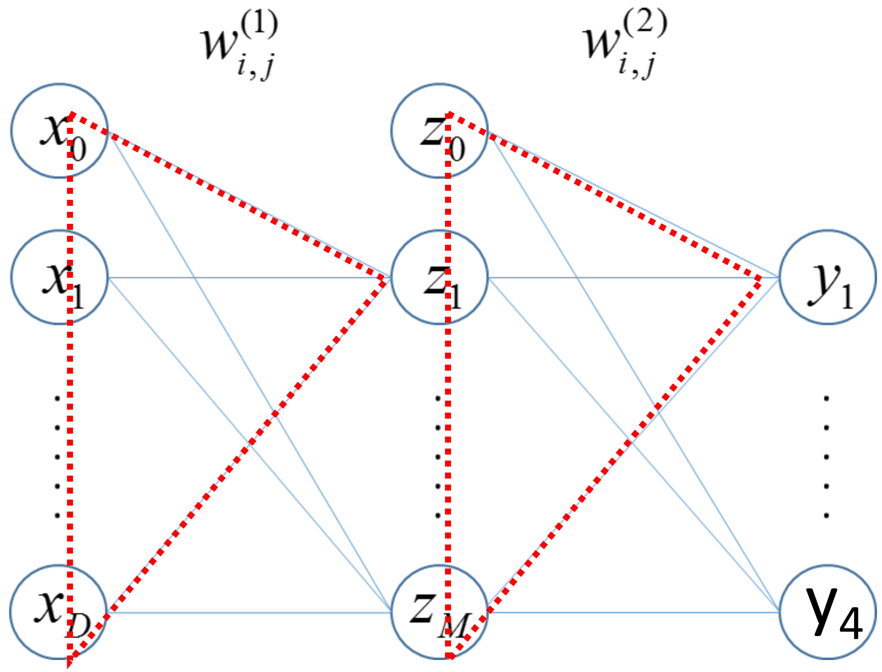


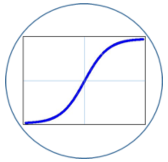
$$\sigma_1(a) = \frac{1}{1+e^{-a}}$$

Sigmoid function

$$\sigma_2(a_j) = \frac{e^{a_j}}{\sum_i e^{a_i}}$$

Neural networks: Cross entropy with Softmax





y

output		Sigmoid	Normalization	Softmax
-5	frog	0.00669	0.00334	0.00004
-1	bird	0.26894	0.13447	0.00243
1	dog	0.73106	0.36553	0.01794
5	cat	0.99331	0.49666	0.97959

t

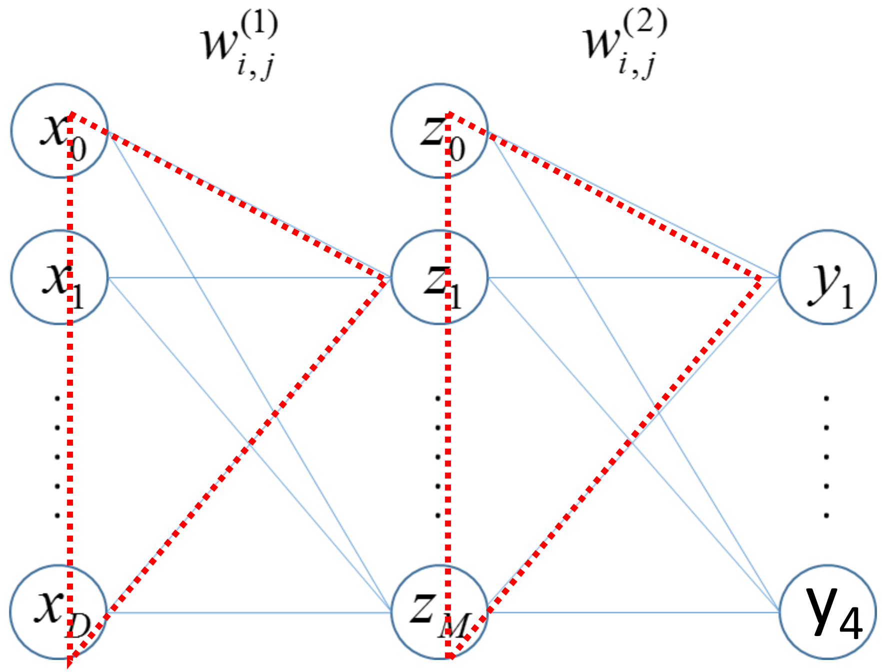
Label
0
0
0
1

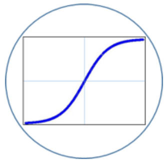
<u>One hot encoding</u>

$$H(\mathrm{y}) = -\sum_{\mathrm{i}} \mathrm{t_i} \ln(y_i) = 0.020621$$

ERROR between output (y) and label (t)

Neural networks: Cross entropy with Softmax





y

output		Sigmoid	Normalization	Softmax
-5	frog	0.00669	0.00334	0.00004
-1	bird	0.26894	0.13447	0.00243
1	dog	0.73106	0.36553	0.01794
5	cat	0.99331	0.49666	0.97959

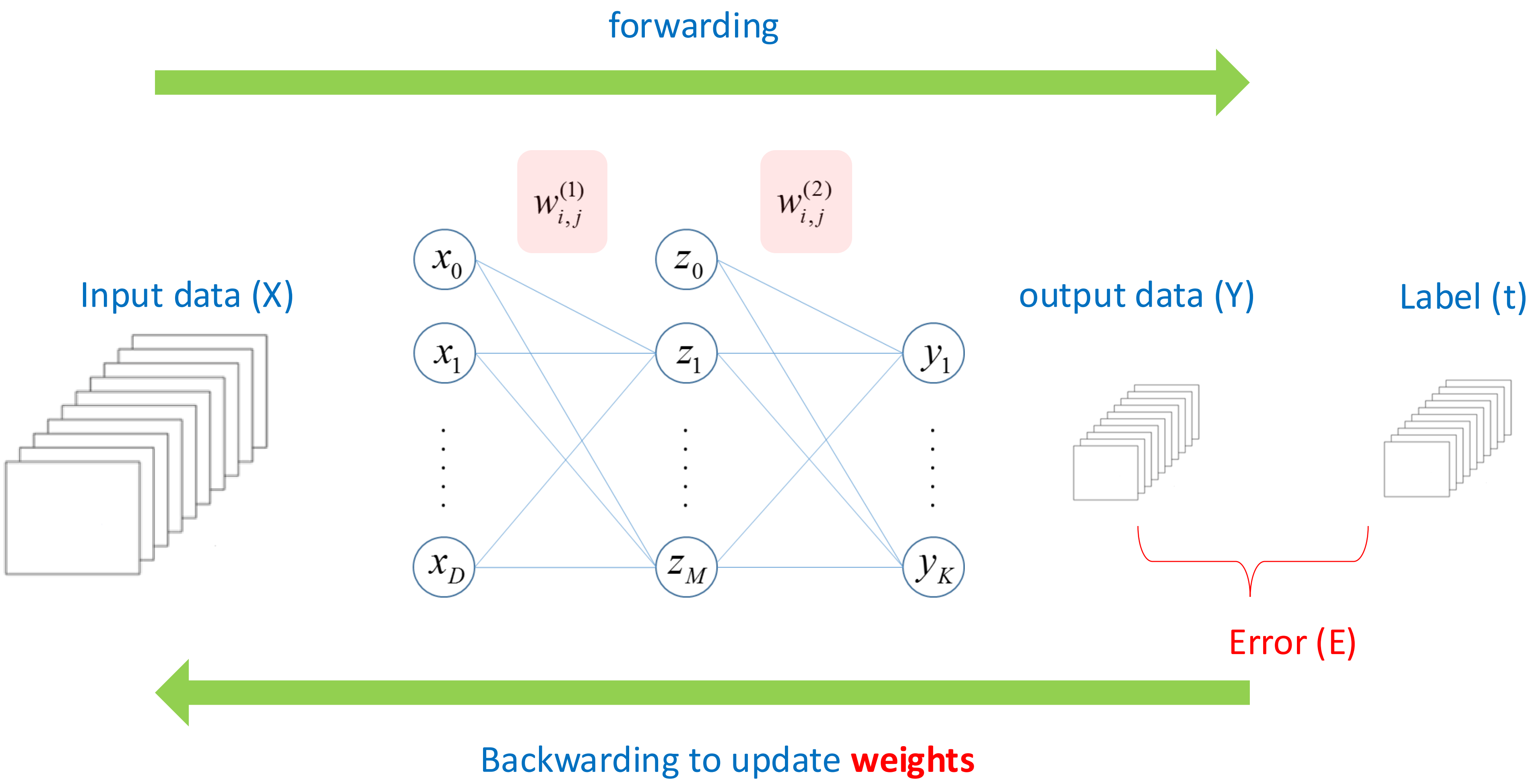
t

Label
0
0
0
1

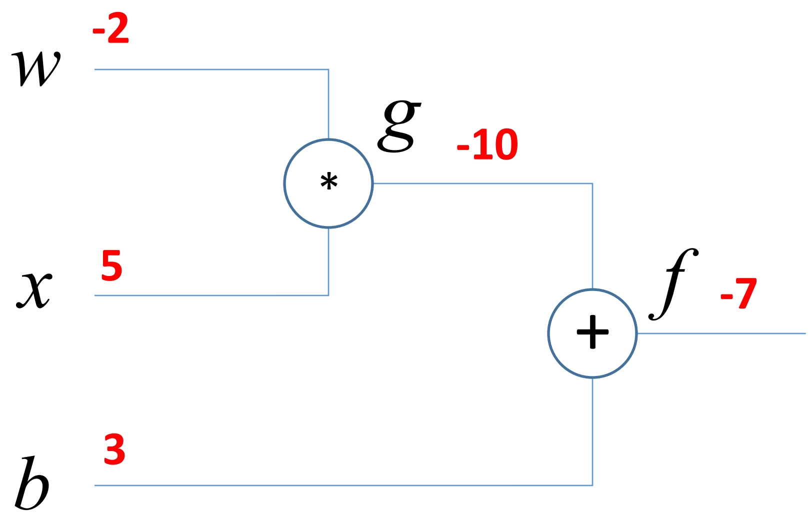
$$H(y) = -\sum_{i} t_i \ln(y_i) = 0.020621$$

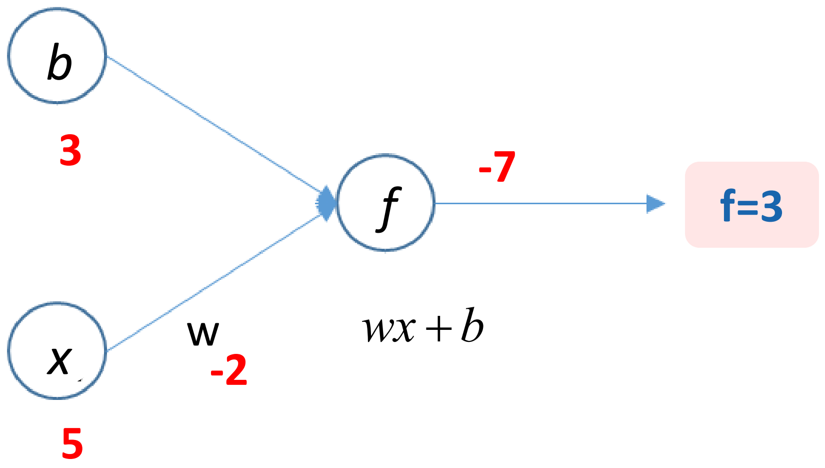
ERROR between output (y) and label (t)

Overview of the operation



Backpropagation: a toy example

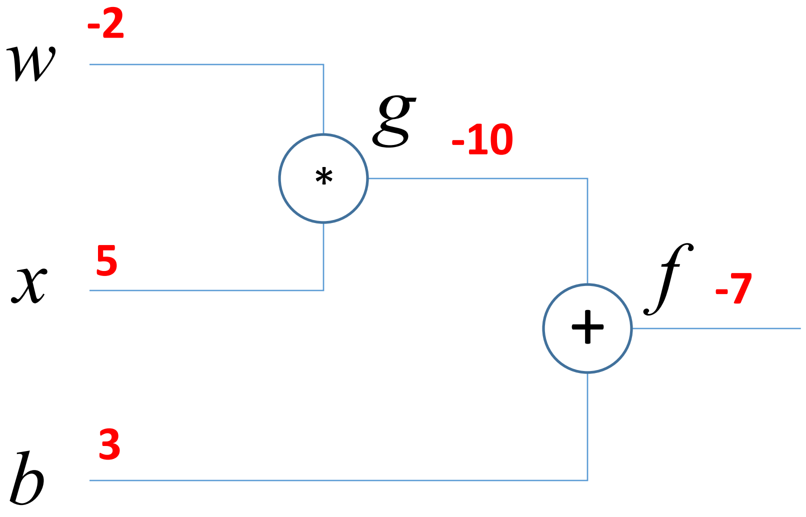




$$f = g + b$$

$$g = wx$$

Backpropagation: a toy example



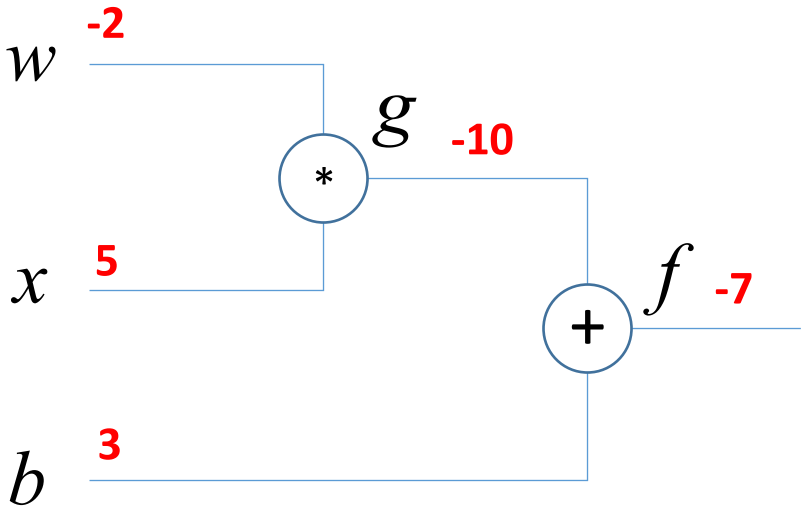
$$\frac{\partial f}{\partial w} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial w} = x = 5$$

When “w” is changed by 1 unit,
it will change the value of “f” by 5 unit.

$$f = g + b$$

$$g = wx$$

Backprogation: a toy example



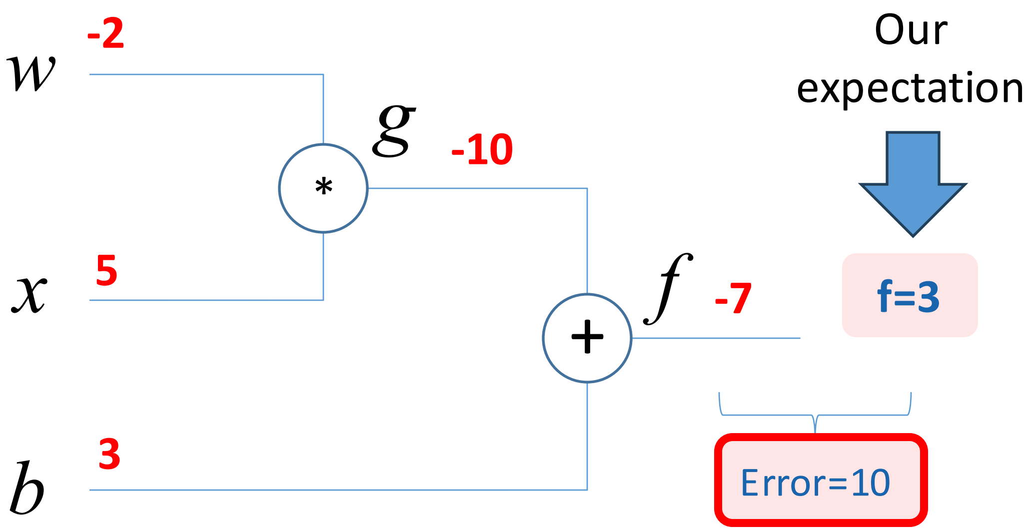
$$\frac{\partial f}{\partial w} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial w} = x = 5$$

$$\frac{\partial f}{\partial b} = 1$$

$$f = g + b$$

$$g = wx$$

Backpropagation: a toy example



$$\frac{\partial f}{\partial w} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial w} = x = 5$$

$$\frac{\partial f}{\partial b} = 1$$

$$f = g + b$$

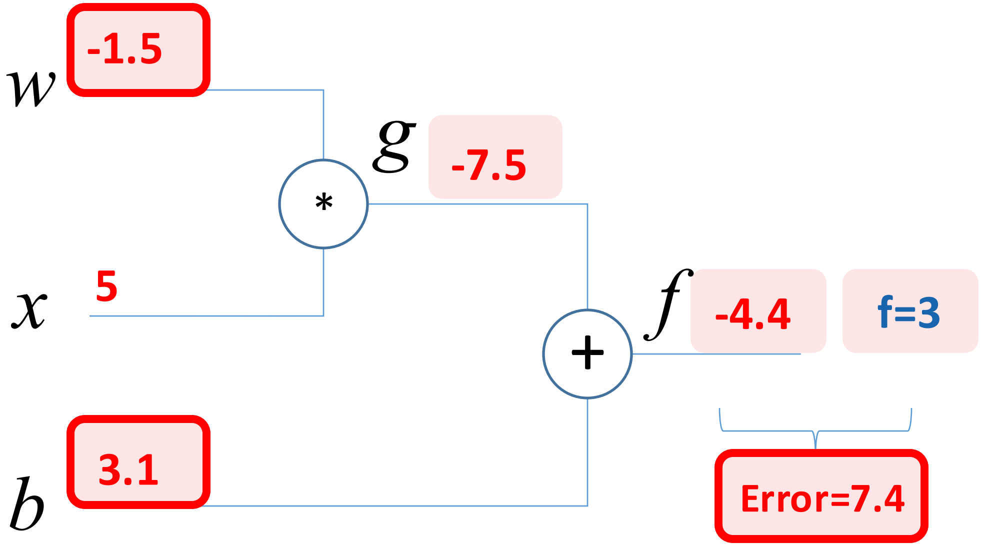
$$g = wx$$

- ❑ Assuming that the value of f should be "3".
- ❑ How to update variables which you are interested?

$$W_{new} = W_{old} + \eta \frac{\partial f}{\partial w_i}$$

$$b_{new} = b_{old} + \eta \frac{\partial f}{\partial b}$$

Backpropagation: a toy example: $\eta = 0.1$



$$\frac{\partial f}{\partial w} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial w} = x = 5$$

$$\frac{\partial f}{\partial b} = 1$$

$$f = g + b$$

$$g = wx$$

- ❑ Assuming that the value of f should be "3".
- ❑ How to update variables which you are interested?

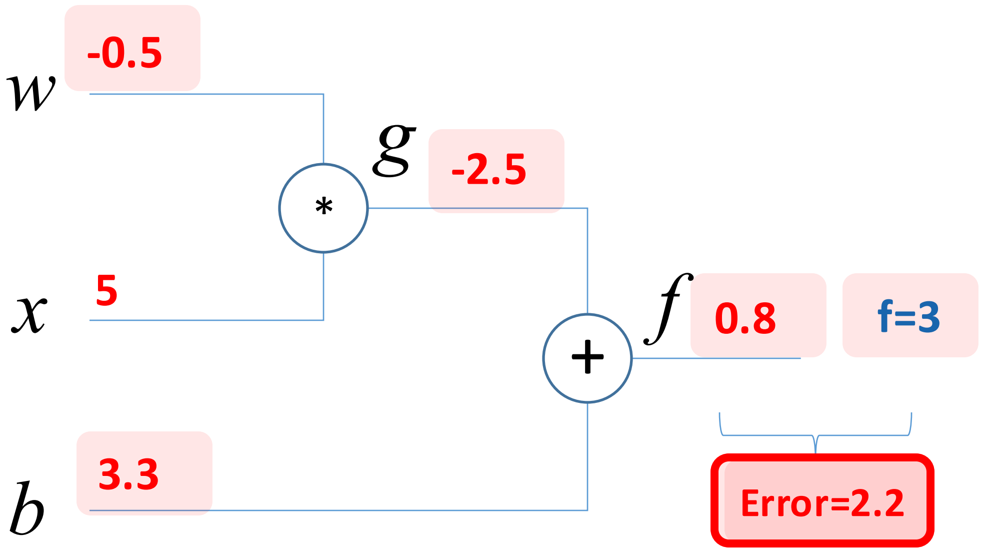
$$W_{new} = W_{old} + \eta \frac{\partial f}{\partial w_i}$$

$$b_{new} = b_{old} + \eta \frac{\partial f}{\partial b}$$

$$W_{new} = -2 + 0.1 \times 5 = \boxed{-1.5}$$

$$b_{new} = 3 + 0.1 \times 1 = \boxed{3.1}$$

Backpropagation: a toy example: $\eta = 0.3$



$$\frac{\partial f}{\partial w} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial w} = x = 5$$

$$\frac{\partial f}{\partial b} = 1$$

$$f = g + b$$

$$g = wx$$

- ❑ Assuming that the value of f should be "3".
- ❑ How to update variables which you are interested?

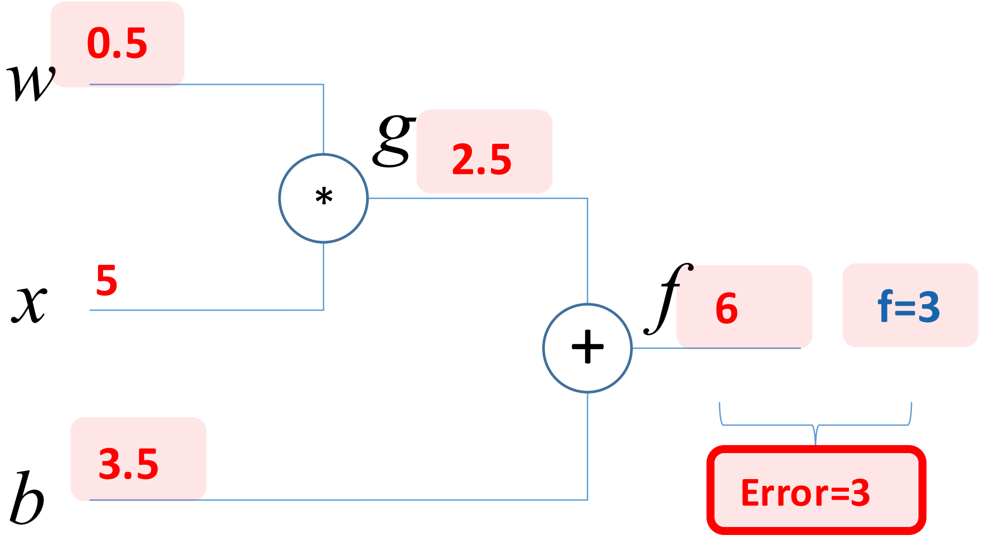
$$W_{new} = W_{old} + \eta \frac{\partial f}{\partial w_i}$$

$$b_{new} = b_{old} + \eta \frac{\partial f}{\partial b}$$

$$W_{new} = -2 + 0.3 \times 5 = -0.5$$

$$b_{new} = 3 + 0.3 \times 1 = 3.3$$

Backpropagation: a toy example: $\boxed{\eta = 0.5}$



$$\frac{\partial f}{\partial w} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial w} = x = 5$$

$$\frac{\partial f}{\partial b} = 1$$

$$f = g + b$$

$$g = wx$$

- ❑ Assuming that the value of f should be “3”.
- ❑ How to update variables which you are interested?

$$W_{new} = W_{old} + \eta \frac{\partial f}{\partial w_i}$$

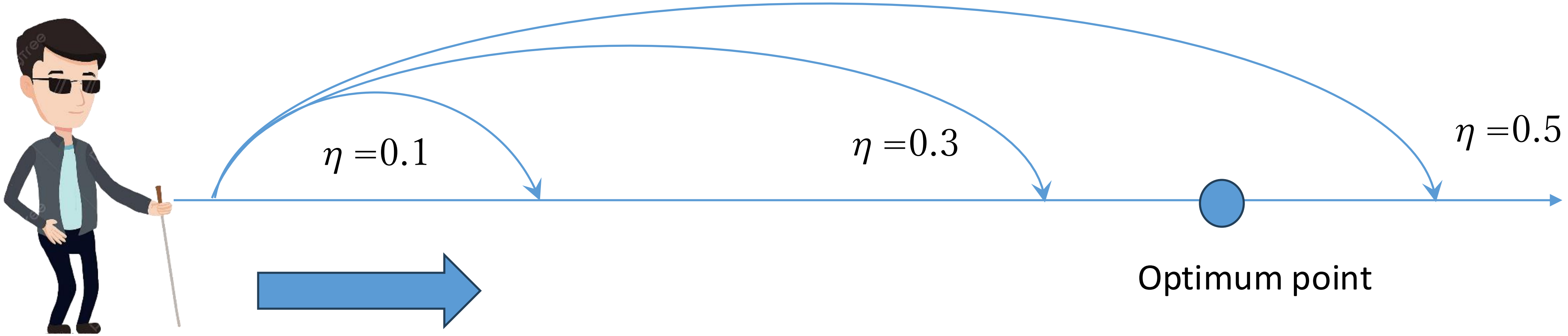
$$b_{new} = b_{old} + \eta \frac{\partial f}{\partial b}$$

$$W_{new} = -2 + 0.5 \times 5 = 0.5$$

$$b_{new} = 3 + 0.5 \times 1 = 3.5$$

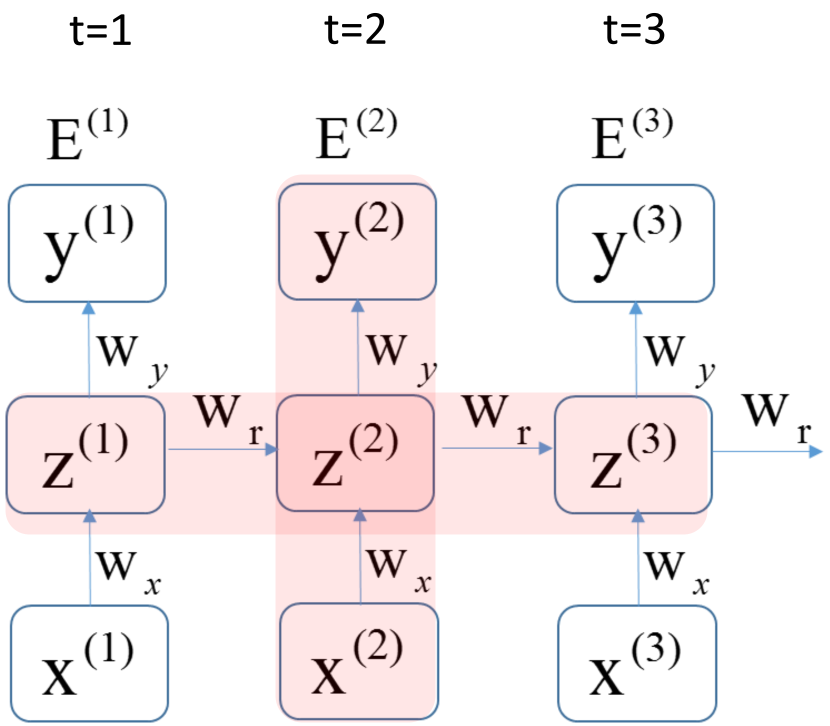
Backpropagation: a toy example

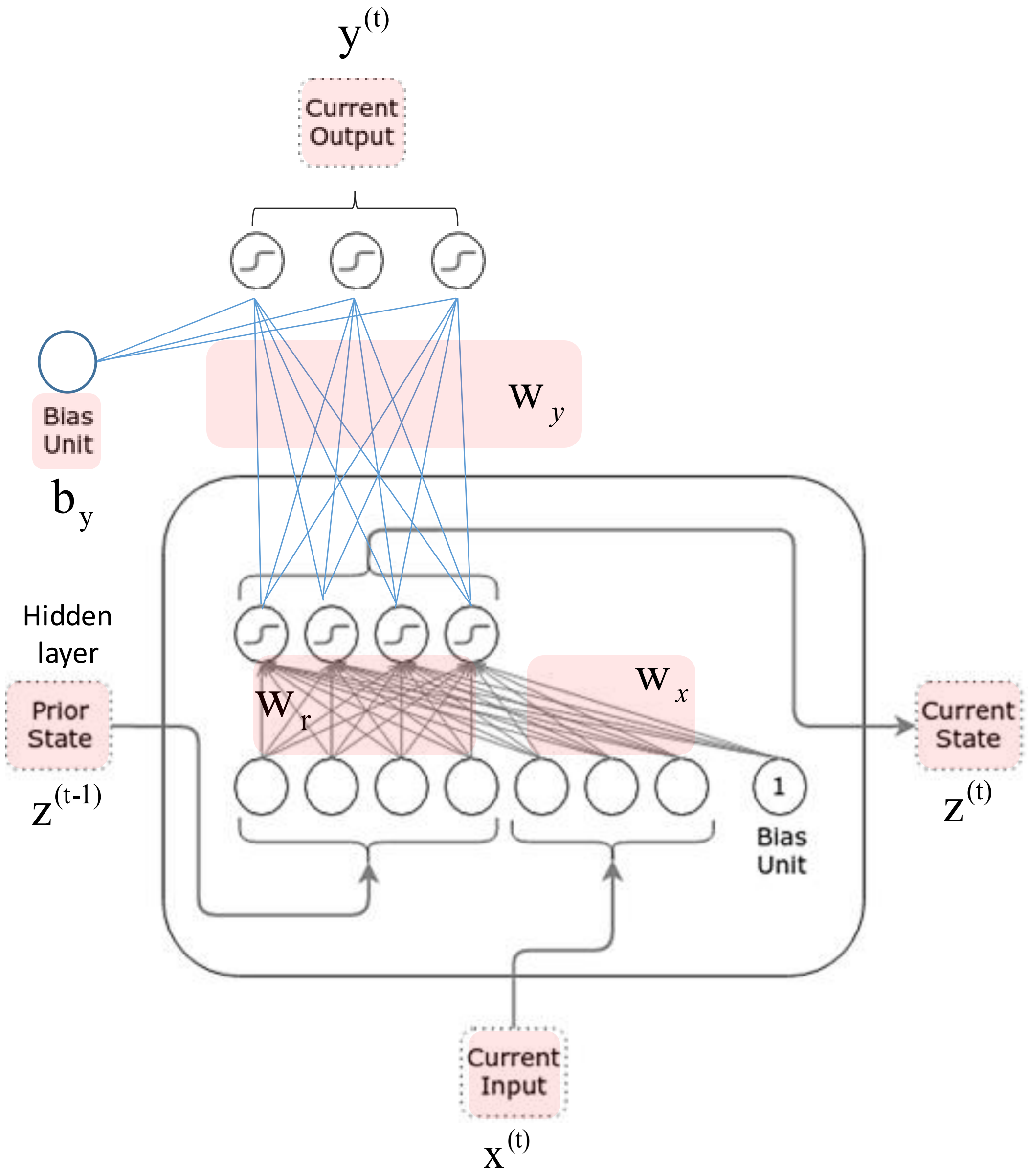
$$W_{new} = W_{old} + \boxed{\eta \frac{\partial f}{\partial w_i}} \qquad b_{new} = b_{old} + \boxed{\eta \frac{\partial f}{\partial b}}$$



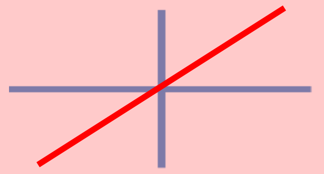
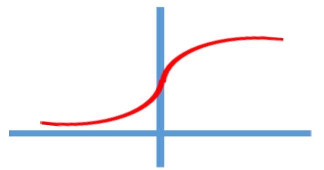
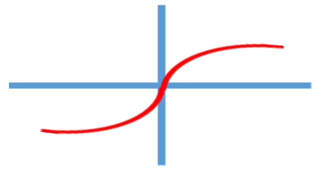
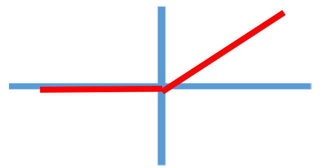
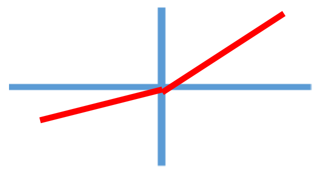
$\frac{\partial f}{\partial w_i}$ $\frac{\partial f}{\partial b}$ Direction !

Another type of neural network: Vanilla RNN





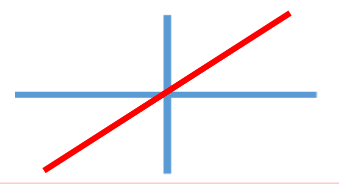
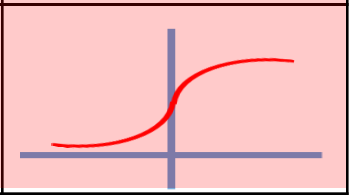
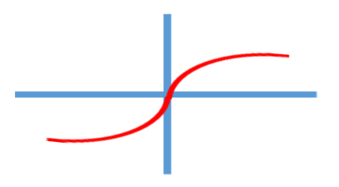
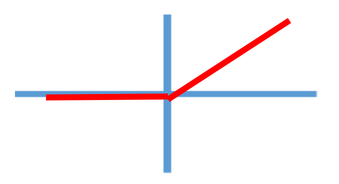
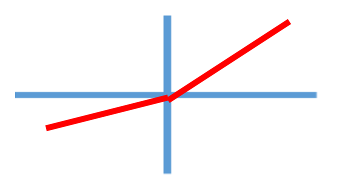
Neural networks: Activation function

Linear	
Sigmoid	
TanH	
ReLU	
Leaky ReLU	

- Cannot apply backpropagation to find how neural weights should change to reduce the error found.

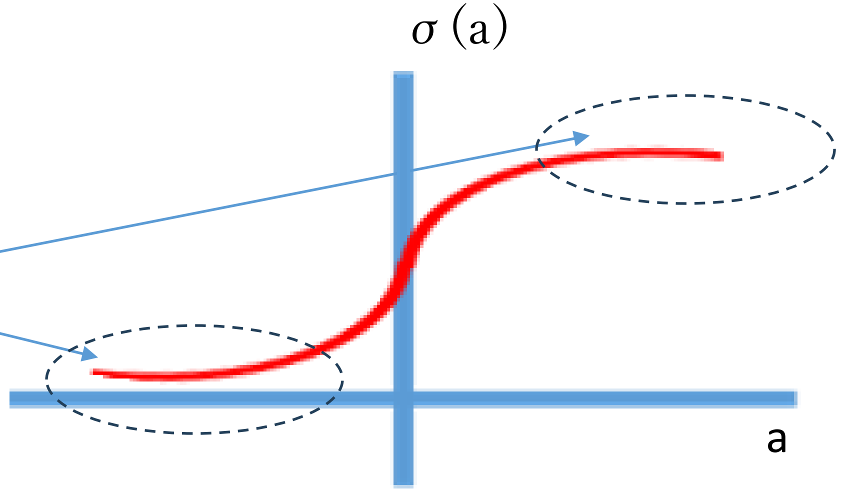
Cannot rescale the input, in other words, it gets exploded!

Neural networks: Activation function

Linear	
Sigmoid	
TanH	
ReLU	
Leaky ReLU	

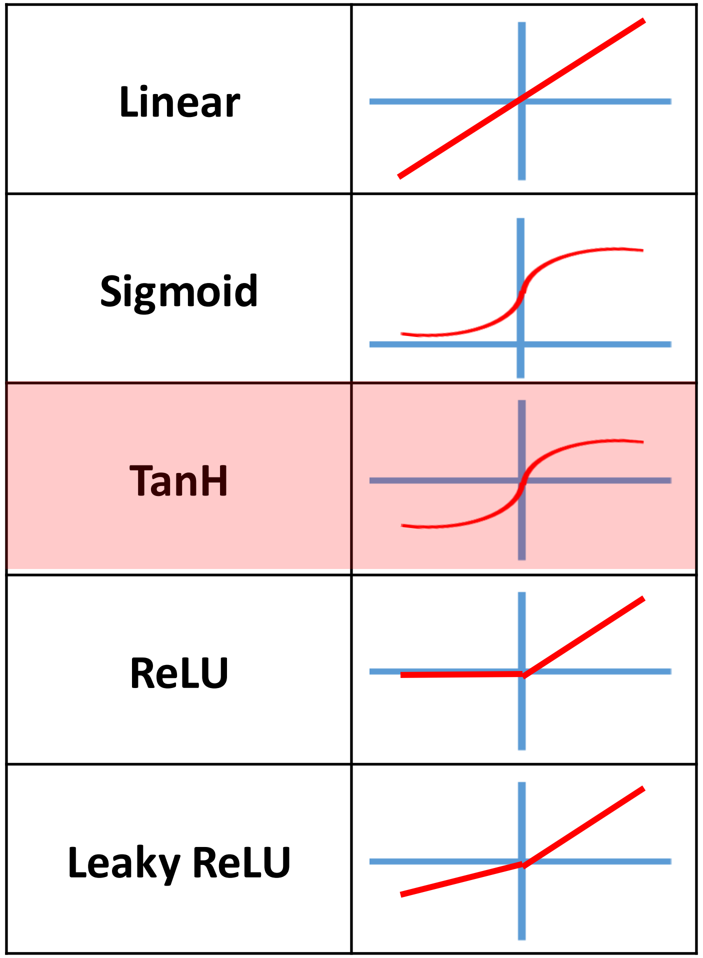
- ❑ Cannot apply backpropagation to find how neural weights should change to reduce the error found.
- ❑ Saturated neuron stops the backpropagation due to the zero gradient at both ends.
- ❑ Non-zero centered: data coming into a neuron is always positive.

$$\frac{\partial E(w)}{\partial w_1} = \frac{\partial E(\sigma(a))}{\partial \sigma} \times \boxed{\frac{\partial \sigma}{\partial a}} \times \frac{\partial a}{\partial w_1}$$



Known as "**vanishing gradient problem**"

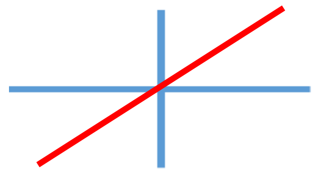
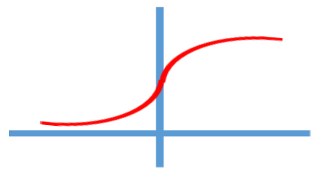
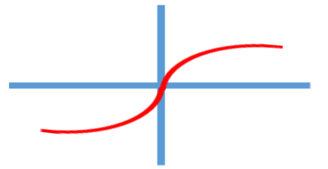
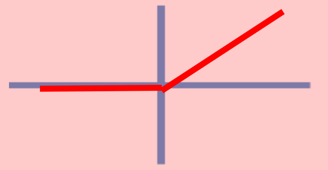
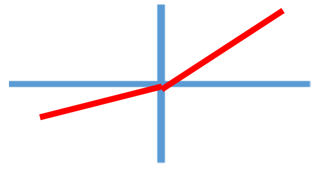
Neural networks: Activation function



- ❑ Cannot apply backpropagation to find how neural weights should change to reduce the error found.
- ❑ Saturated neuron stops the backpropagation due to the zero gradient at both ends.
- ❑ Non-zero centered: data coming into a neuron is always positive.
- ❑ Zero centered… but the computation of exp() is expensive.

$$f(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}}$$

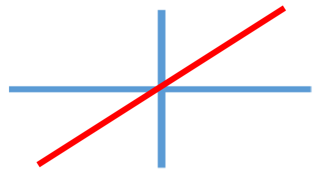
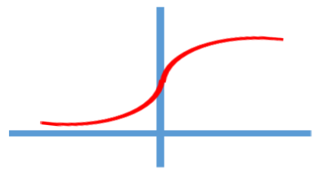
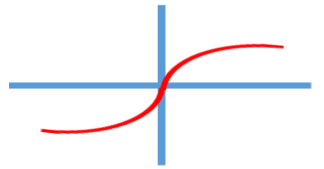
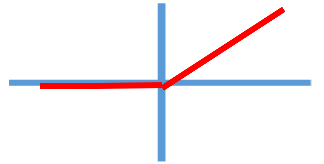
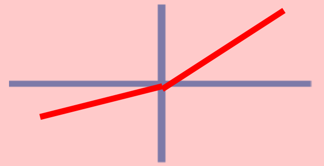
Neural networks: Activation function

Linear	
Sigmoid	
TanH	
ReLU	
Leaky ReLU	

- ❑ Cannot apply backpropagation to find how neural weights should change to reduce the error found.
- ❑ Saturated neuron stops the backpropagation due to the zero gradient at both ends.
- ❑ Non-zero centered: data coming into a neuron is always positive.
- ❑ Zero centered… but the computation of exp() is expensive.
- ❑ The convergence speed with ReLU is 6 times faster than TanH [1]

[1] ImageNet Classification with Deep Convolutional Neural Networks

Neural networks: Activation function

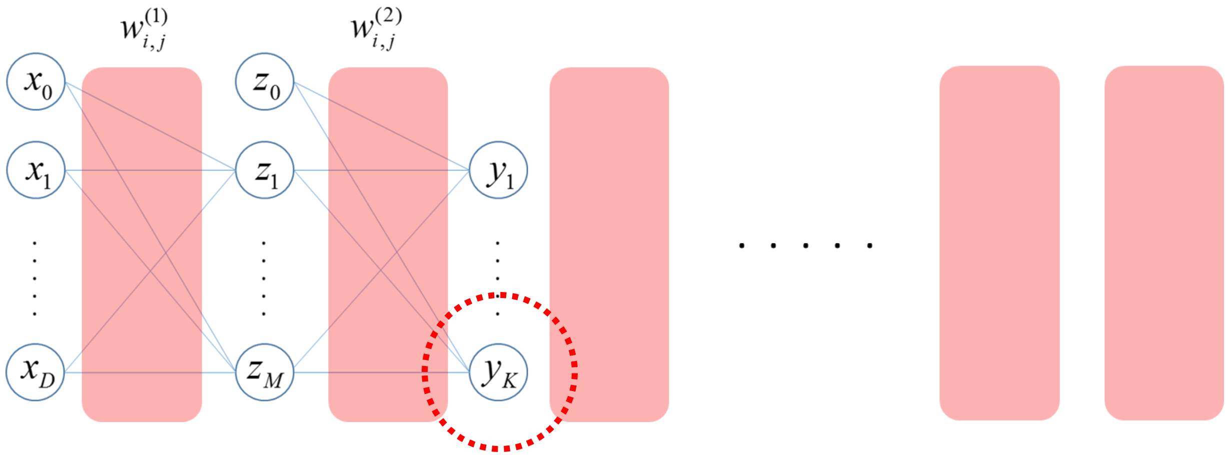
Linear	
Sigmoid	
TanH	
ReLU	
Leaky ReLU	

- ❑ Cannot apply backpropagation to find how neural weights should change to reduce the error found.
- ❑ Saturated neuron stops the backpropagation due to the zero gradient at both ends.
- ❑ Non-zero centered: data coming into a neuron is always positive.
- ❑ Zero centered… but the computation of exp() is expensive.
- ❑ The convergence speed with ReLU is 6 times faster than TanH [1]
- ❑ Zero centered and fast convergence …

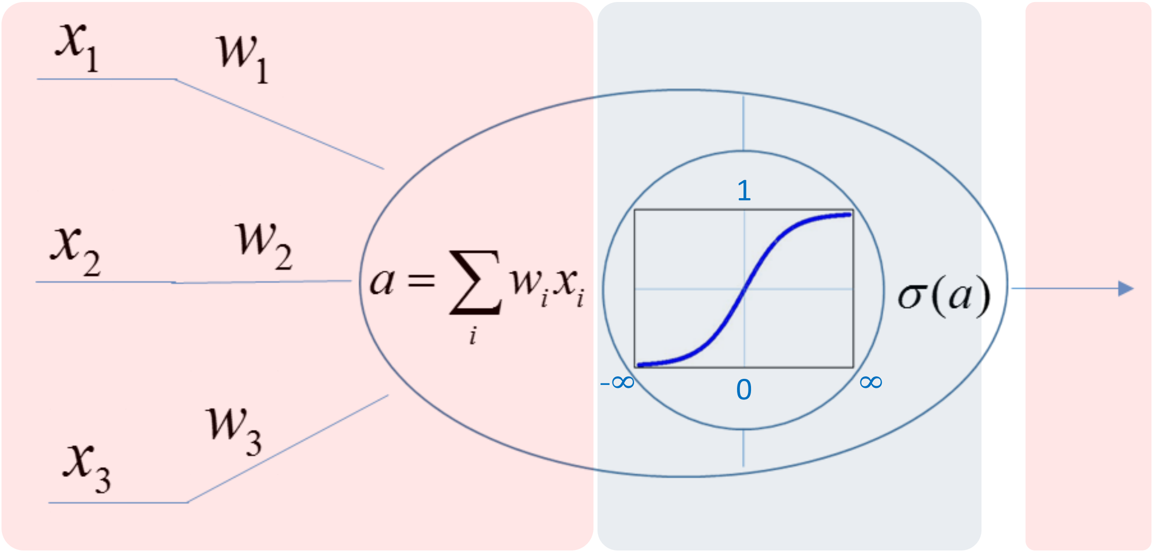
[1] ImageNet Classification with Deep Convolutional Neural Networks

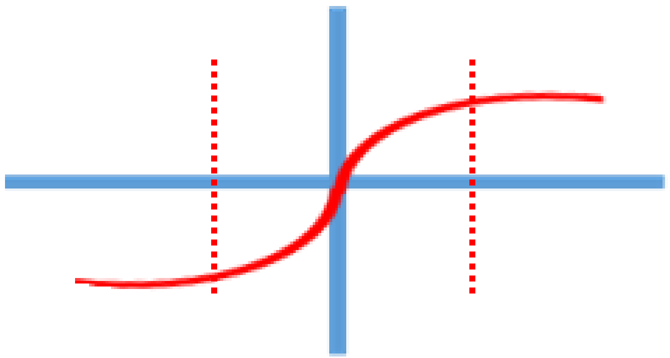
Neural networks: Weight initialization

- How do we set the weight of each link initially?



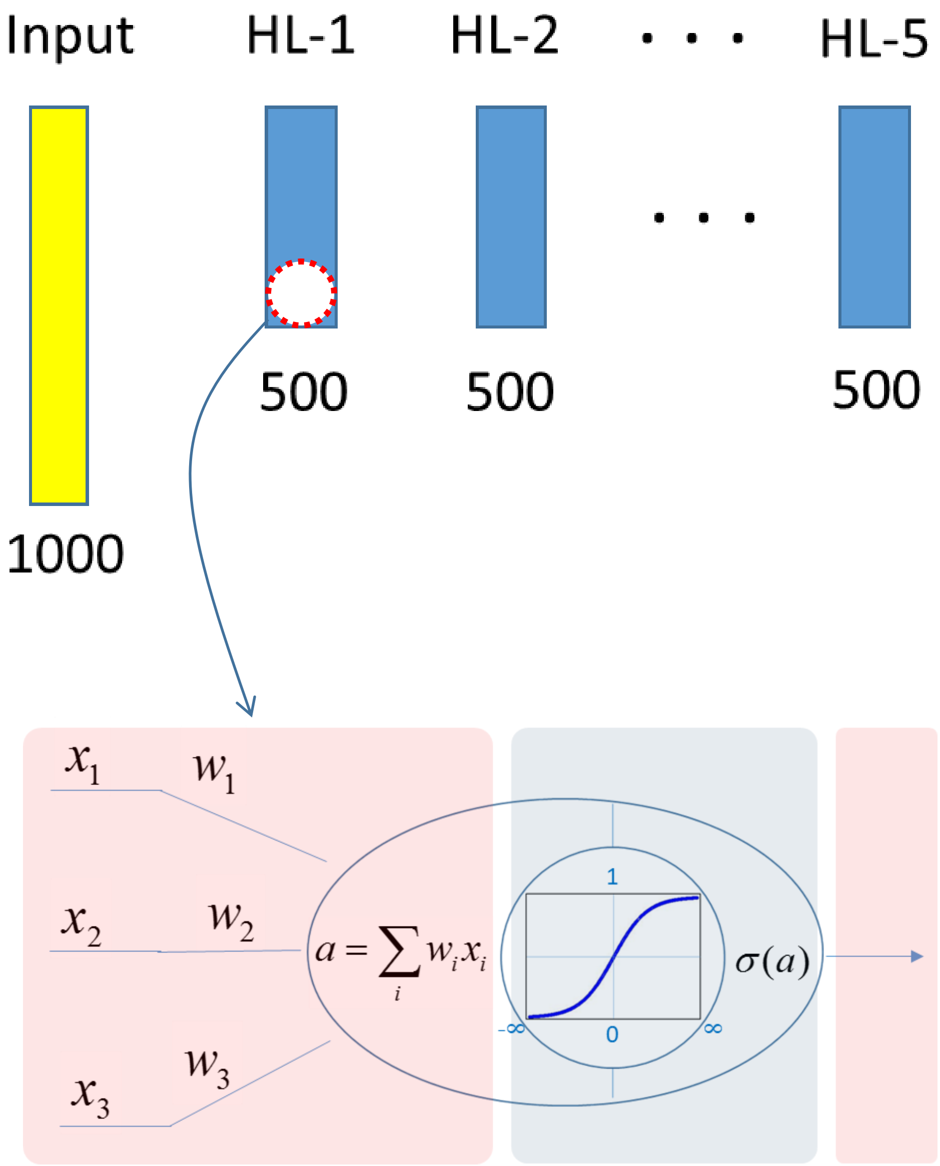
- An input to an activation function had better be within a certain range rather than either an extremely large or small value.





Neural networks: Weight initialization experiment

- A neural network is created as shown in the right, e.g., with 5 layers.
  - Each of the 1000 inputs is drawn from N(0, 1) and goes through the 5 hidden layers,
  - Then, the outputs of each hidden layer, e.g., after activation function, are plotted.



Neural networks: Weight initialization experiment

- How about random setting? **Tanh**
  - N (mean=0, std=1)

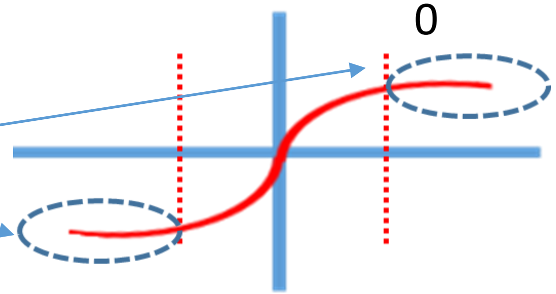
- Random setting with smaller std?
  - N (mean=0, std=0.01)

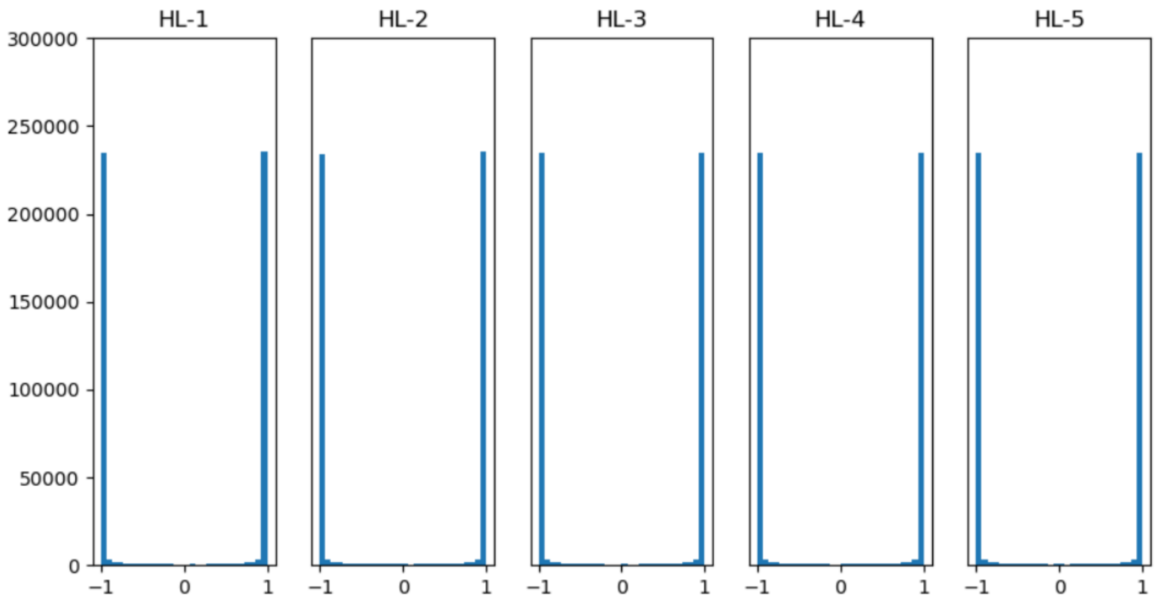
- How about <u>Xavier</u> initialization?
  - N (mean=0, std= $\sqrt{\frac{2}{fan\ in + fan\ out}}$)

- How about <u>He</u> initialization?
  - N (mean=0, std= $\sqrt{\frac{4}{fan\ in + fan\ out}}$)

- The output values of the activation functions, tanh(), in each hidden layer are mostly distributed at -1 and 1
- Vanishing gradient problem

$$\frac{\partial E(w)}{\partial w_1} = \frac{\partial E(\sigma(a))}{\partial \sigma} \times \boxed{\frac{\partial \sigma}{\partial a}} \times \frac{\partial a}{\partial w_1}$$





Neural networks: Weight initialization experiment

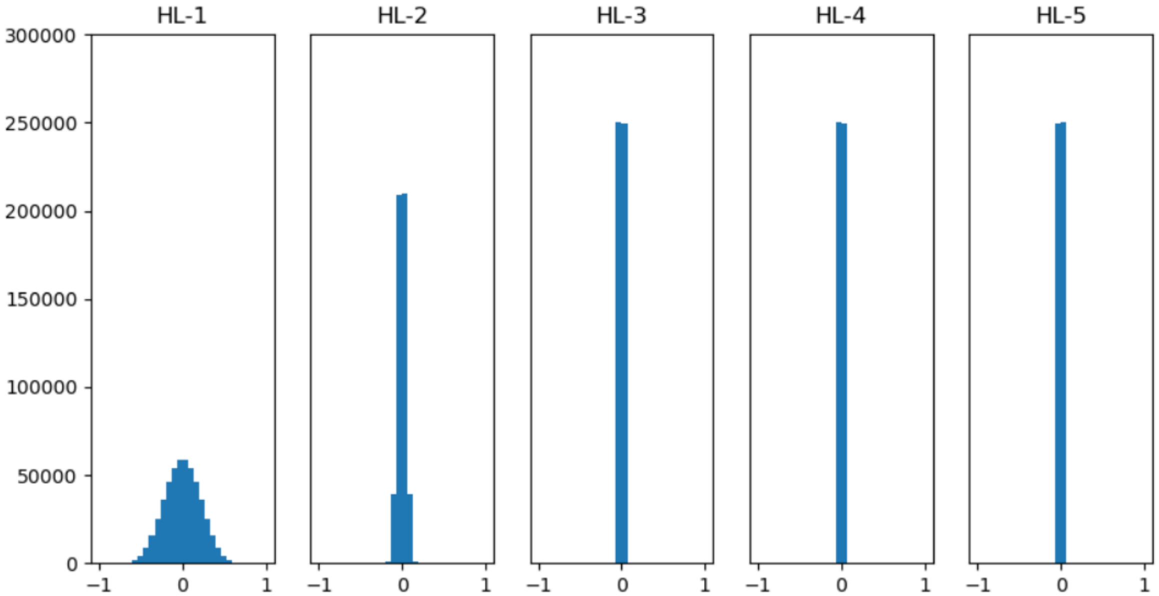
- ❑ How about random setting? **Tanh**
  - N (mean=0, std=1)

- ❑ Random setting with smaller std? **Tanh**
  - N (mean=0, std=0.01)

- ❑ How about Xavier initialization?
  - N (mean=0, std=$\sqrt{\frac{2}{fan\ in + fan\ out}}$)

- ❑ How about He initialization?
  - N (mean=0, std=$\sqrt{\frac{4}{fan\ in + fan\ out}}$)

- ❑ It solves the vanishing gradient problem but each weight tends to have same value,
- ❑ which implies some learning problem.



Neural networks: Weight initialization experiment

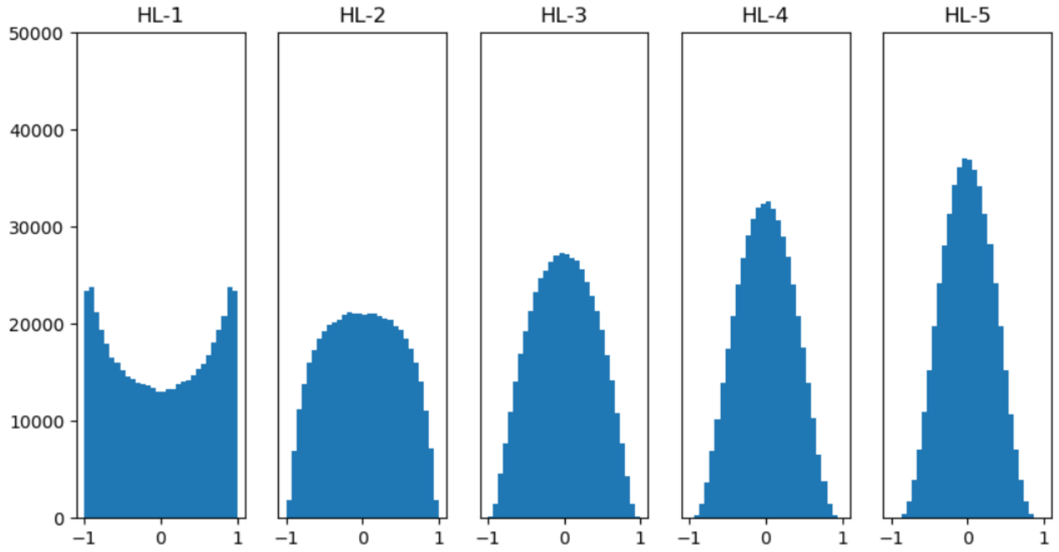
- How about random setting? **Tanh**
  - N (mean=0, std=1)

- Random setting with smaller std? **Tanh**
  - N (mean=0, std=0.01)

- How about Xavier initialization? **Tanh**
  - N (mean=0, std= $\sqrt{\frac{2}{fan\ in + fan\ out}}$)

- How about He initialization?
  - N (mean=0, std= $\sqrt{\frac{4}{fan\ in + fan\ out}}$)

- If S-curve function, e.g., sigmoid or tanh, is used as an activation function, Xavier is a way to initialize weight
- Solving the vanishing gradient and learning issue shown previously, e.g., well distributed
- STD is a function of the number of neurons in each hidden layer



Neural networks: Weight initialization experiment

- How about random setting? **Tanh**
  - N (mean=0, std=1)

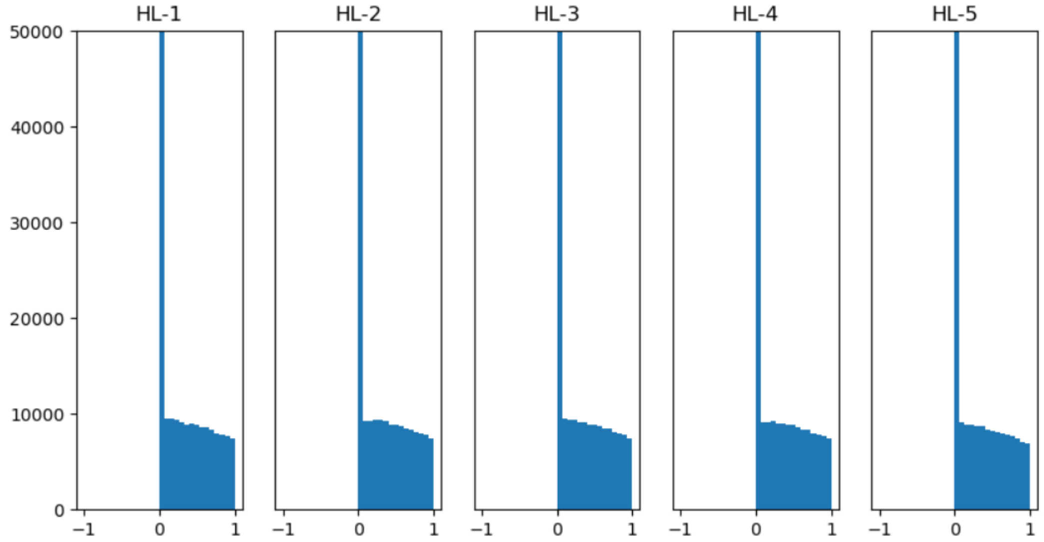
- Random setting with smaller std? **Tanh**
  - N (mean=0, std=0.01)

- How about Xavier initialization? **Tanh**
  - N (mean=0, std=$\sqrt{\frac{2}{fan\ in + fan\ out}}$)

- How about He initialization? **Relu**
  - N (mean=0, std=$\sqrt{\frac{4}{fan\ in + fan\ out}}$)

- As mentioned previously, Relu is 6 times faster than s-curve function.

- "He" is a choice for weight initialization when Relu is used as an activation function.



Summary

- A deep neural network is a class of neural networks inspired by the human brain's structure and functioning.
- We call a deep neural network as modern style machine learning because it can be operable now due to the abundant data, and powerful machines, etc.
- The backpropagation algorithm of neural networks was explained.
- Several design issues of neural networks such as activation functions, initial link weight setting, were explored.