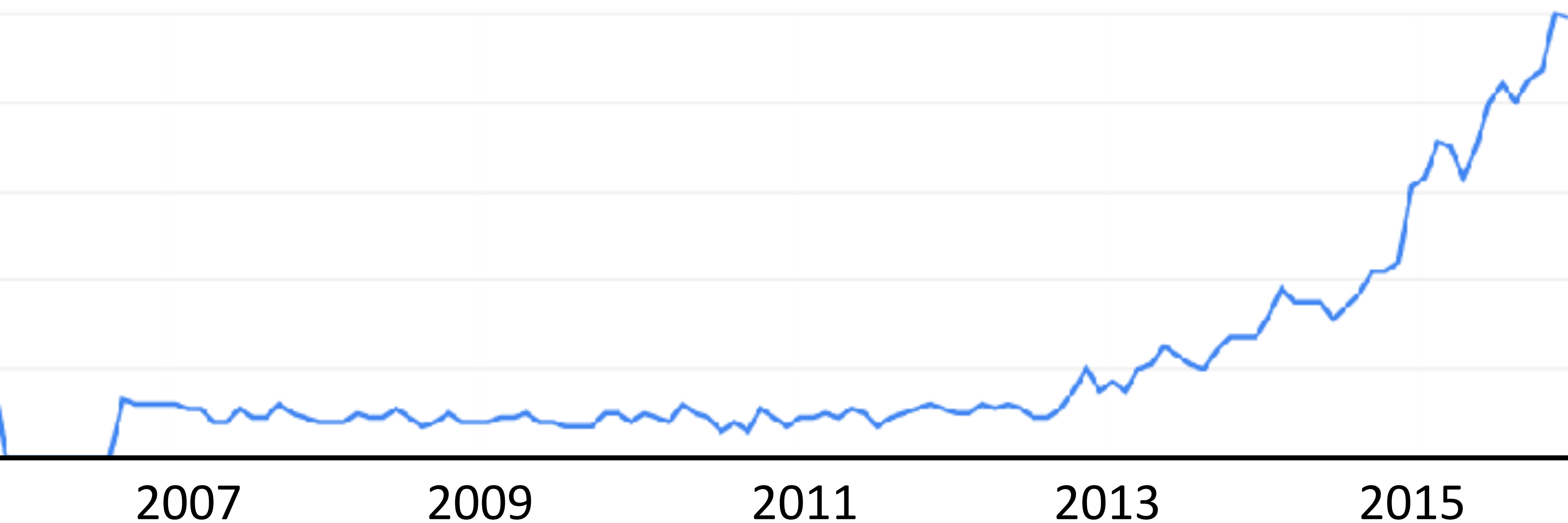
Deep Learning Tutorial

Courtesy of Hung-yi Lee

Deep learning attracts lots of attention.

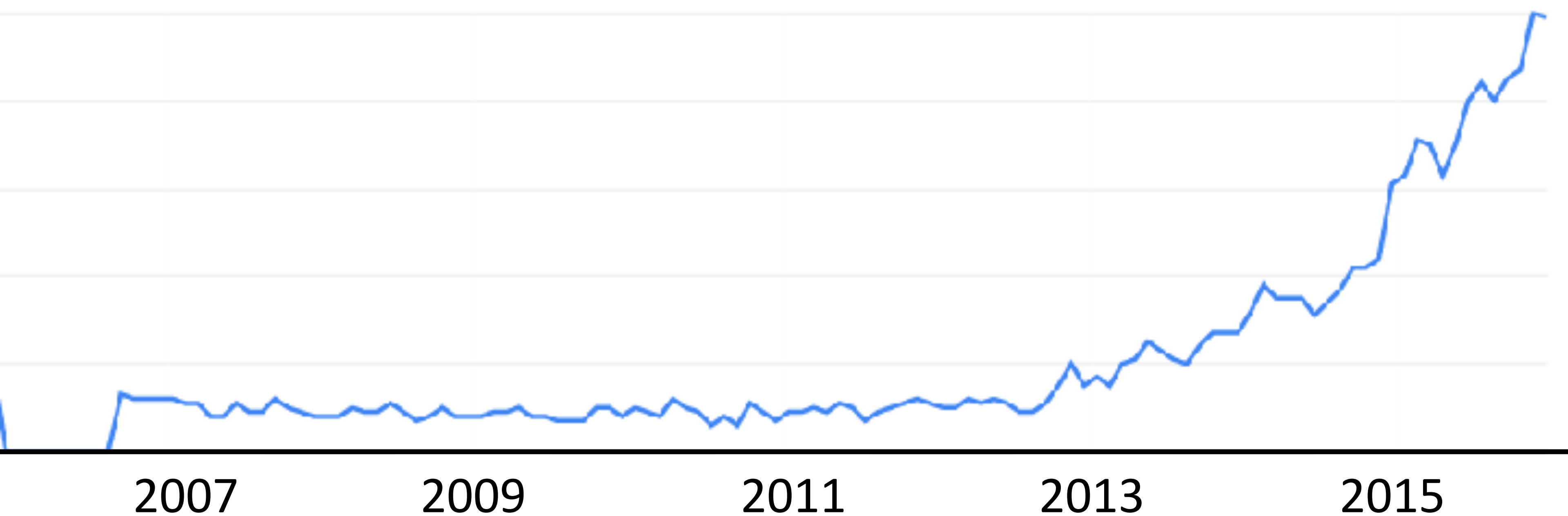
- Google Trends



Deep learning attracts lots of attention.

- Google Trends

Deep learning obtains many exciting results.



Outline

- Part I: Introduction of Deep Learning
- Part II: Why Deep?
- Part III: Tips for Training Deep Neural Network

Part I:
Introduction of Deep Learning

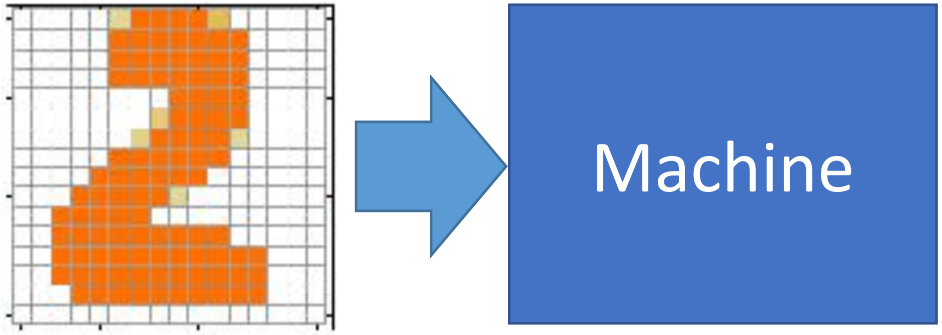
What people already knew in 1980s

Example Application

- Handwriting Digit Recognition

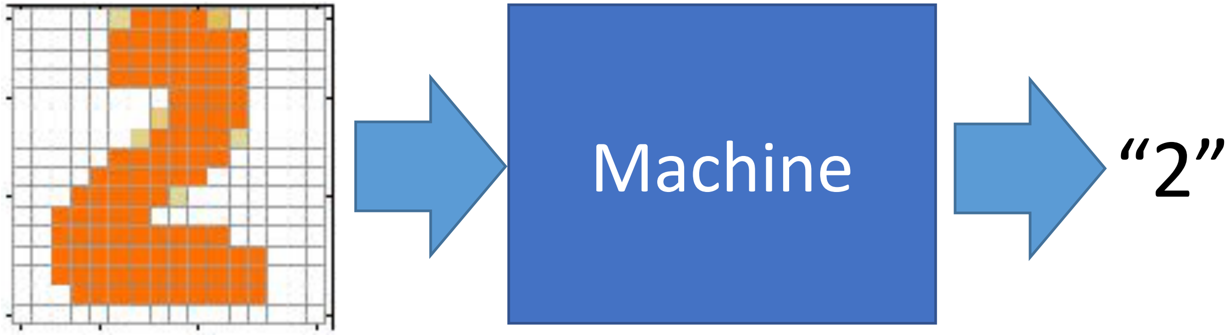
Example Application

- Handwriting Digit Recognition



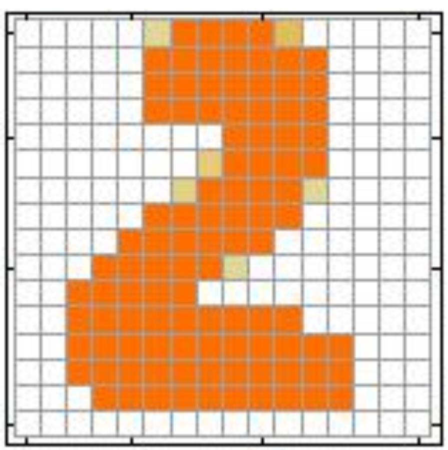
Example Application

- Handwriting Digit Recognition



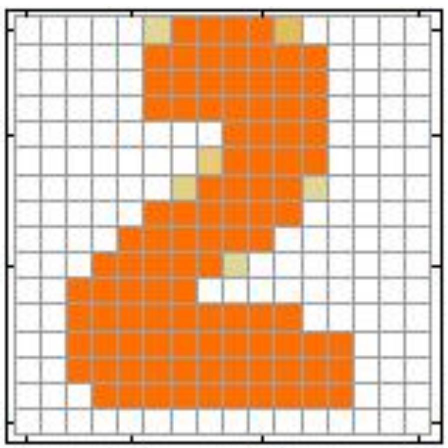
Handwriting Digit Recognition

Input **Output**



Handwriting Digit Recognition

Input



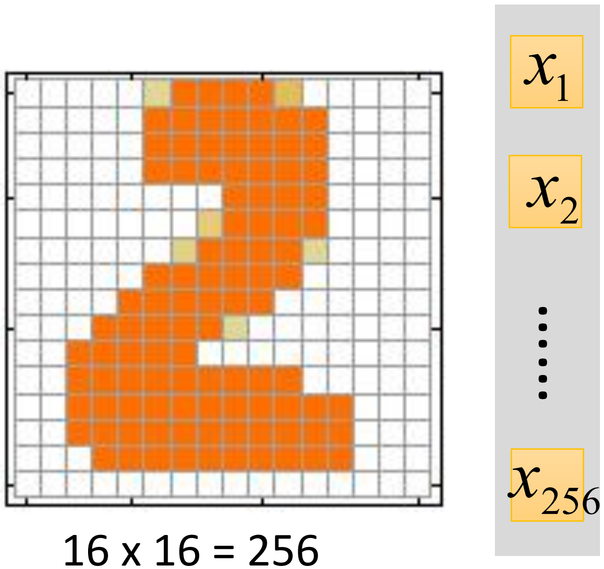
16 x 16 = 256

Output

Handwriting Digit Recognition

Input

Output

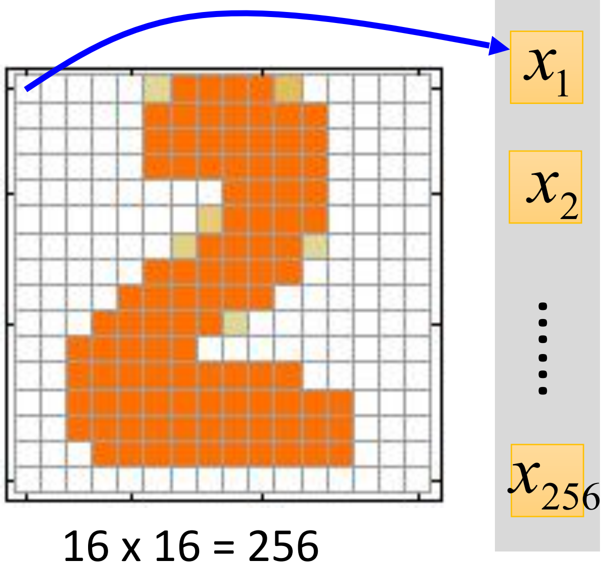


16 x 16 = 256

Handwriting Digit Recognition

Input

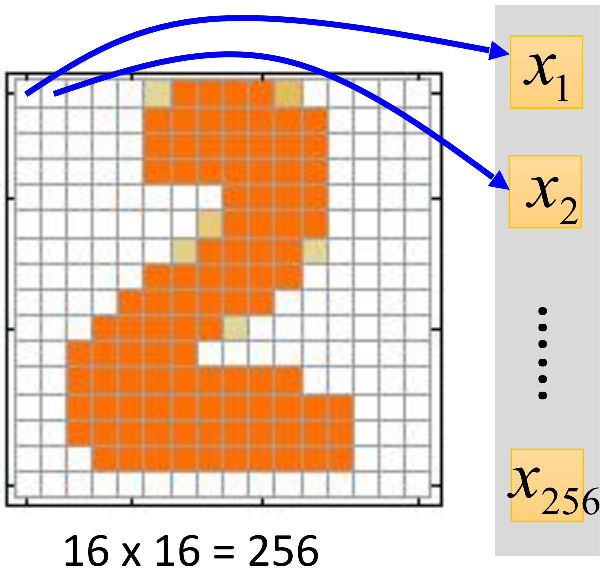
Output



Handwriting Digit Recognition

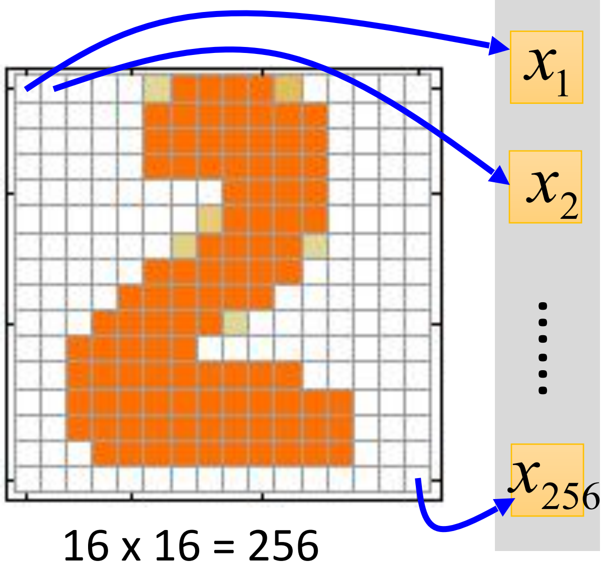
Input

Output



Handwriting Digit Recognition

Input

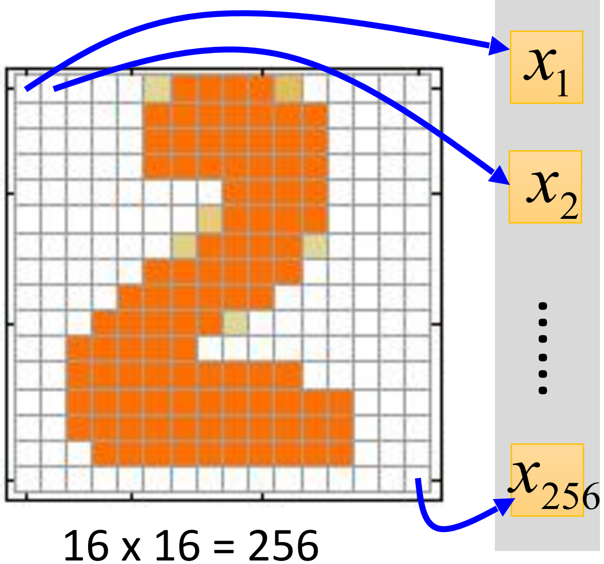


16 x 16 = 256

Output

Handwriting Digit Recognition

Input



16 x 16 = 256

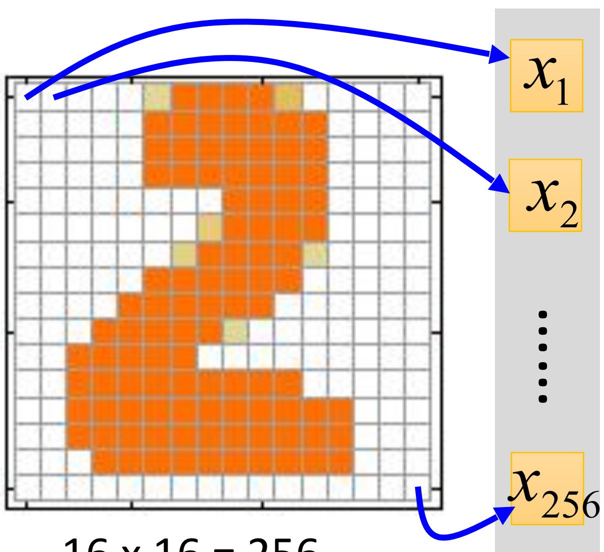
Ink → 1

No ink → 0

Output

Handwriting Digit Recognition

Input

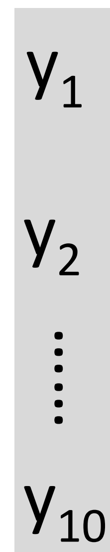


16 x 16 = 256

Ink → 1

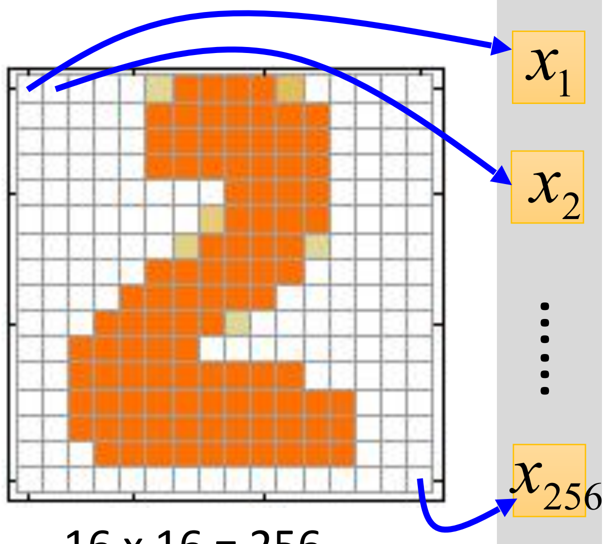
No ink → 0

Output



Handwriting Digit Recognition

Input



16 x 16 = 256

Ink → 1

No ink → 0

Output

y_1

y_2

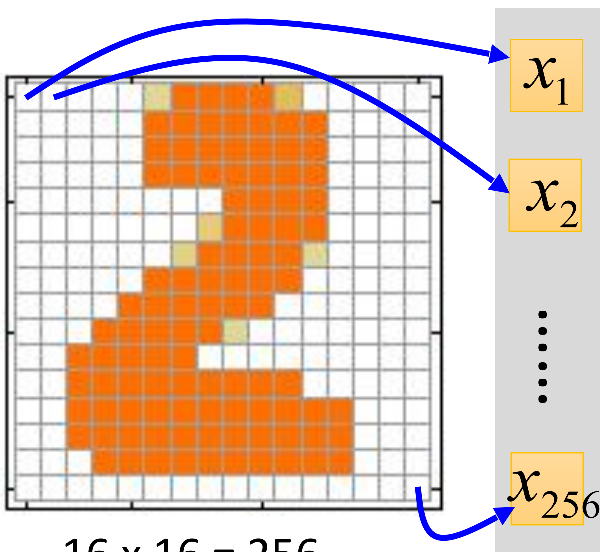
$\vdots$

y_{10}

Each dimension represents the confidence of a digit.

Handwriting Digit Recognition

Input

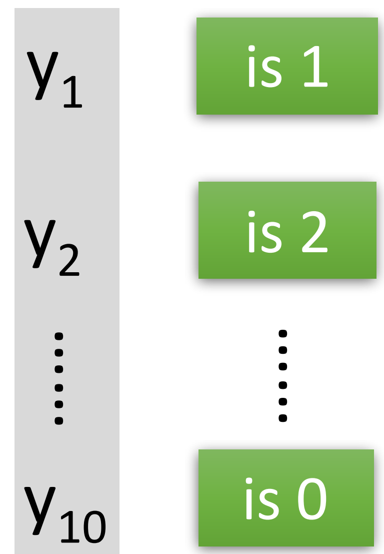


16 x 16 = 256

Ink → 1

No ink → 0

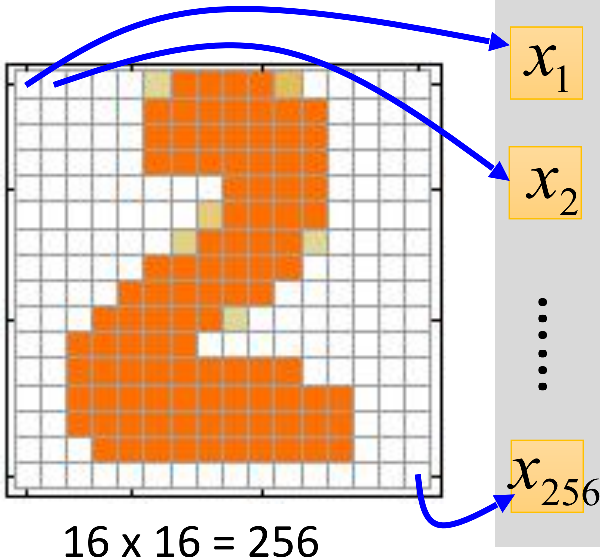
Output



Each dimension represents the confidence of a digit.

Handwriting Digit Recognition

Input

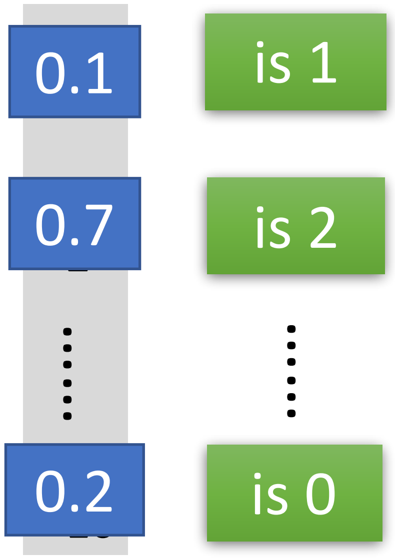


16 x 16 = 256

Ink → 1

No ink → 0

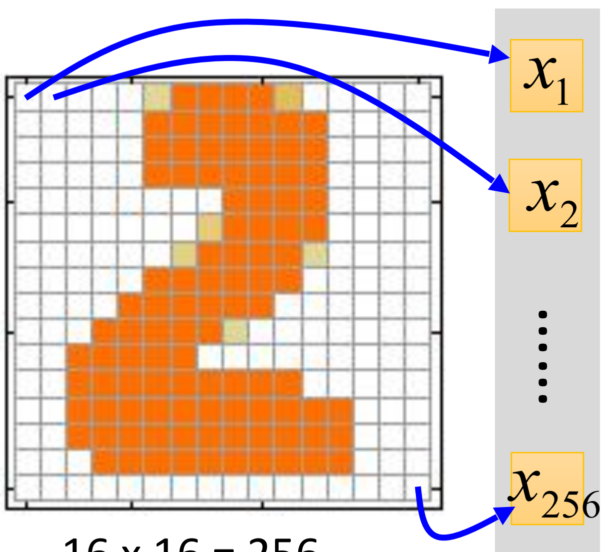
Output



Each dimension represents the confidence of a digit.

Handwriting Digit Recognition

Input

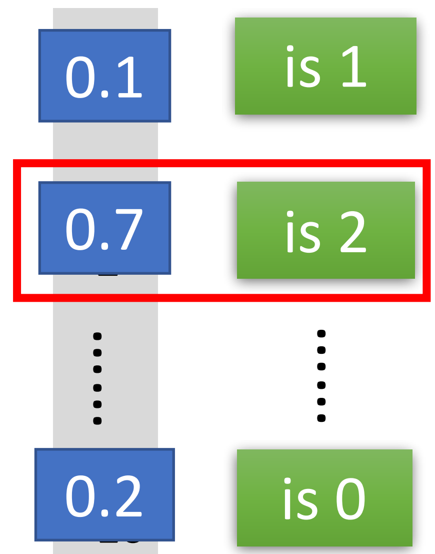


16 x 16 = 256

Ink → 1

No ink → 0

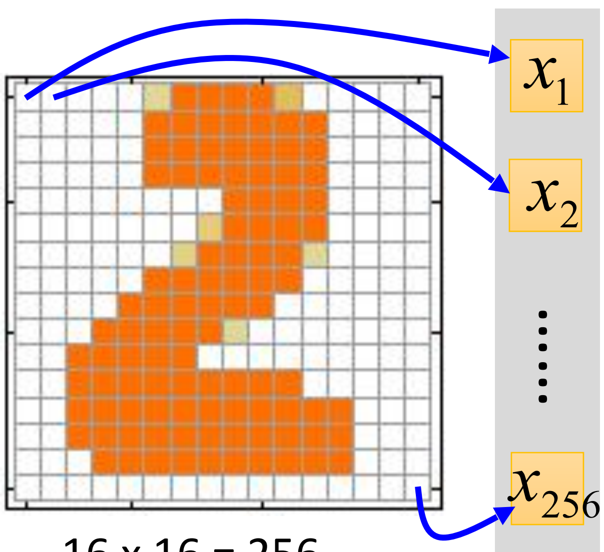
Output



Each dimension represents the confidence of a digit.

Handwriting Digit Recognition

Input

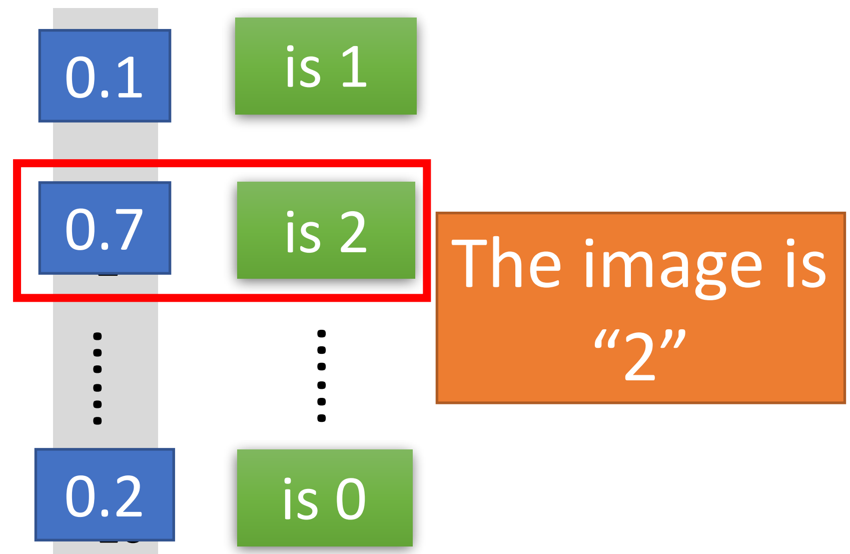


16 x 16 = 256

Ink → 1

No ink → 0

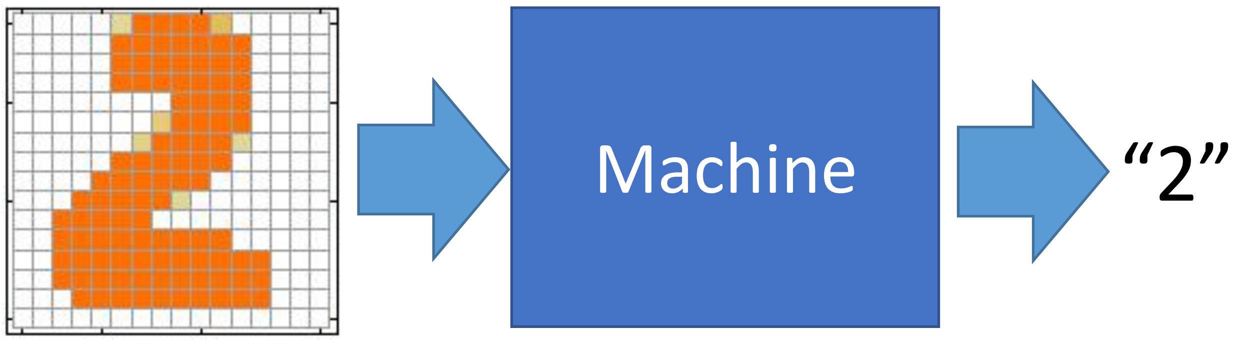
Output



Each dimension represents the confidence of a digit.

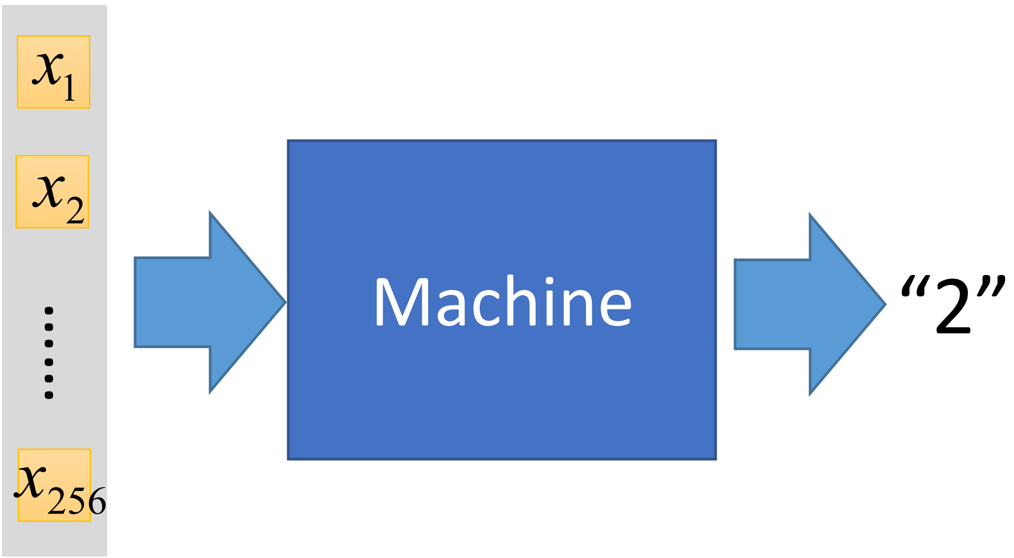
Example Application

- Handwriting Digit Recognition



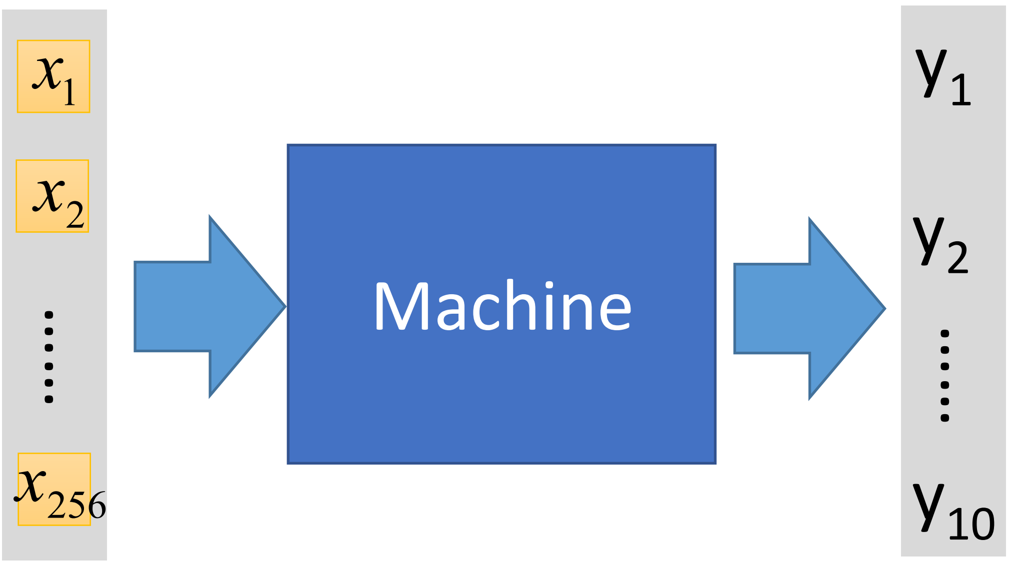
Example Application

- Handwriting Digit Recognition



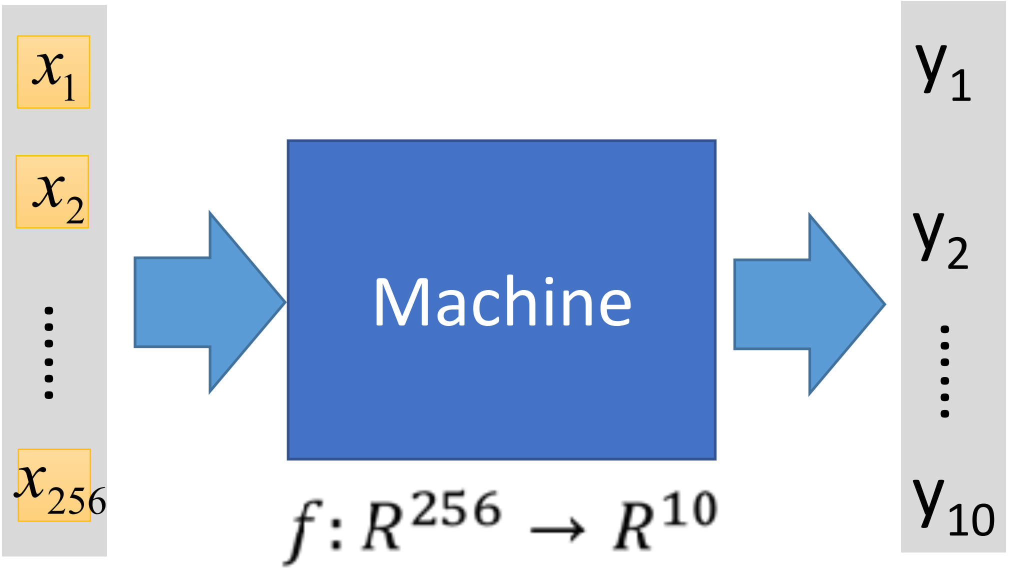
Example Application

- Handwriting Digit Recognition



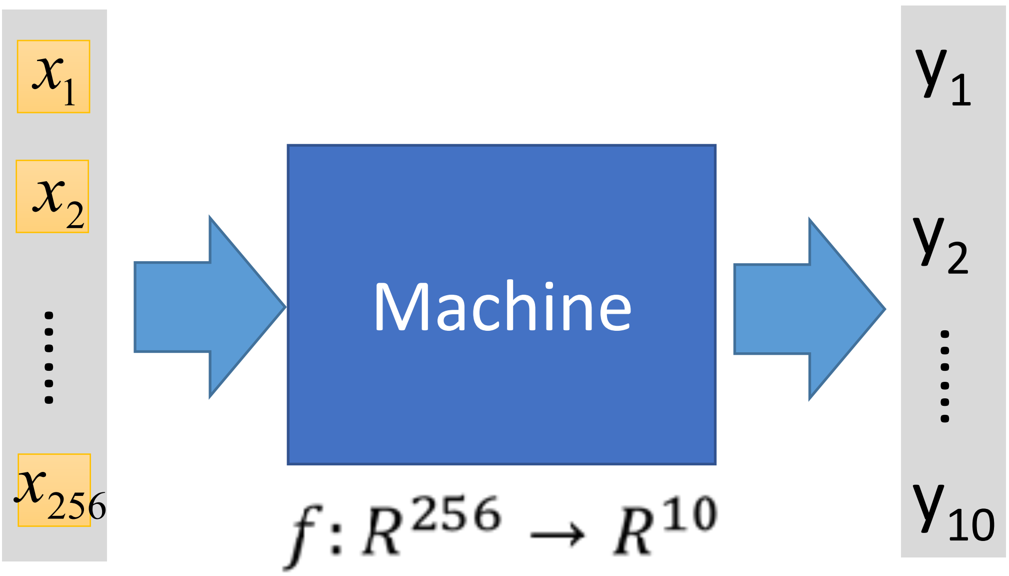
Example Application

- Handwriting Digit Recognition



Example Application

- Handwriting Digit Recognition



In deep learning, the function f is represented by neural network

Element of Neural Network

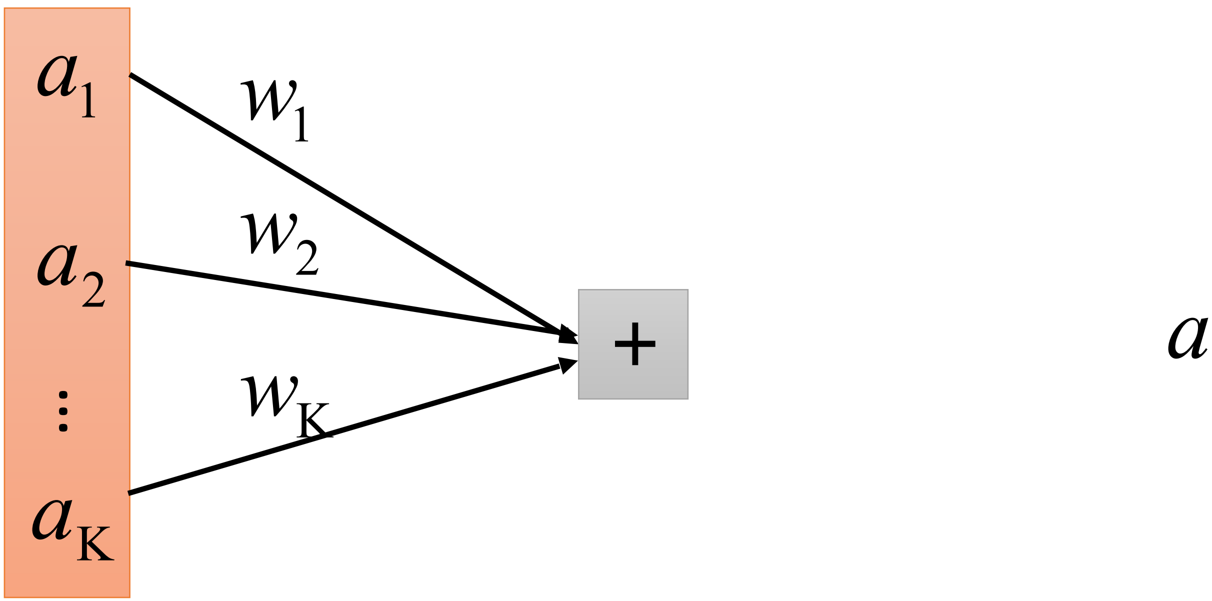
Neuron $f: R^K \rightarrow R$

$$\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_K \end{bmatrix}$$

a

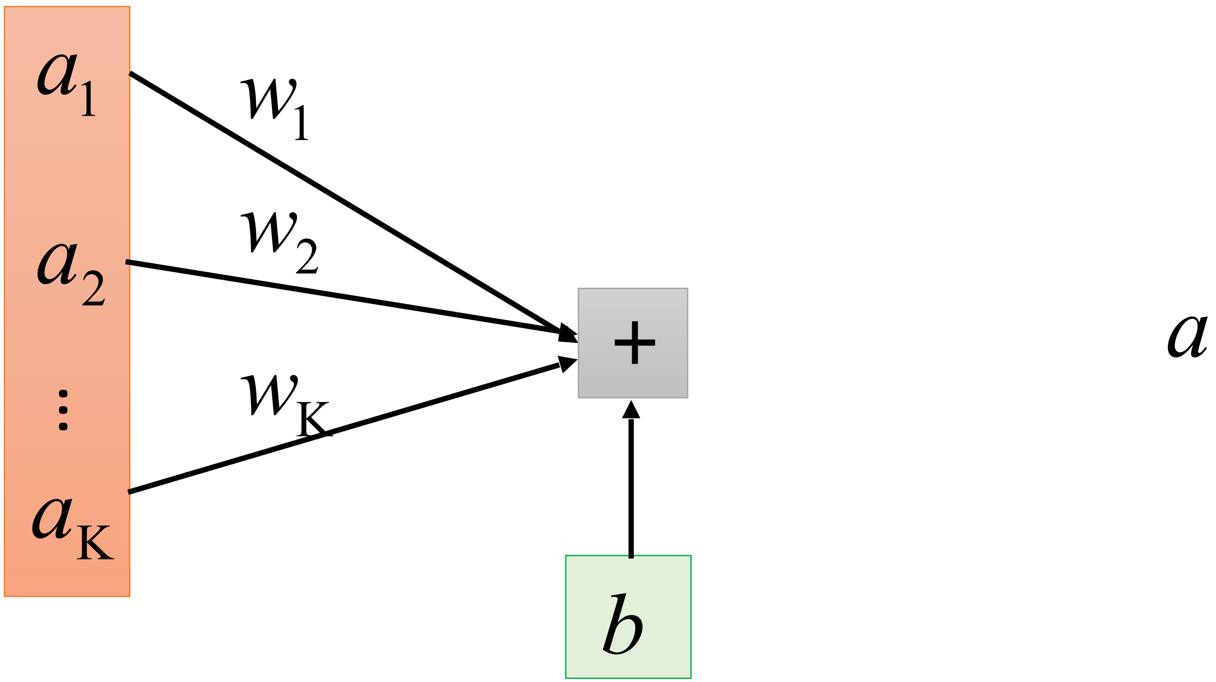
Element of Neural Network

Neuron $f: R^K \rightarrow R$



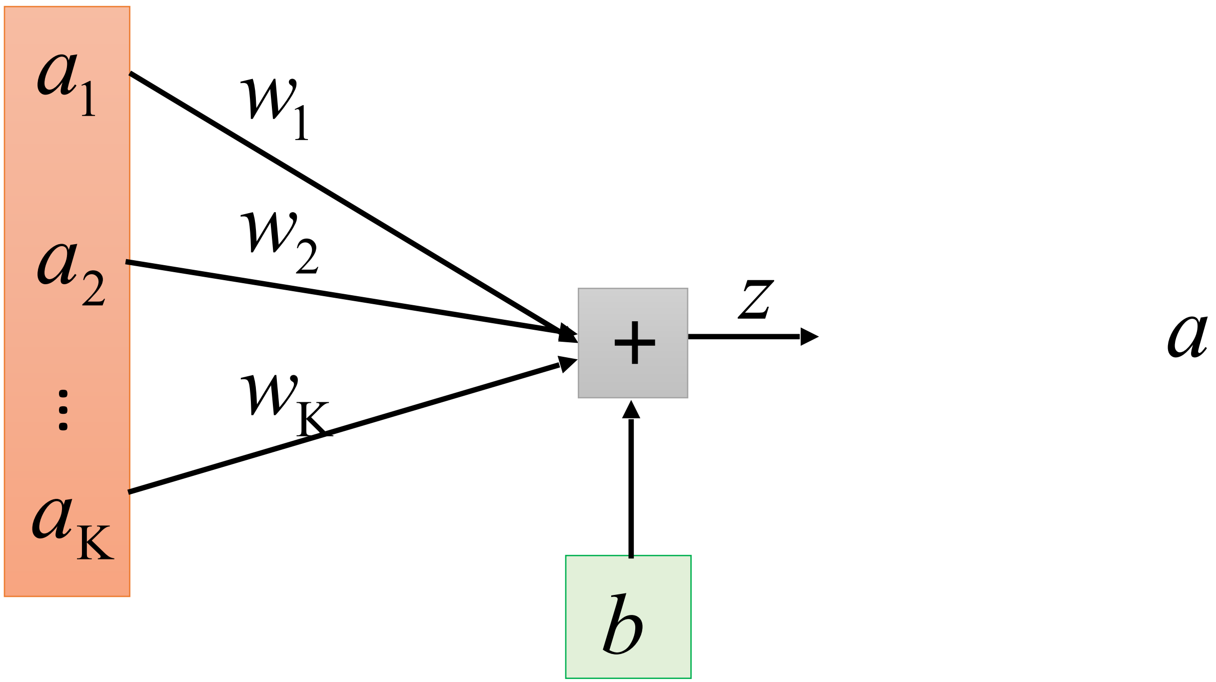
Element of Neural Network

Neuron $f: R^K \rightarrow R$



Element of Neural Network

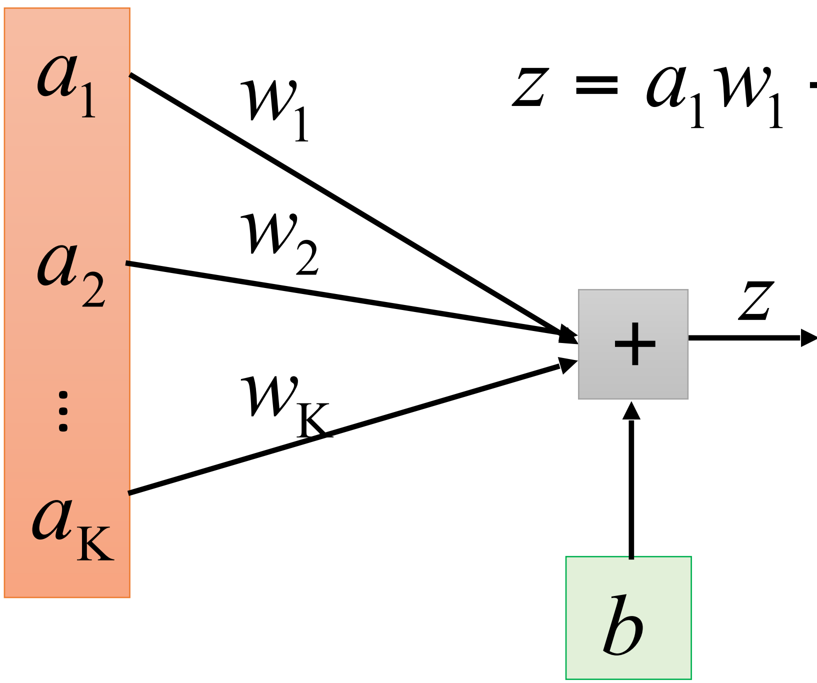
Neuron $f: R^K \rightarrow R$



Element of Neural Network

Neuron $f: R^K \rightarrow R$

$$z = a_1 w_1 + a_2 w_2 + \cdots + a_K w_K + b$$

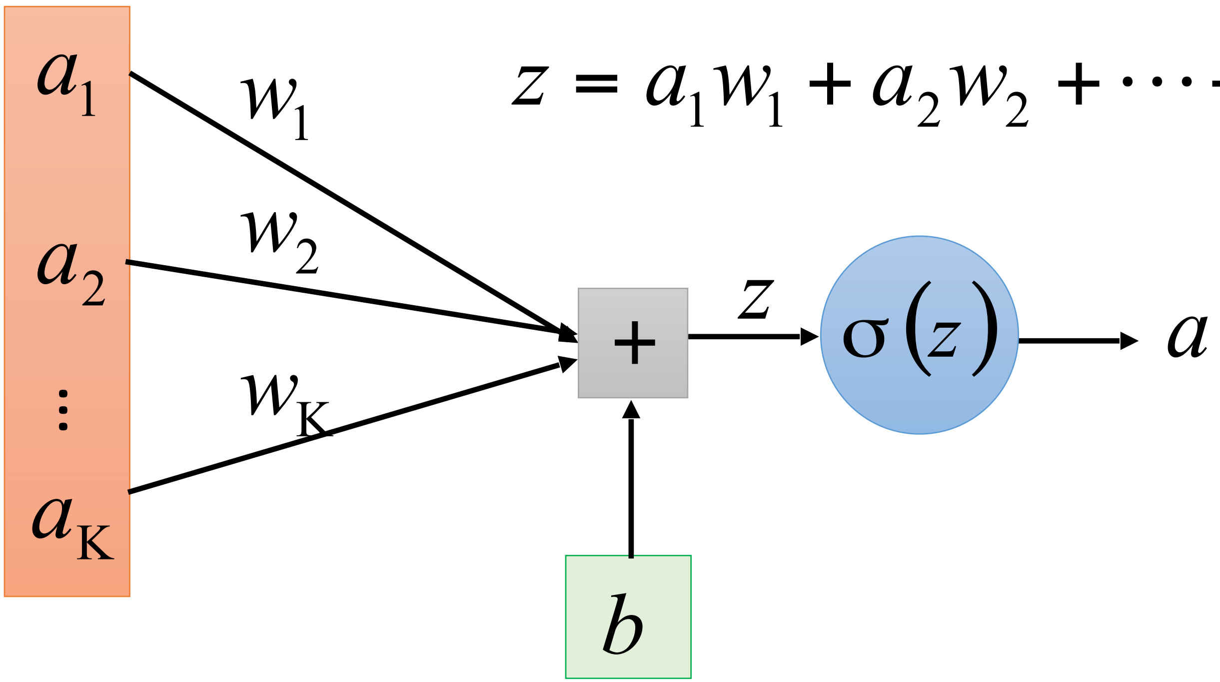


a

Element of Neural Network

Neuron $f: R^K \rightarrow R$

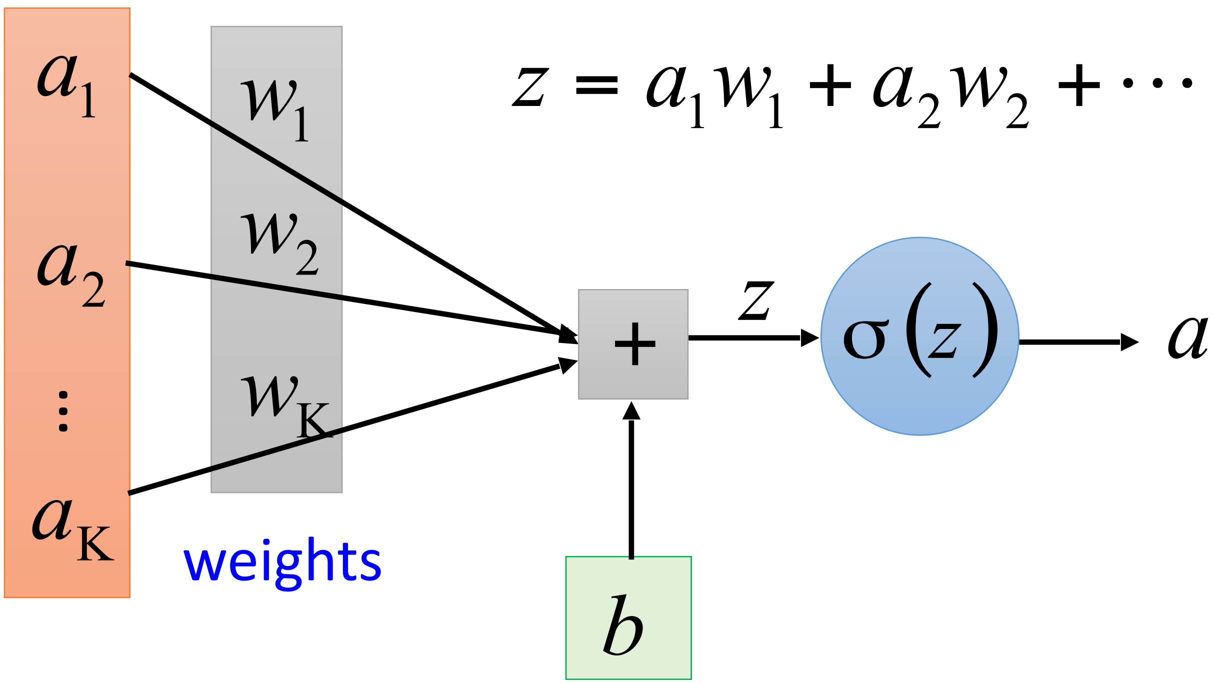
$$z = a_1 w_1 + a_2 w_2 + \cdots + a_K w_K + b$$



Element of Neural Network

Neuron $f: R^K \rightarrow R$

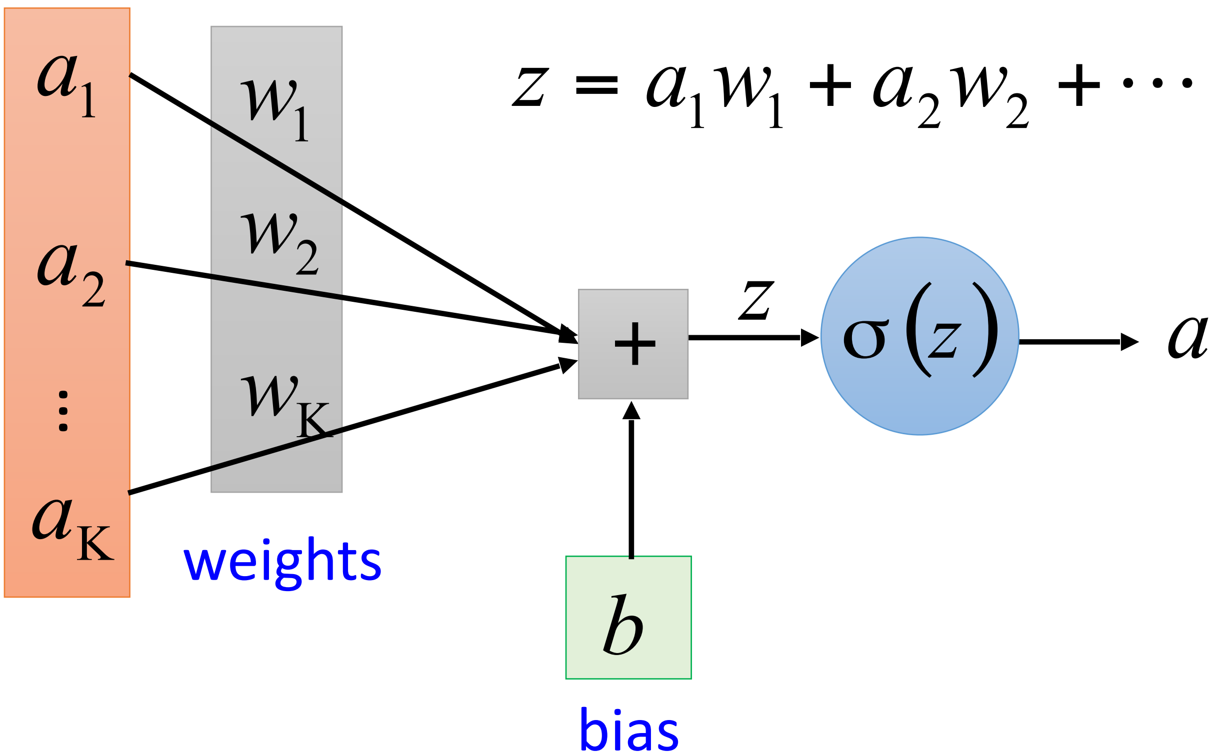
$$z = a_1 w_1 + a_2 w_2 + \cdots + a_K w_K + b$$



Element of Neural Network

Neuron $f: R^K \rightarrow R$

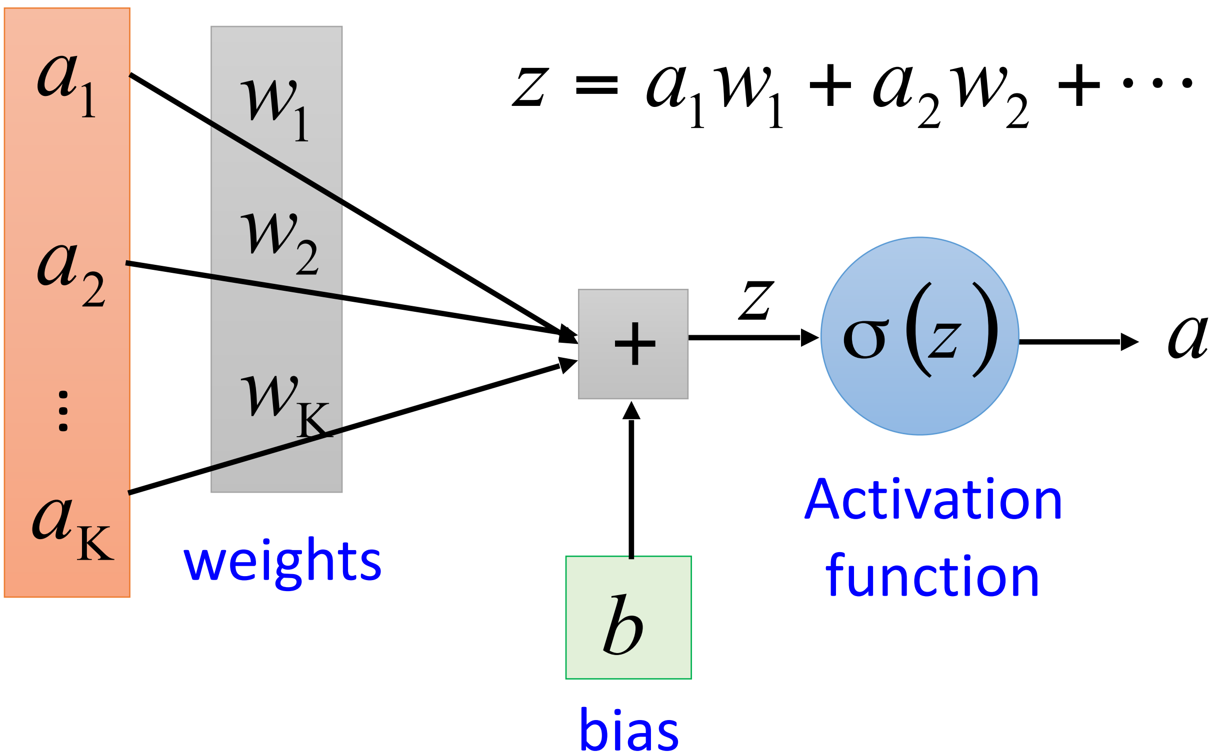
$$z = a_1 w_1 + a_2 w_2 + \cdots + a_K w_K + b$$



Element of Neural Network

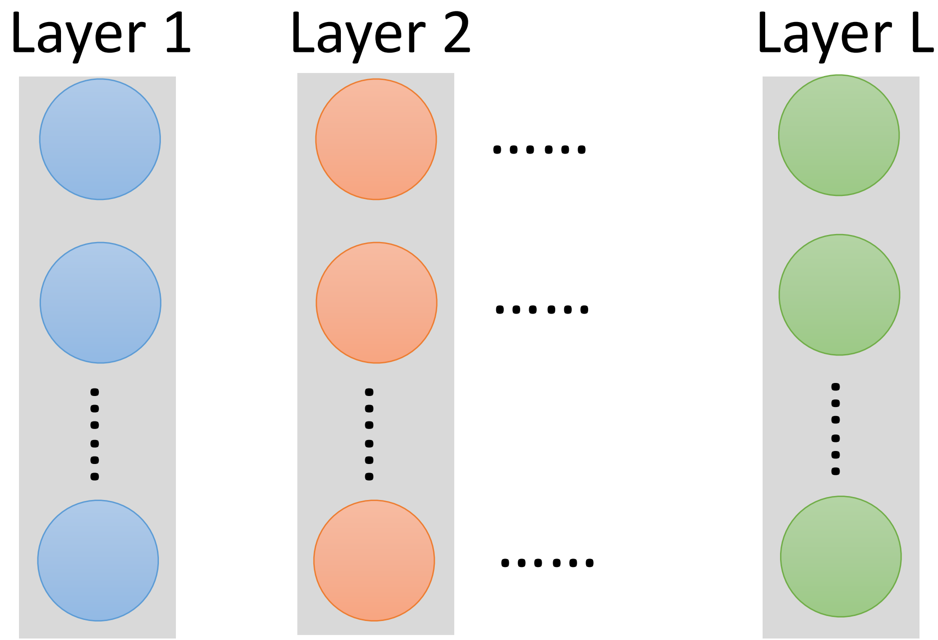
Neuron $f: R^K \rightarrow R$

$$z = a_1 w_1 + a_2 w_2 + \cdots + a_K w_K + b$$

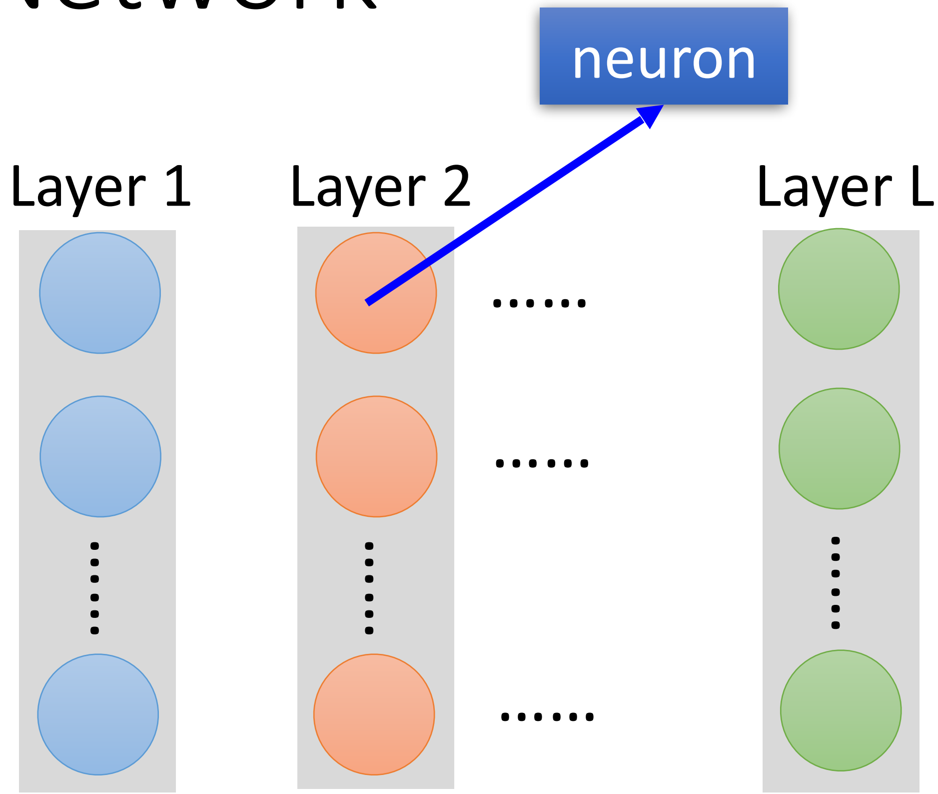


Neural Network

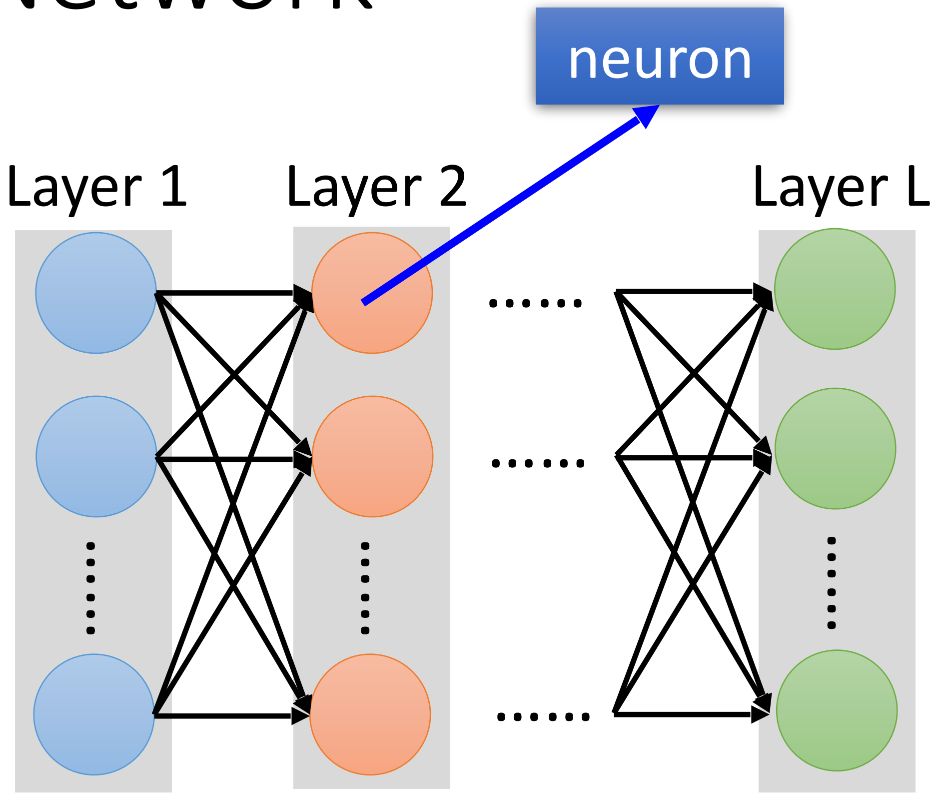
Neural Network



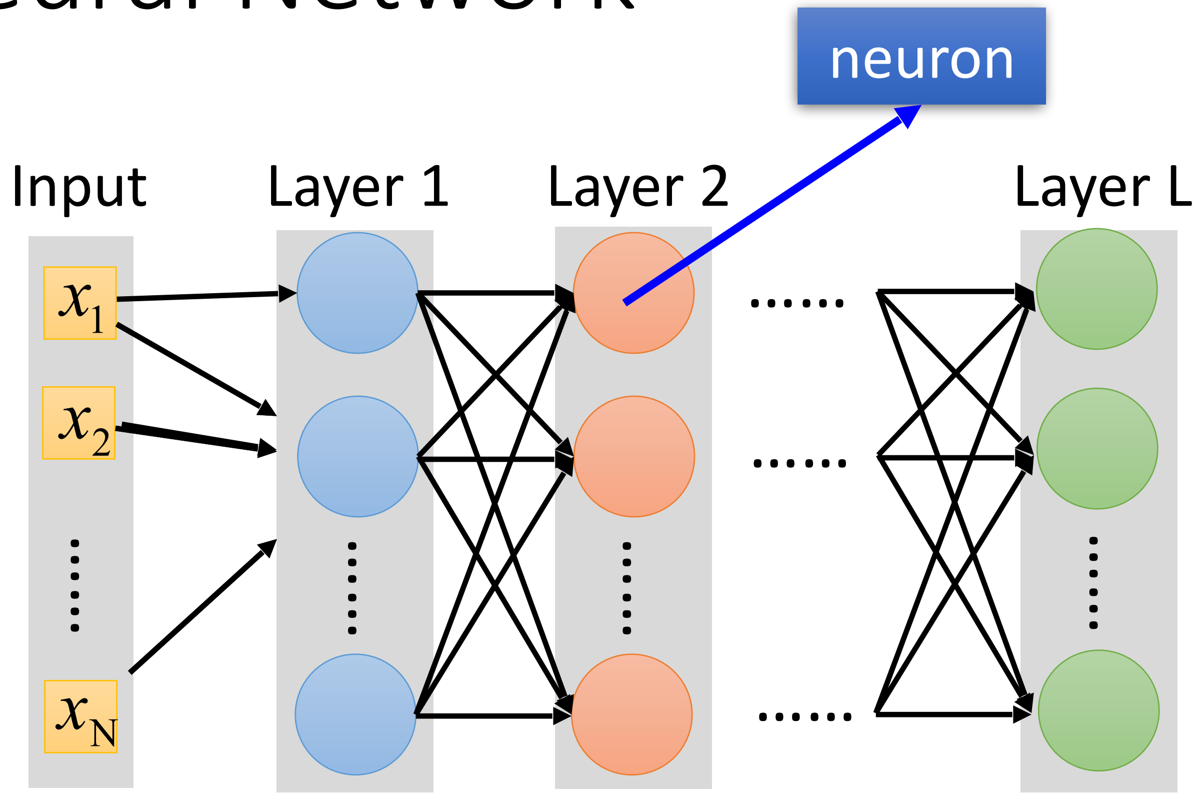
Neural Network



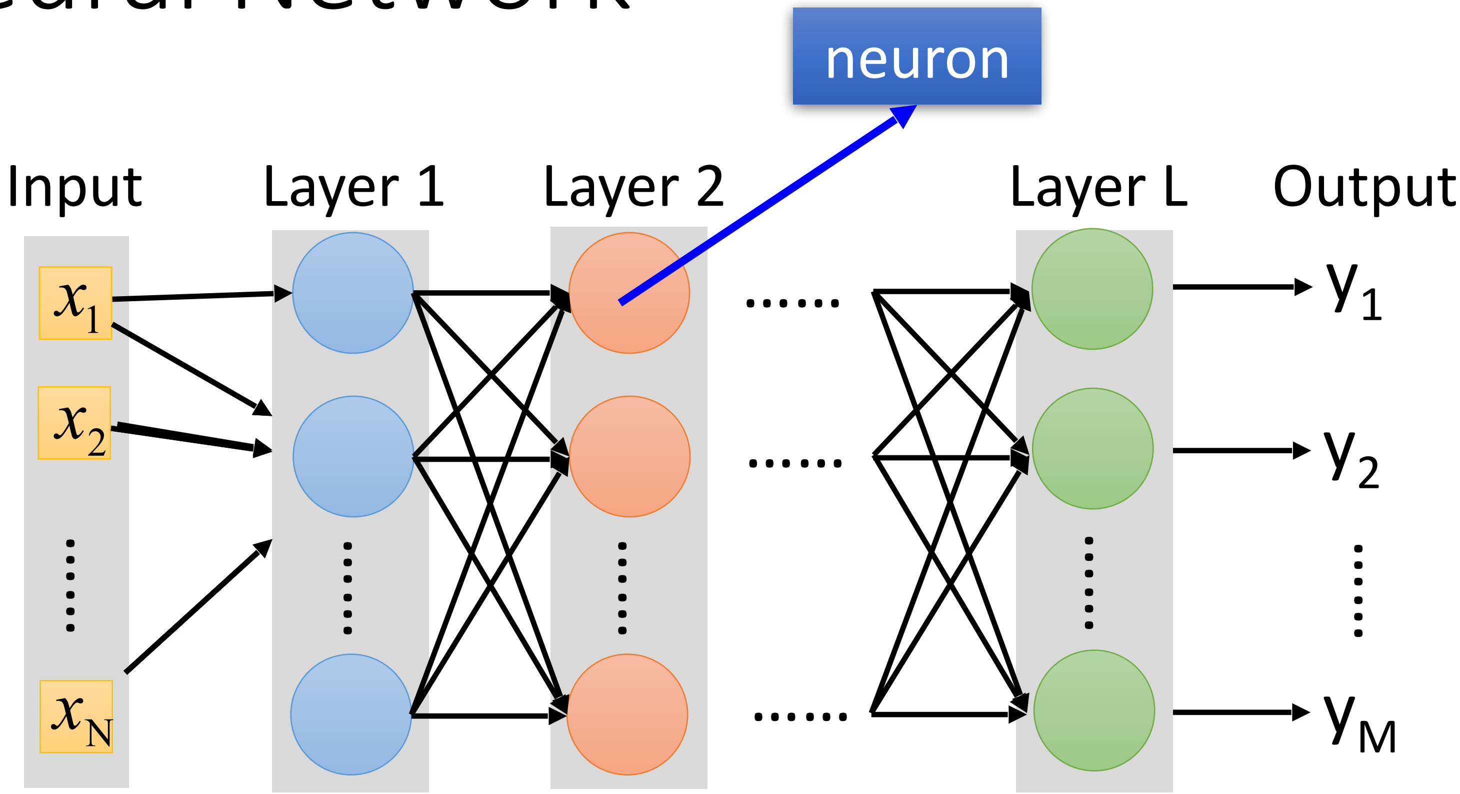
Neural Network



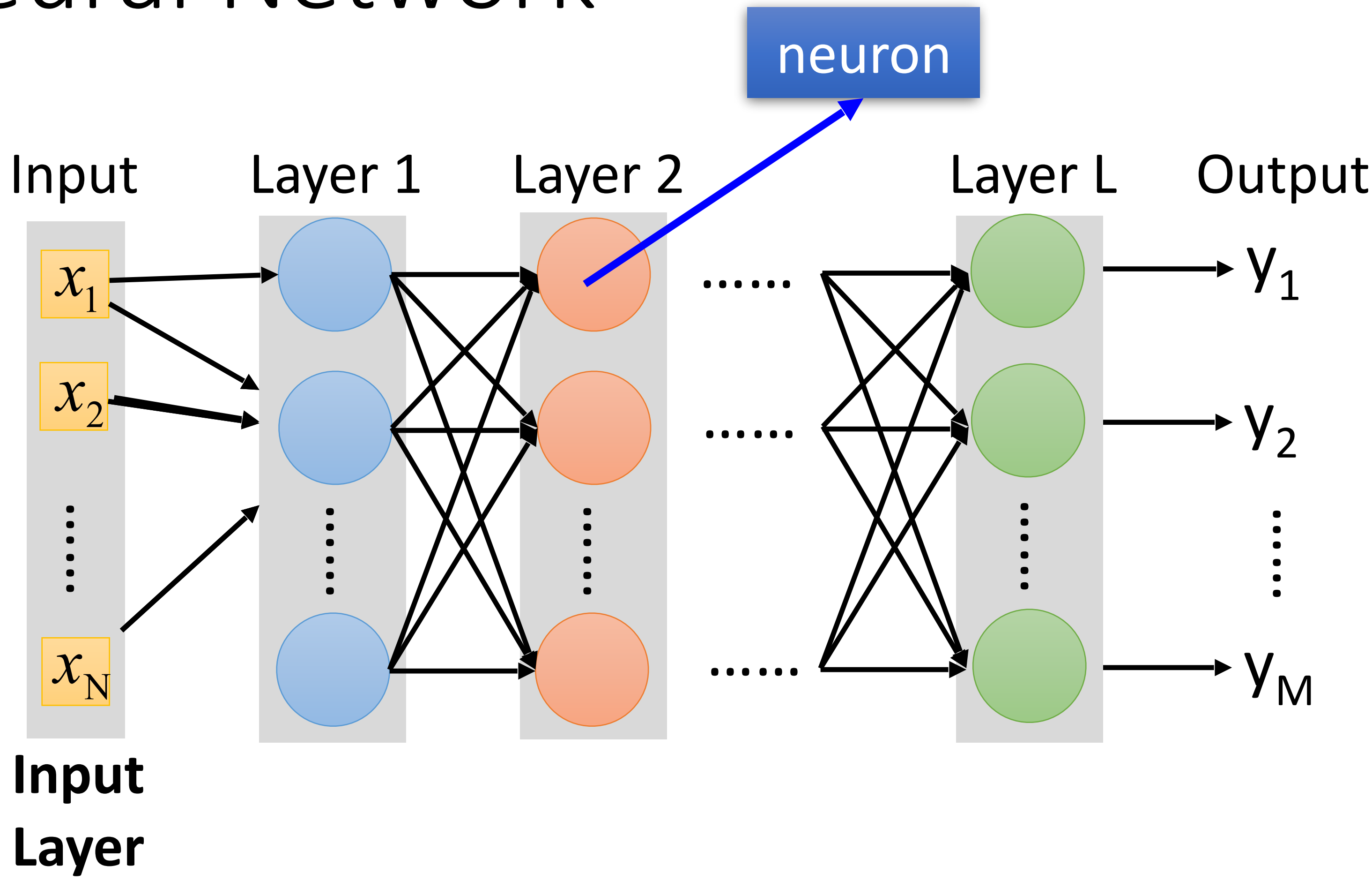
Neural Network



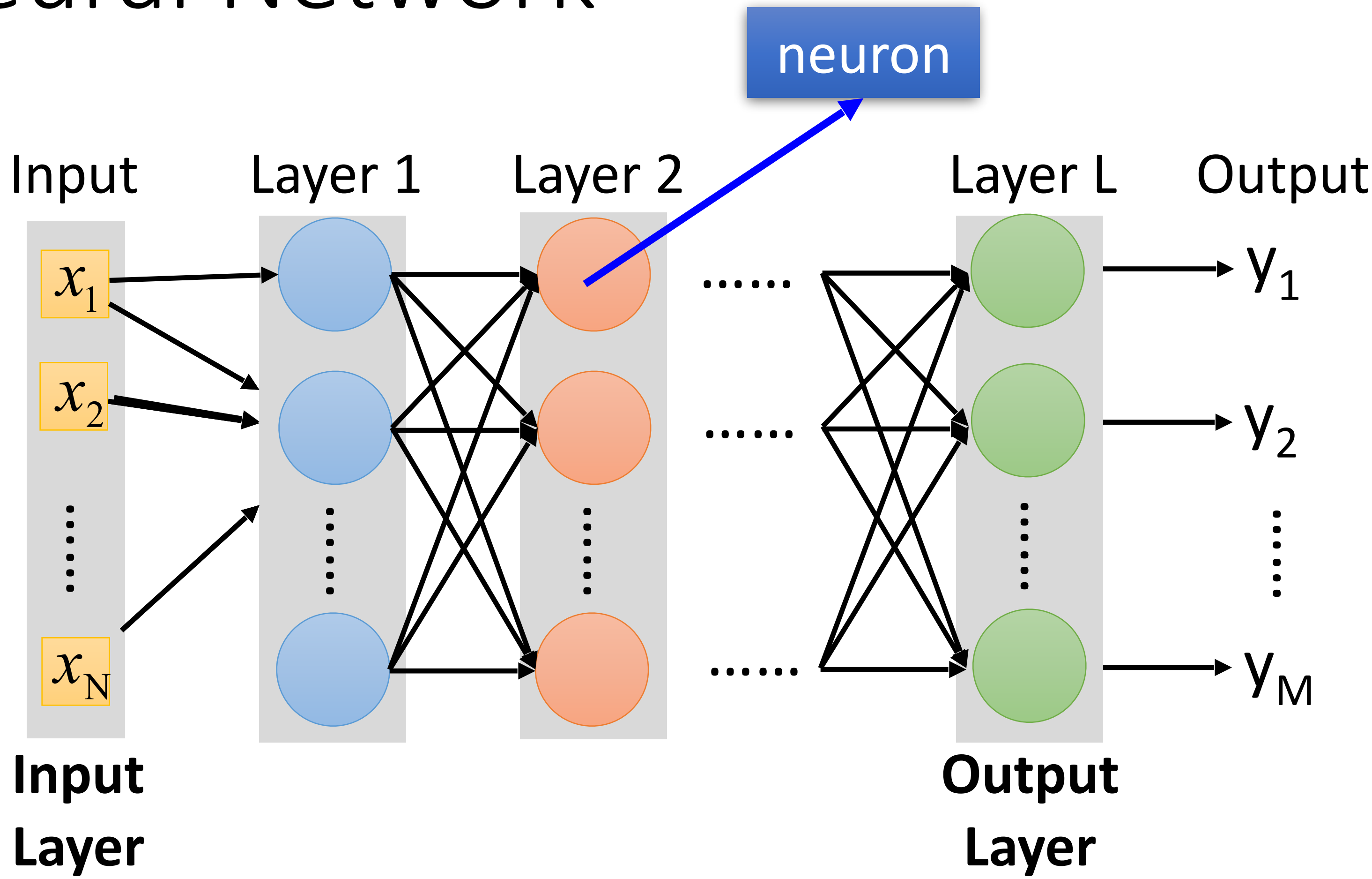
Neural Network



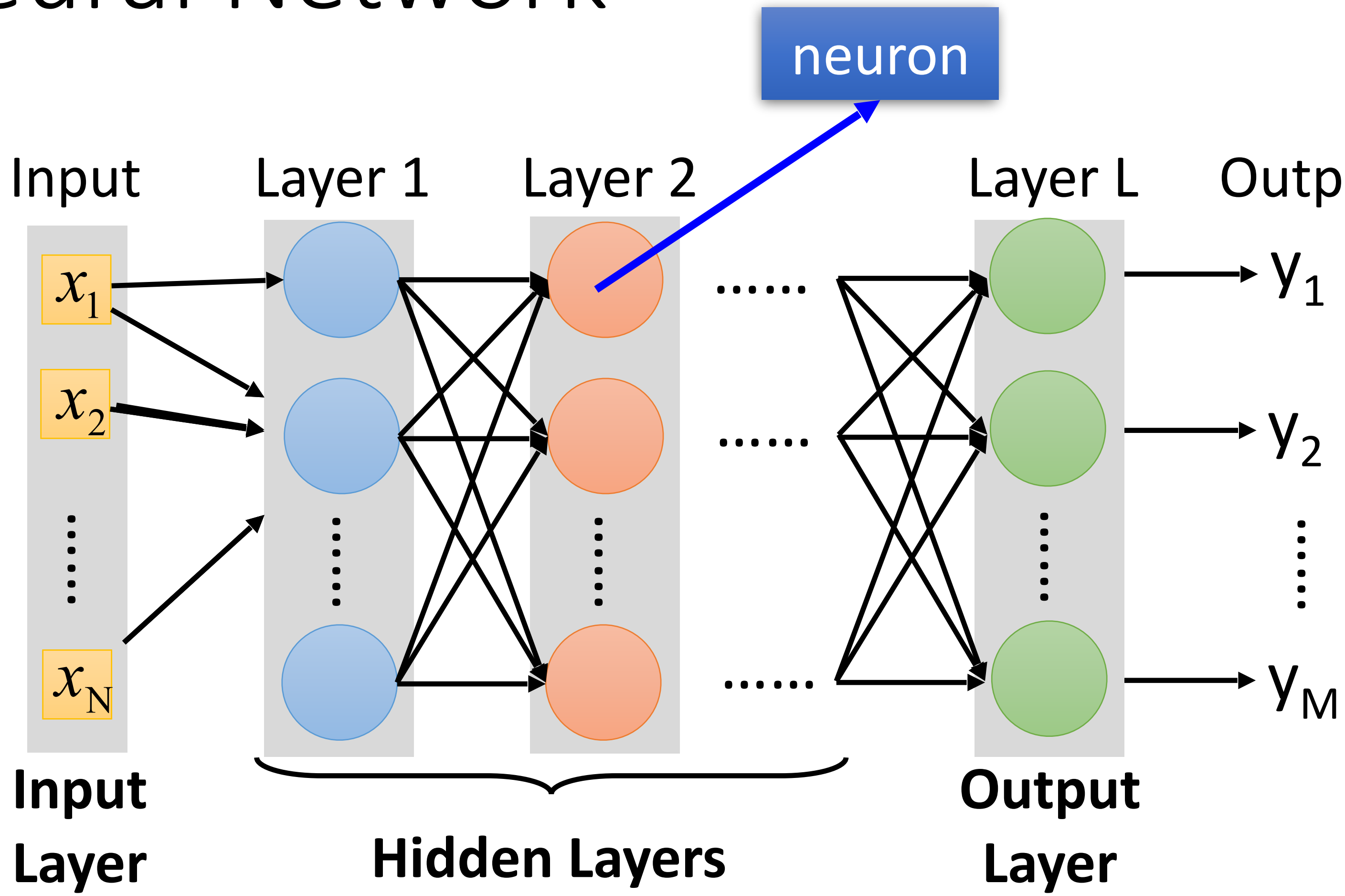
Neural Network



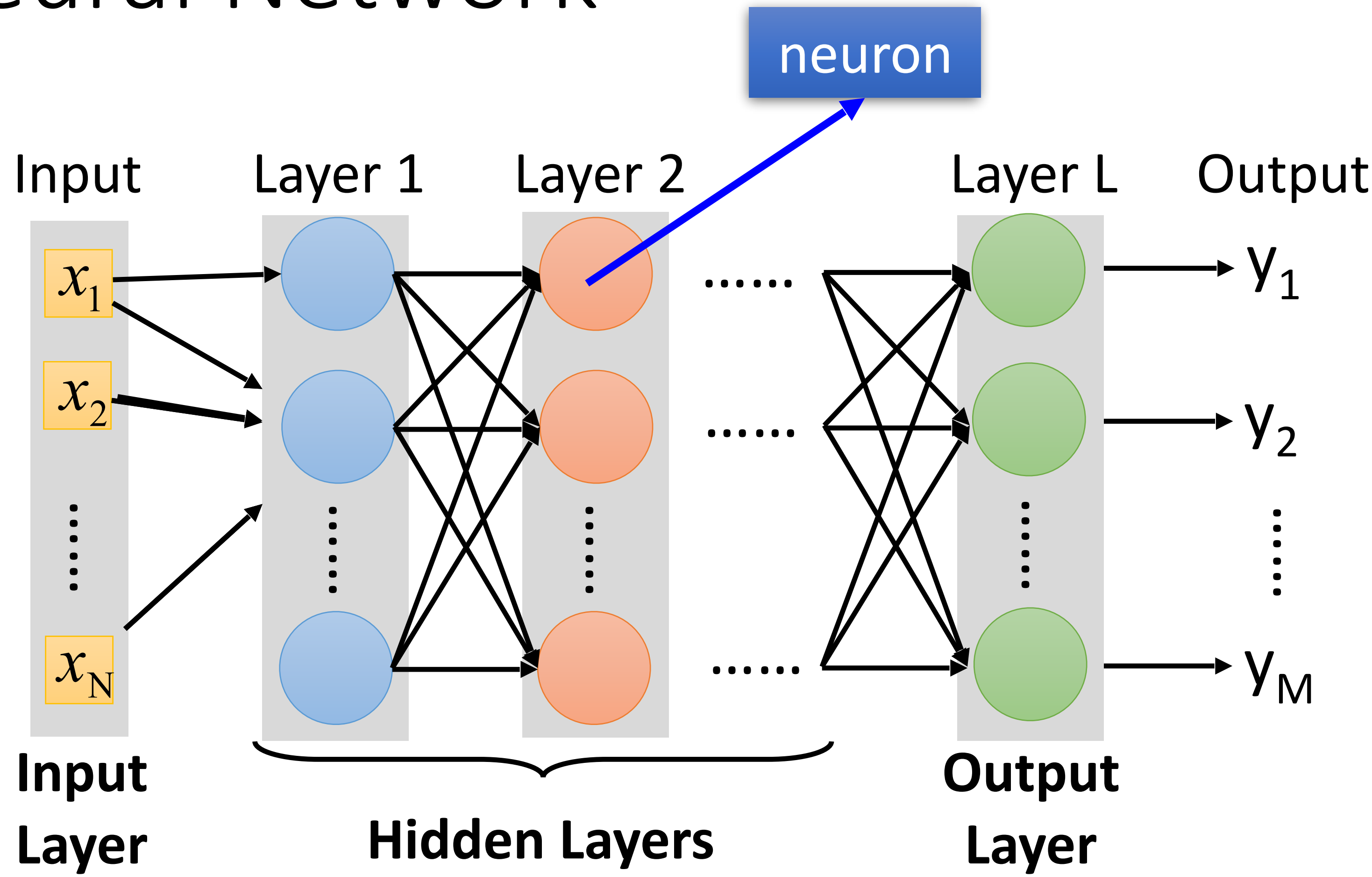
Neural Network



Neural Network

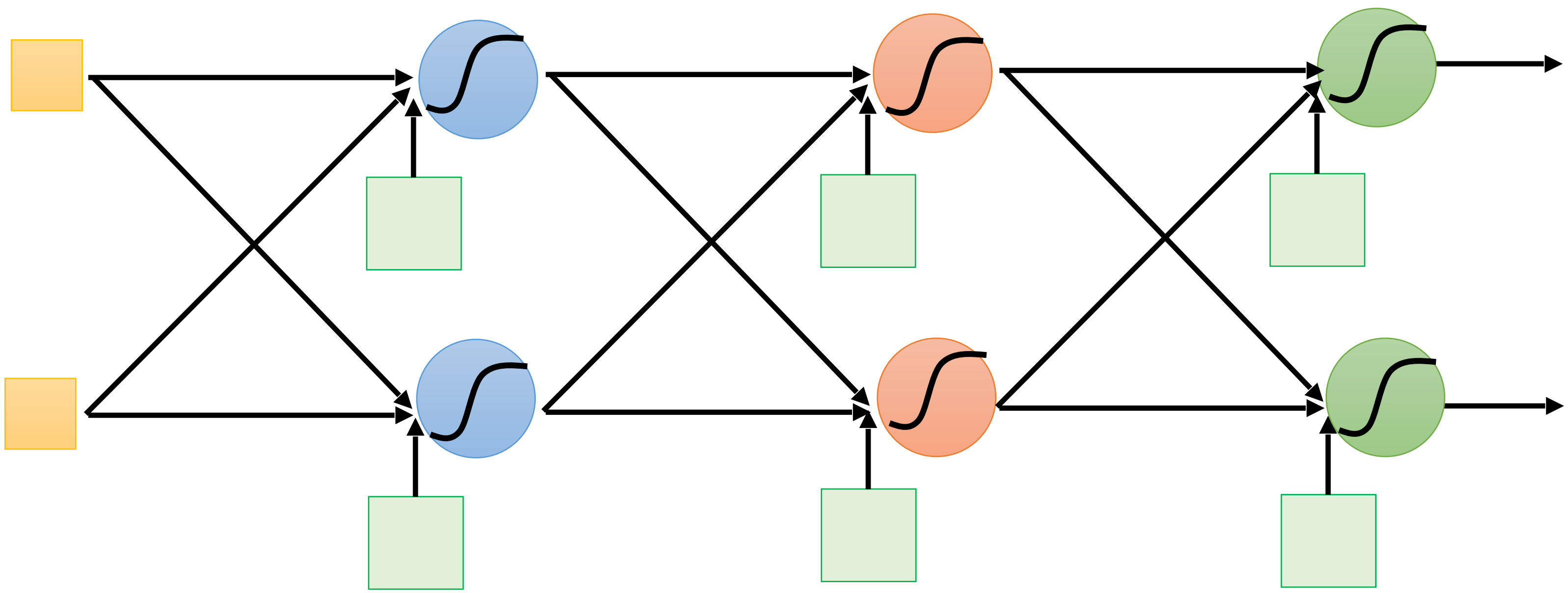


Neural Network

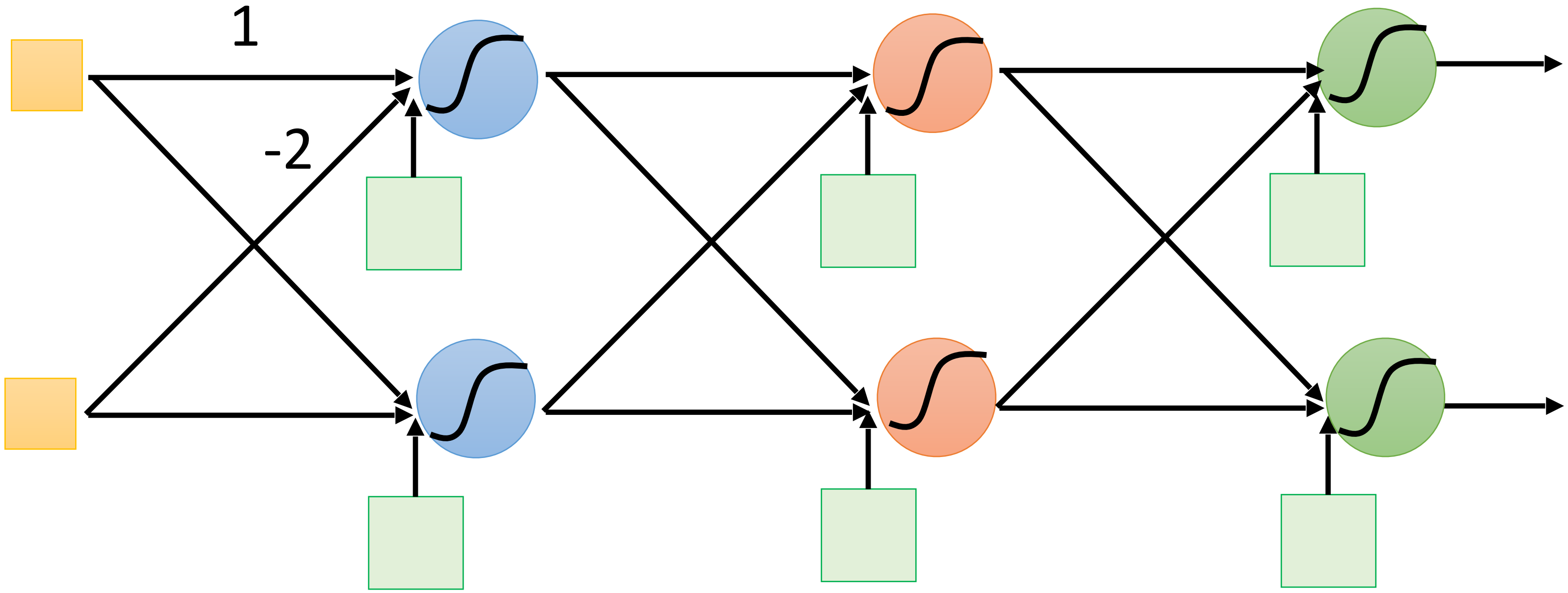


Deep means many hidden layers

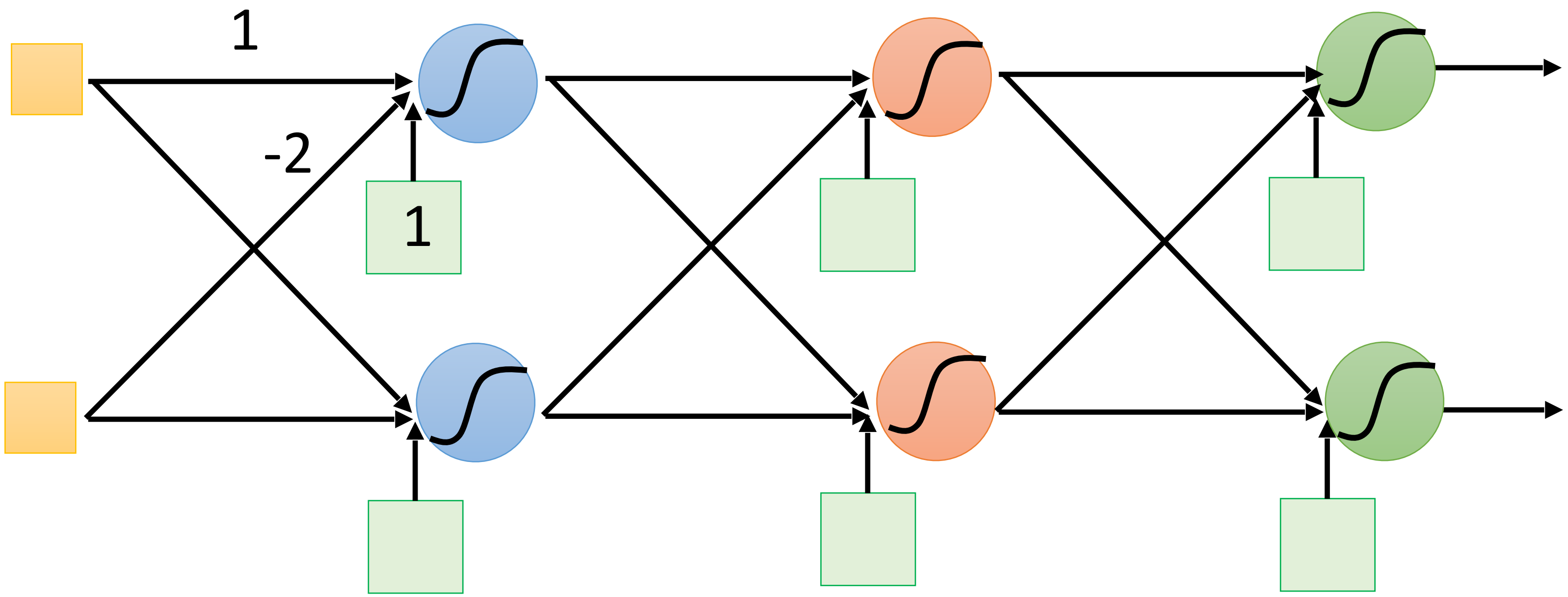
Example of Neural Network



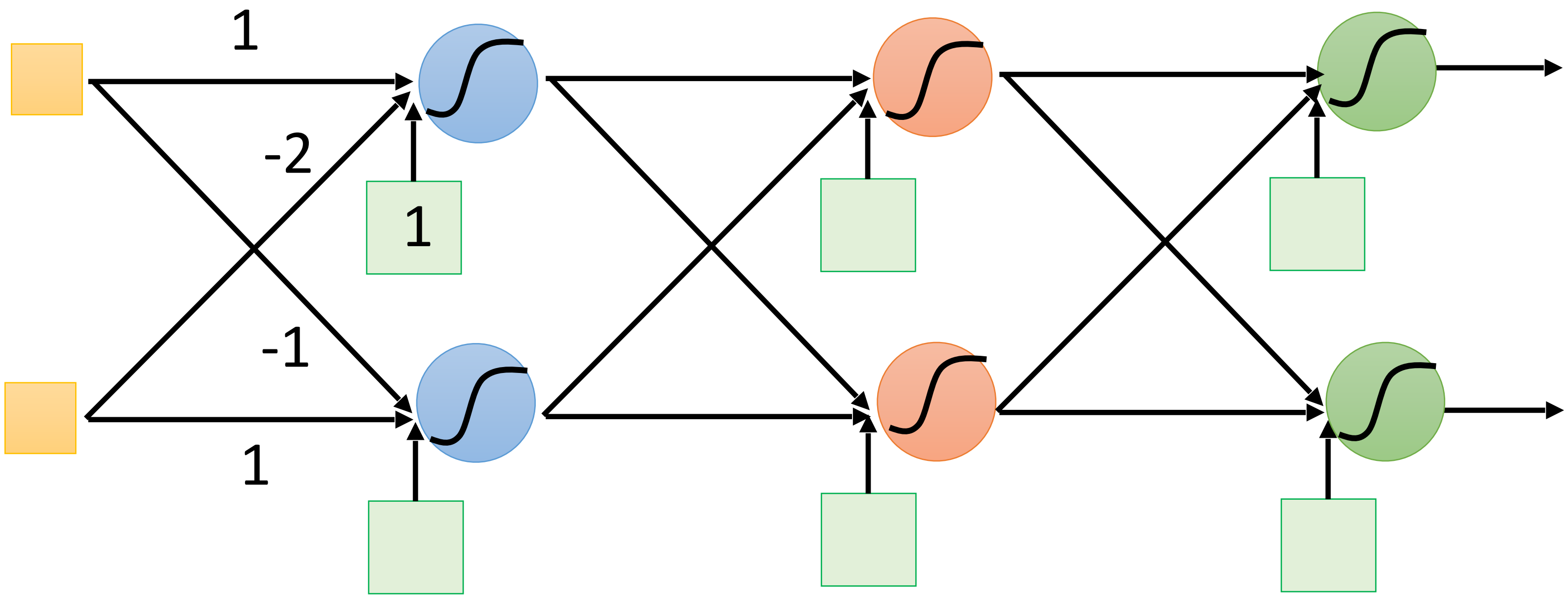
Example of Neural Network



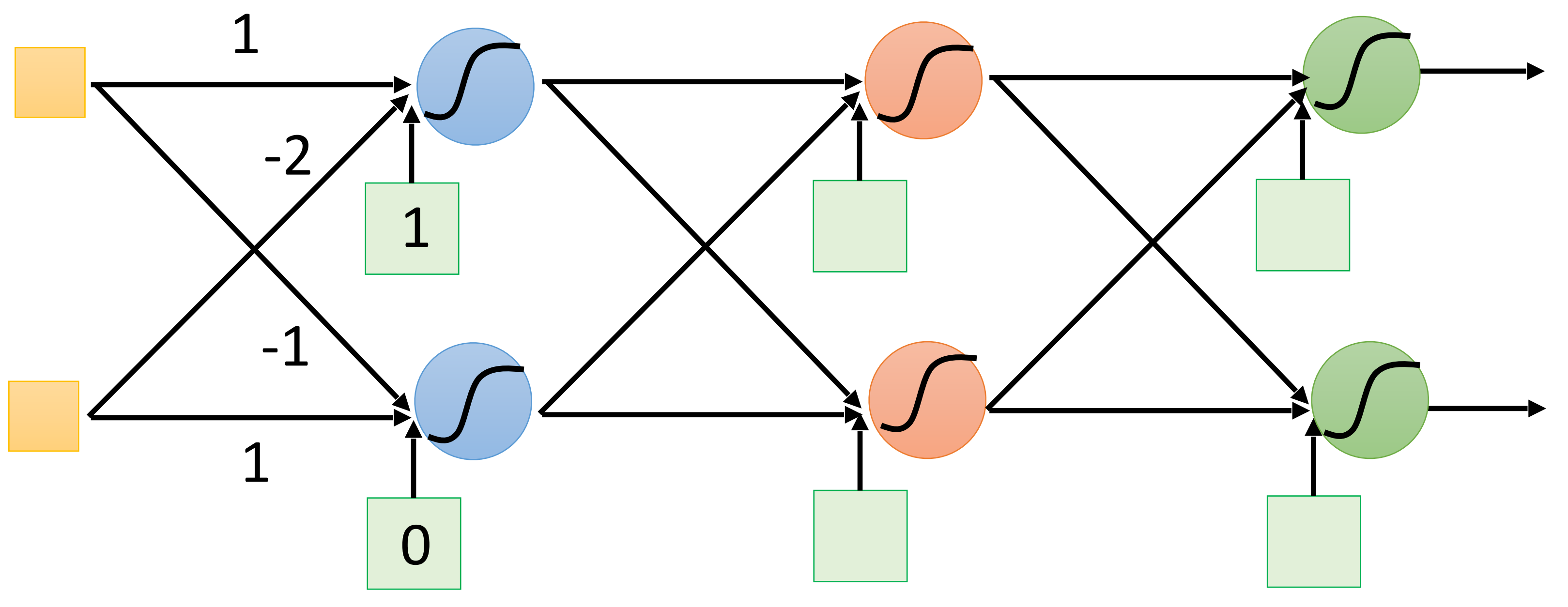
Example of Neural Network



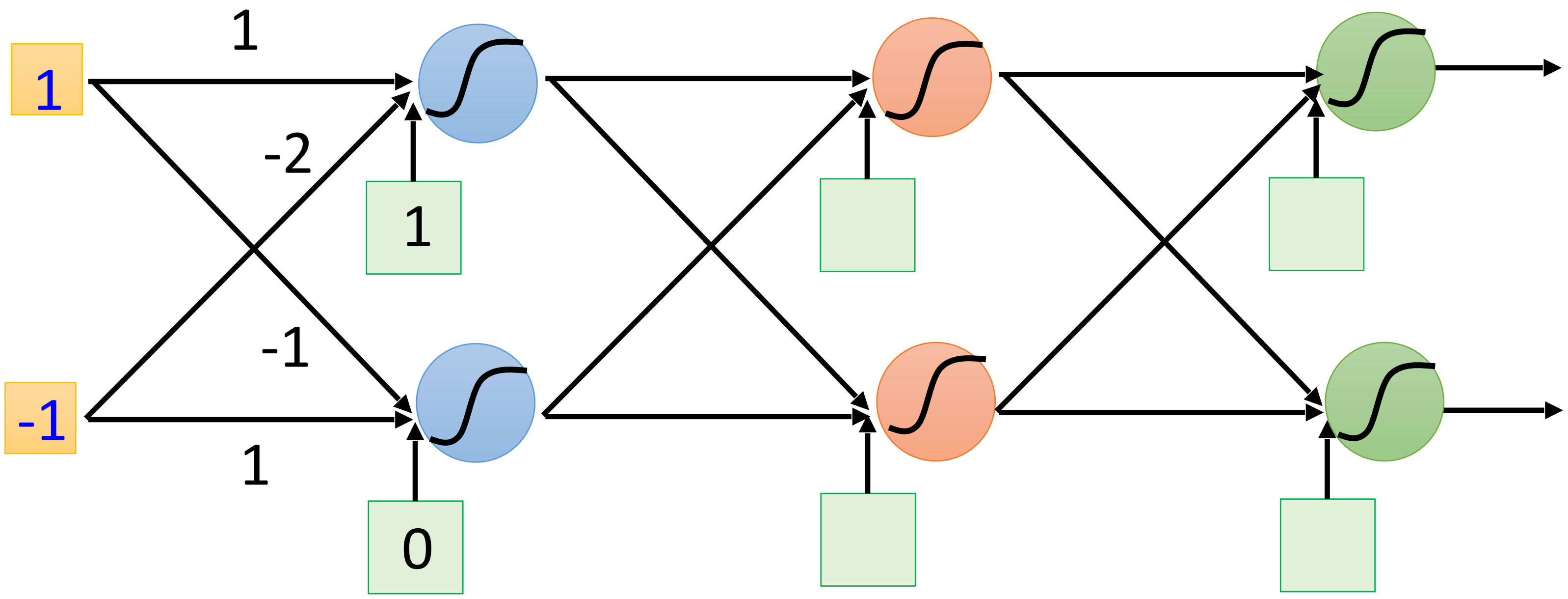
Example of Neural Network



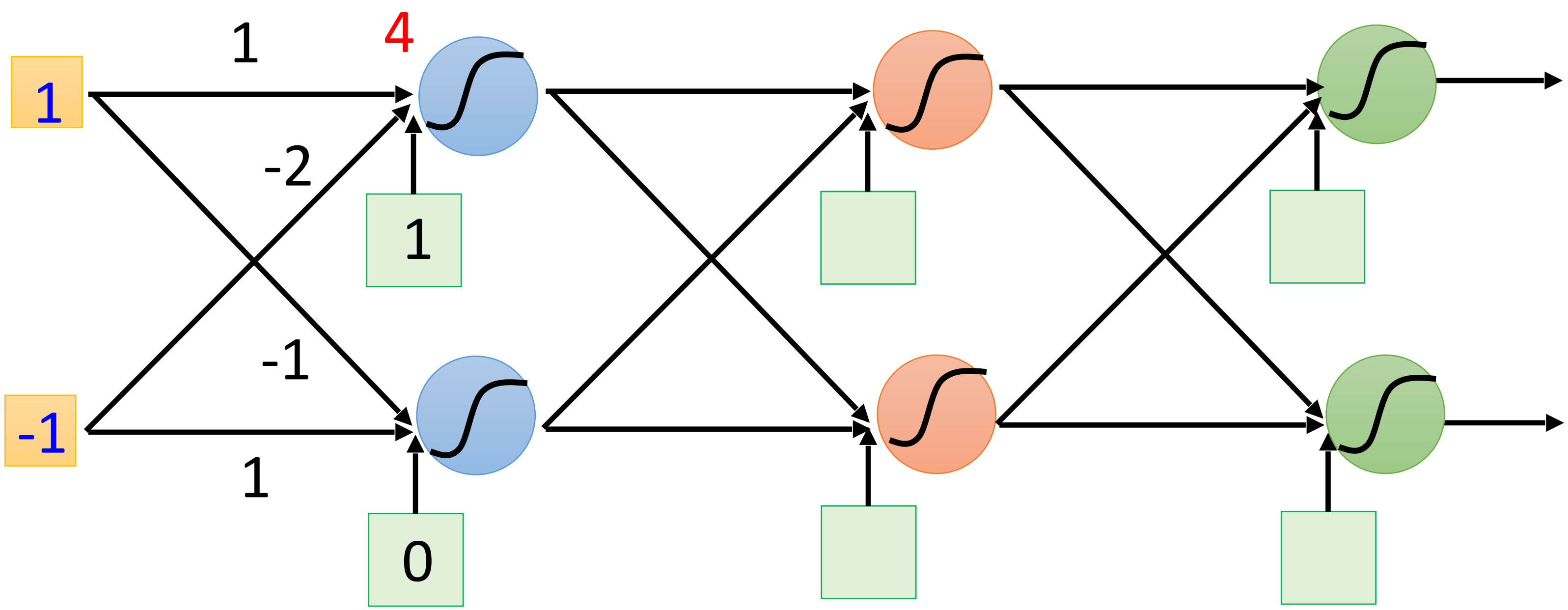
Example of Neural Network



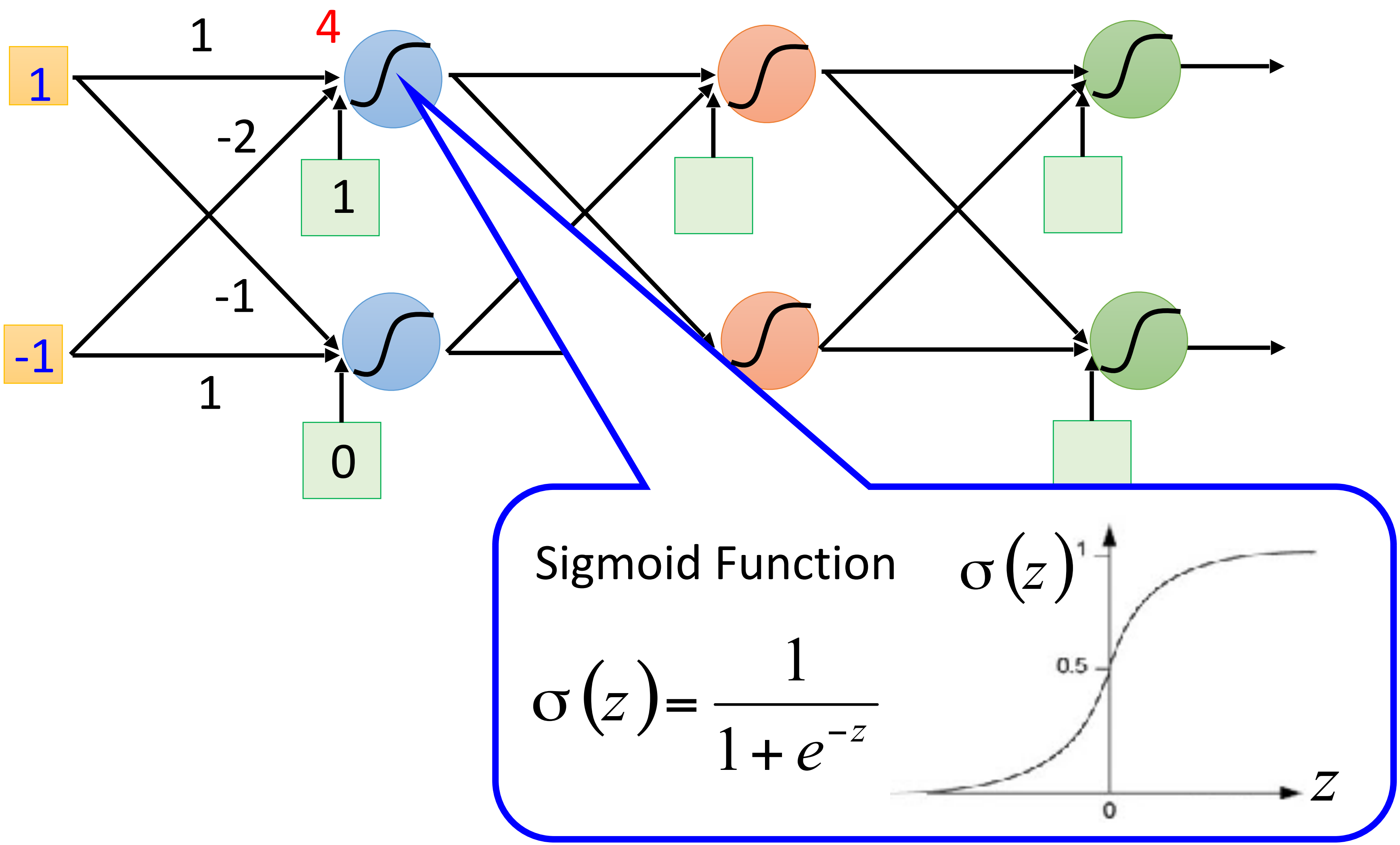
Example of Neural Network



Example of Neural Network

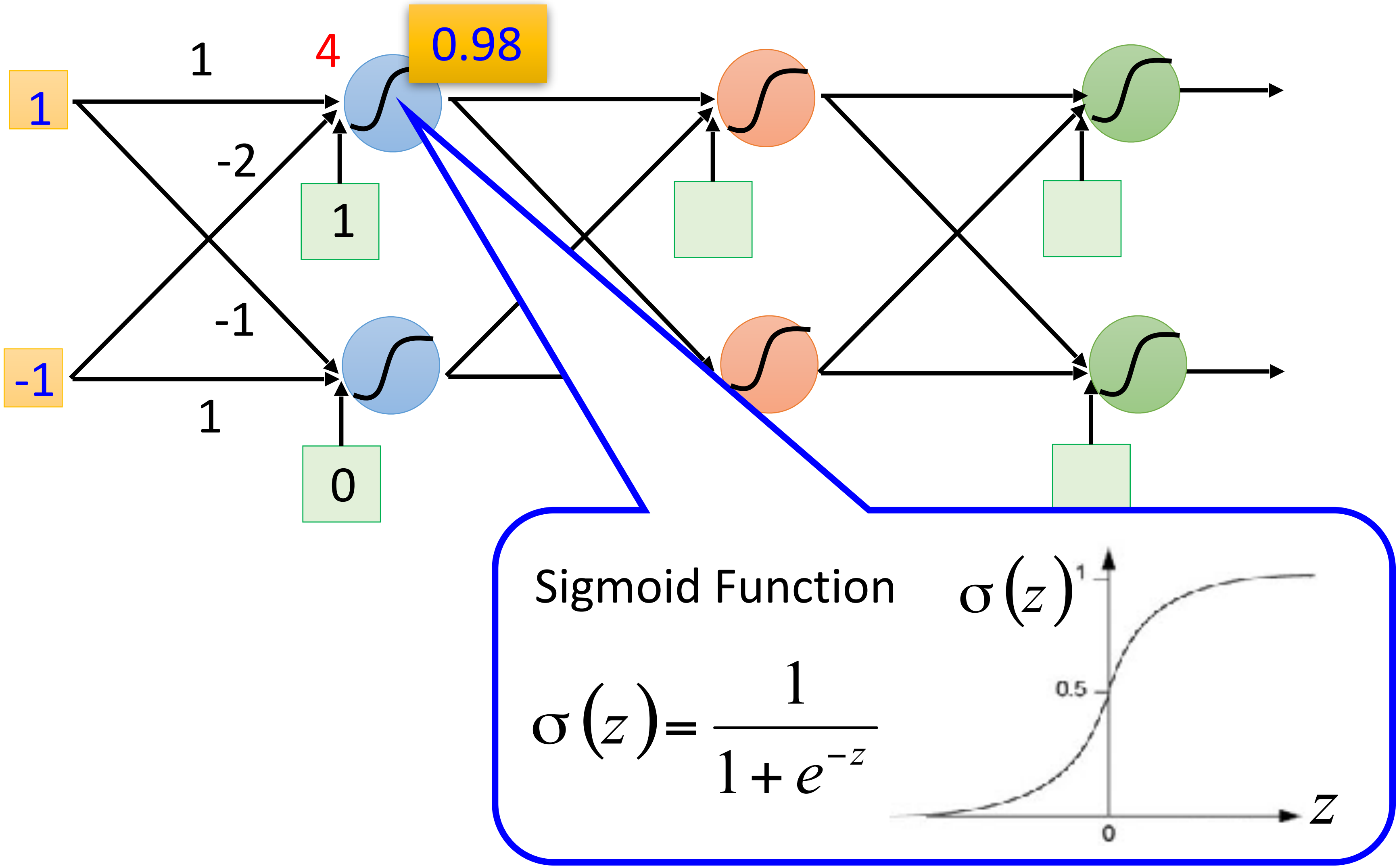


Example of Neural Network



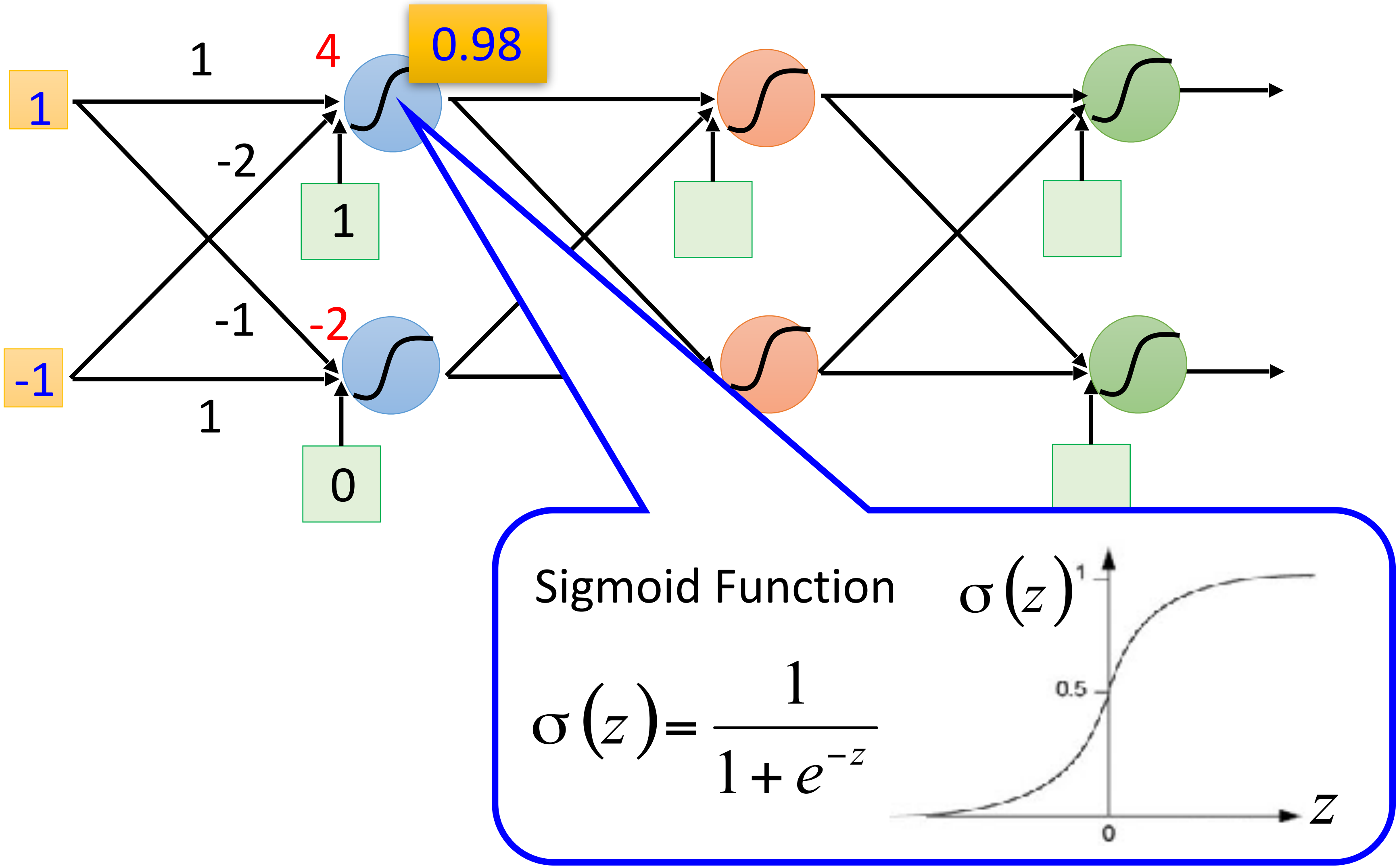
$$\sigma(z) = \frac{1}{1+e^{-z}}$$

Example of Neural Network



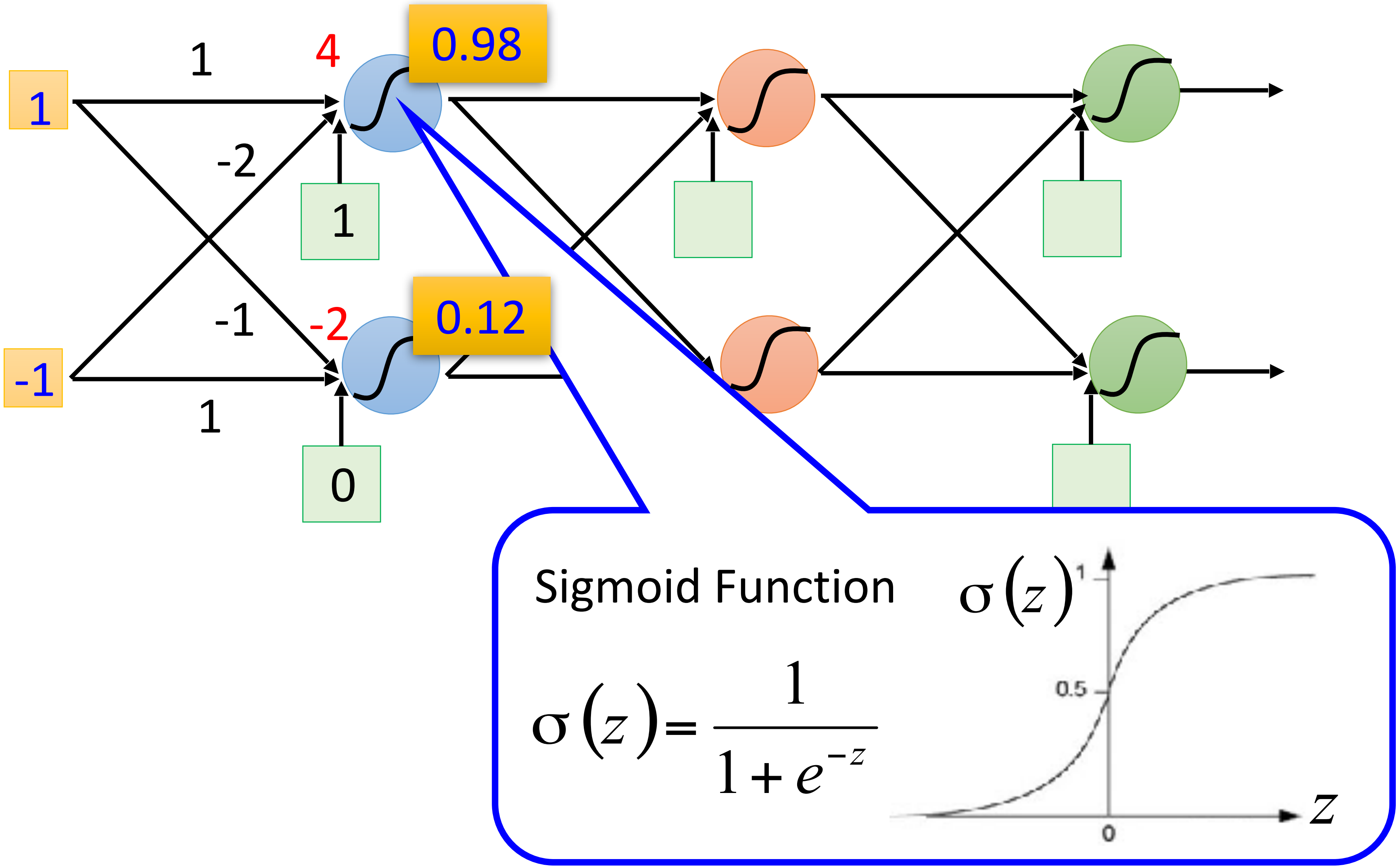
$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Example of Neural Network

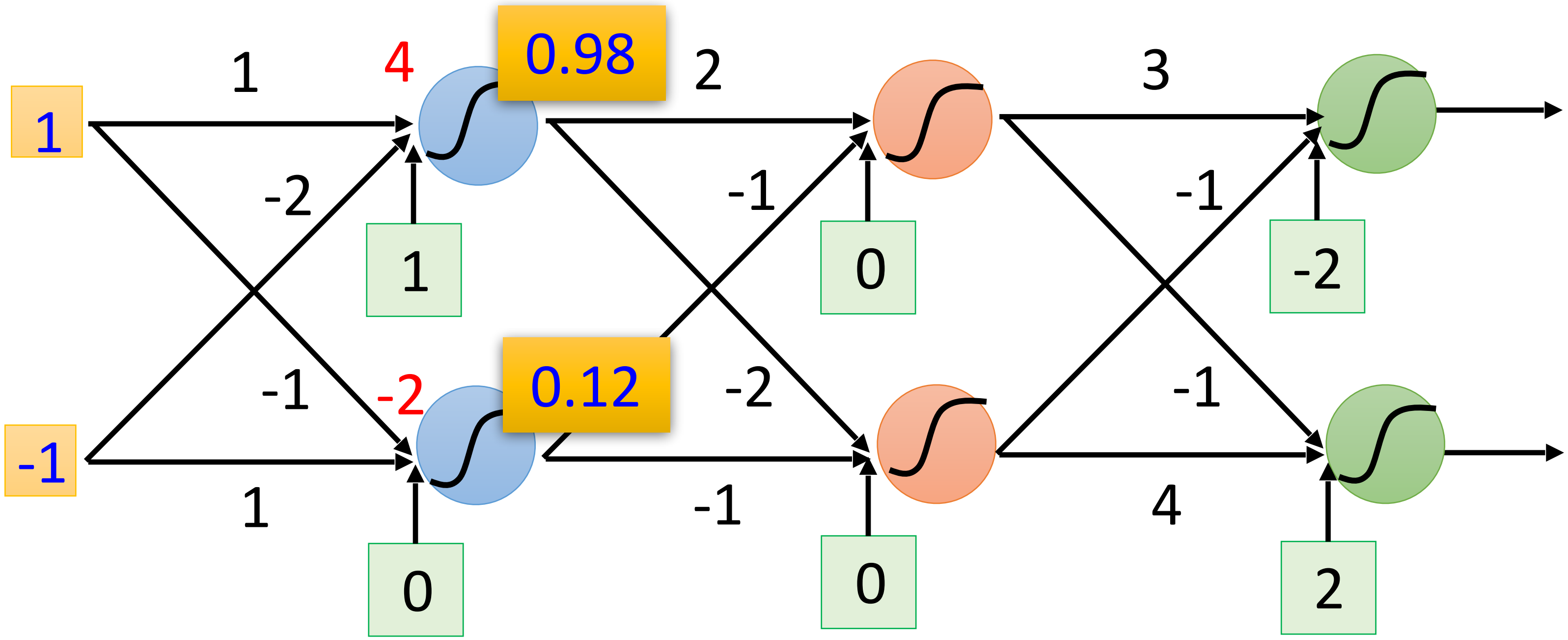


$$\sigma(z) = \frac{1}{1+e^{-z}}$$

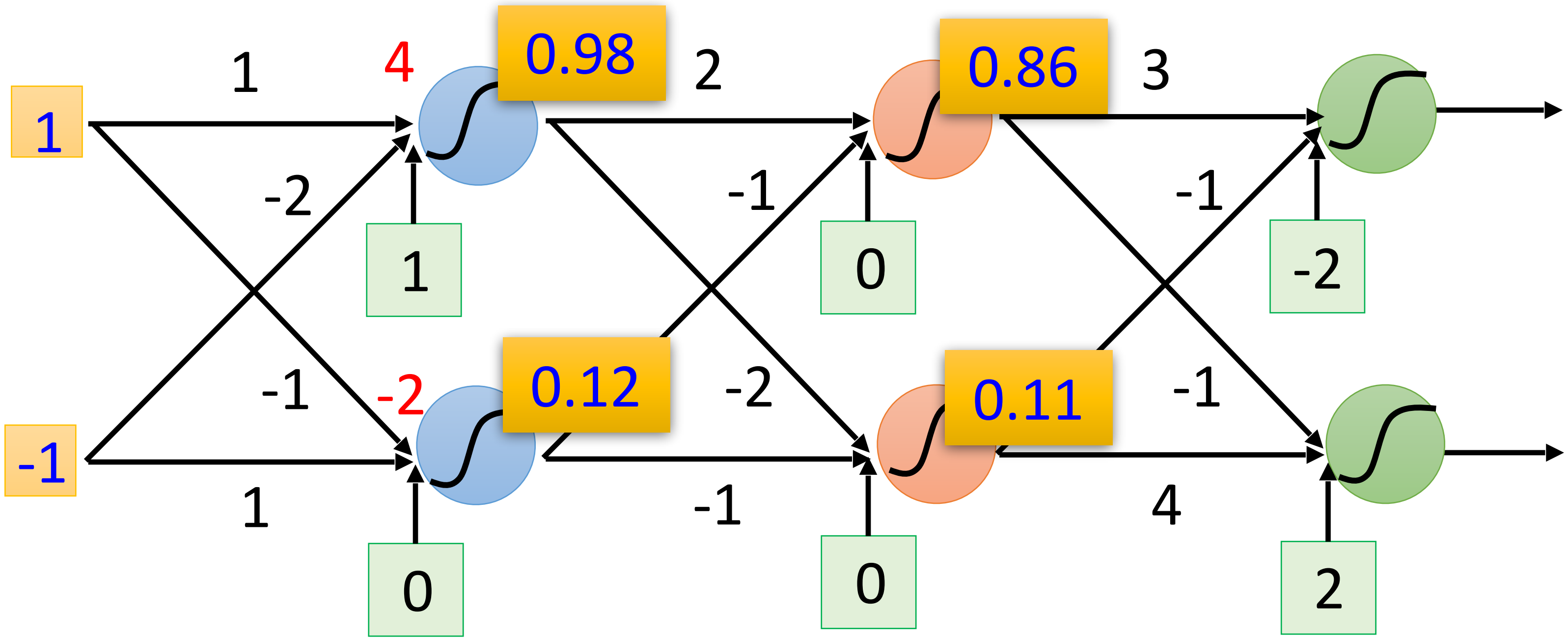
Example of Neural Network



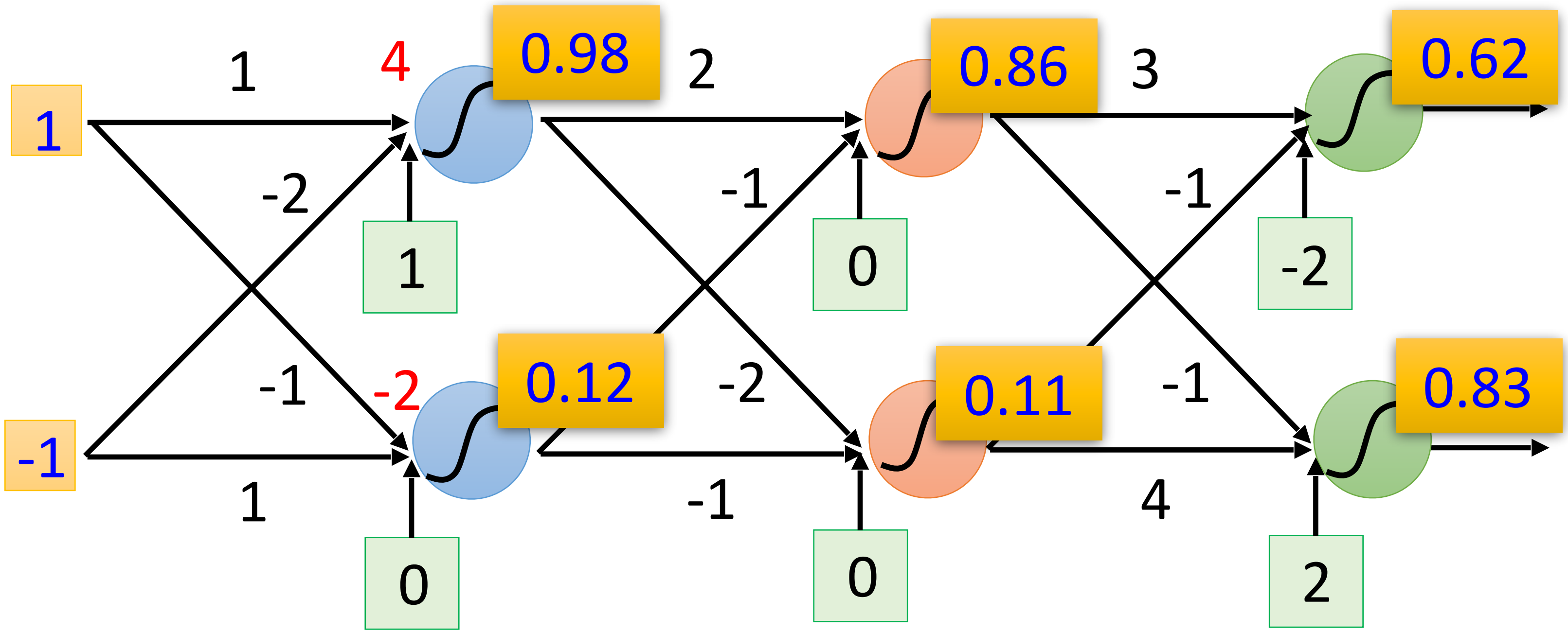
Example of Neural Network



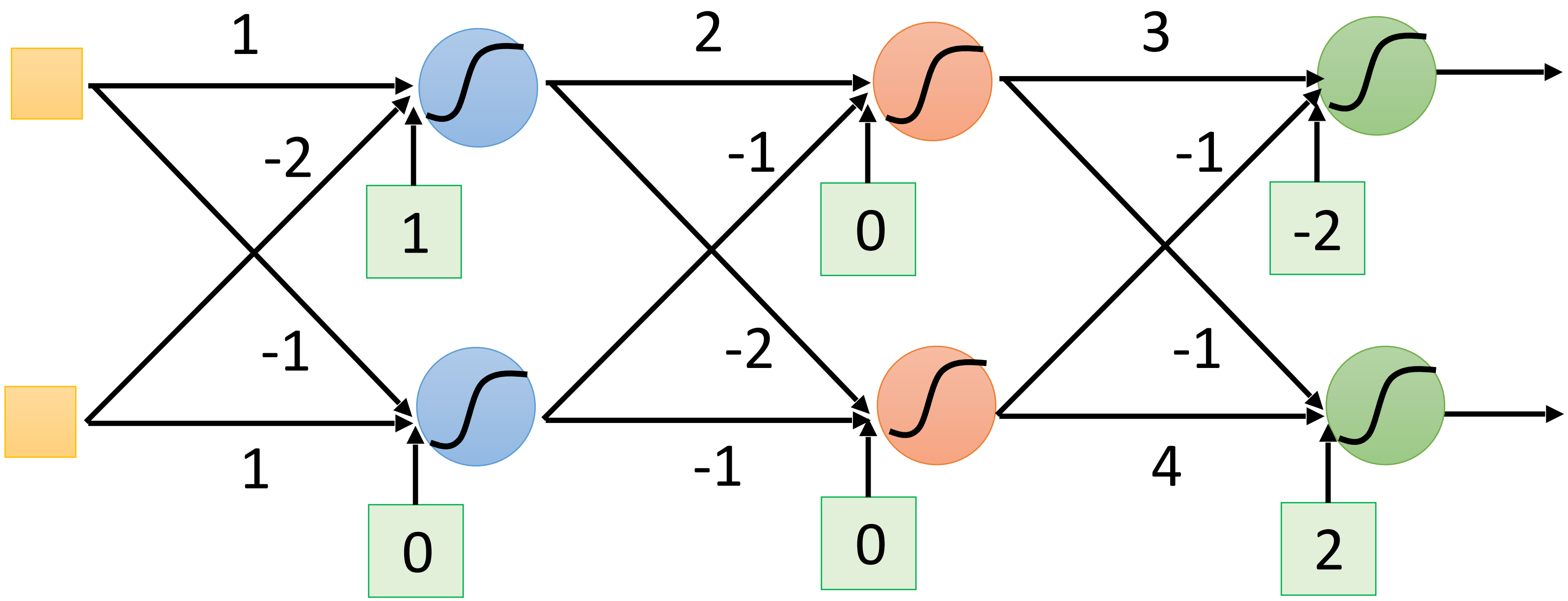
Example of Neural Network



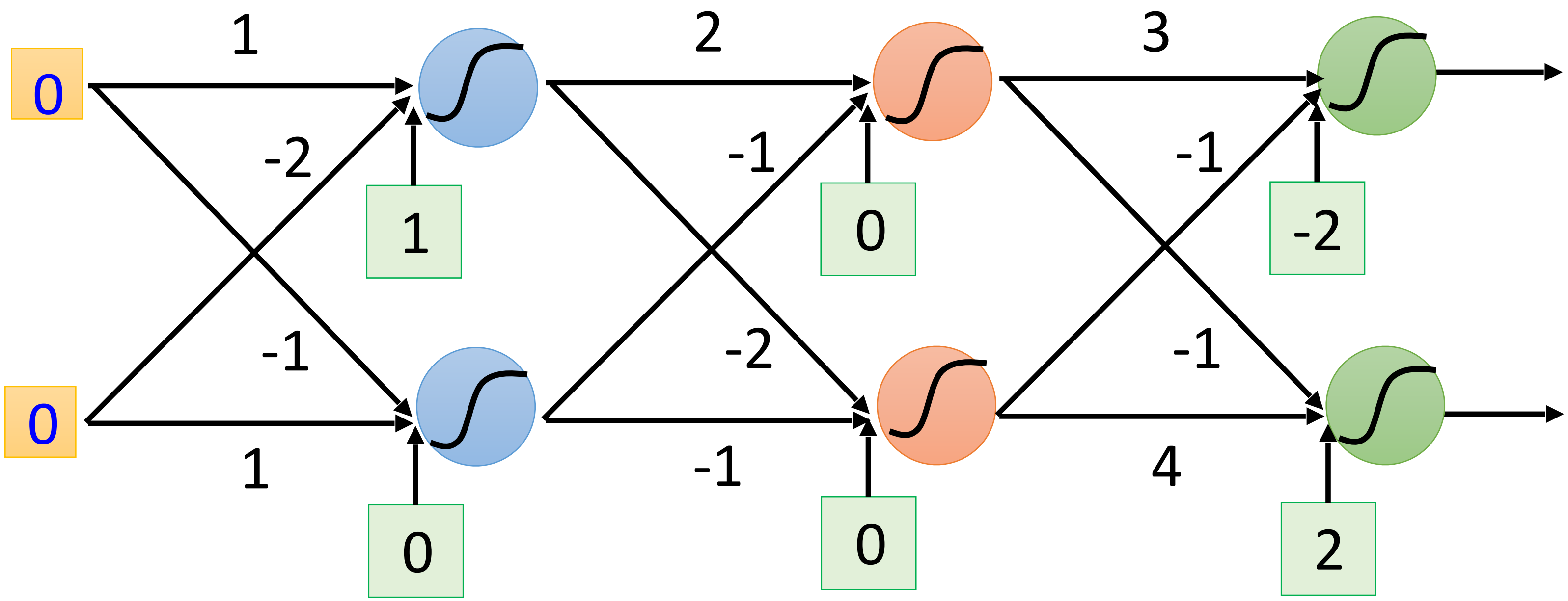
Example of Neural Network



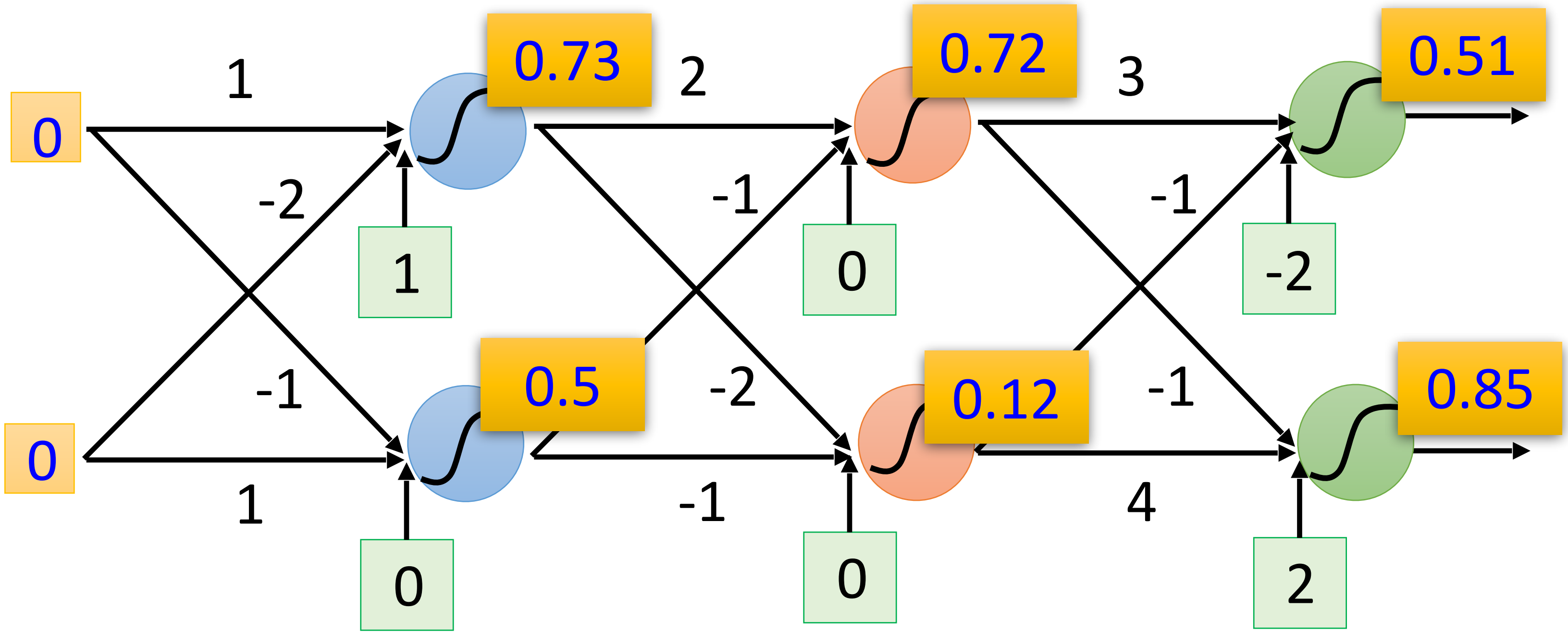
Example of Neural Network



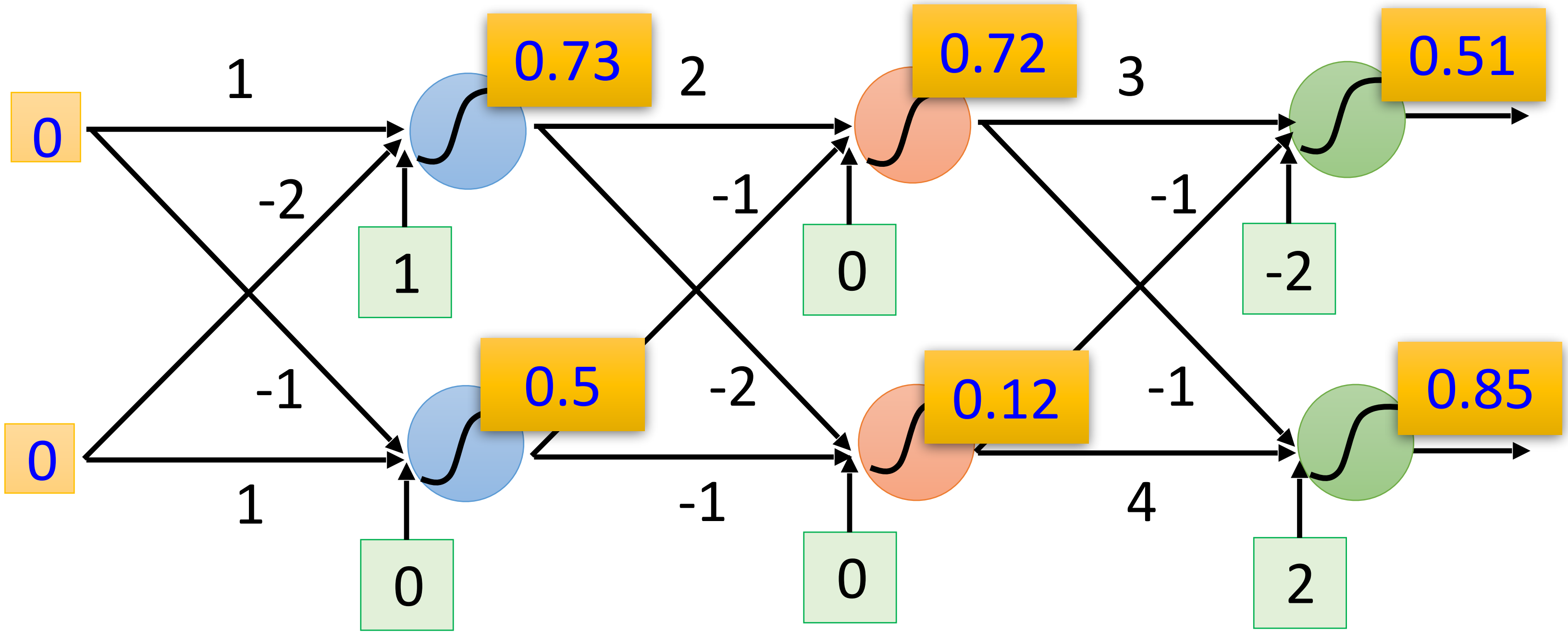
Example of Neural Network



Example of Neural Network

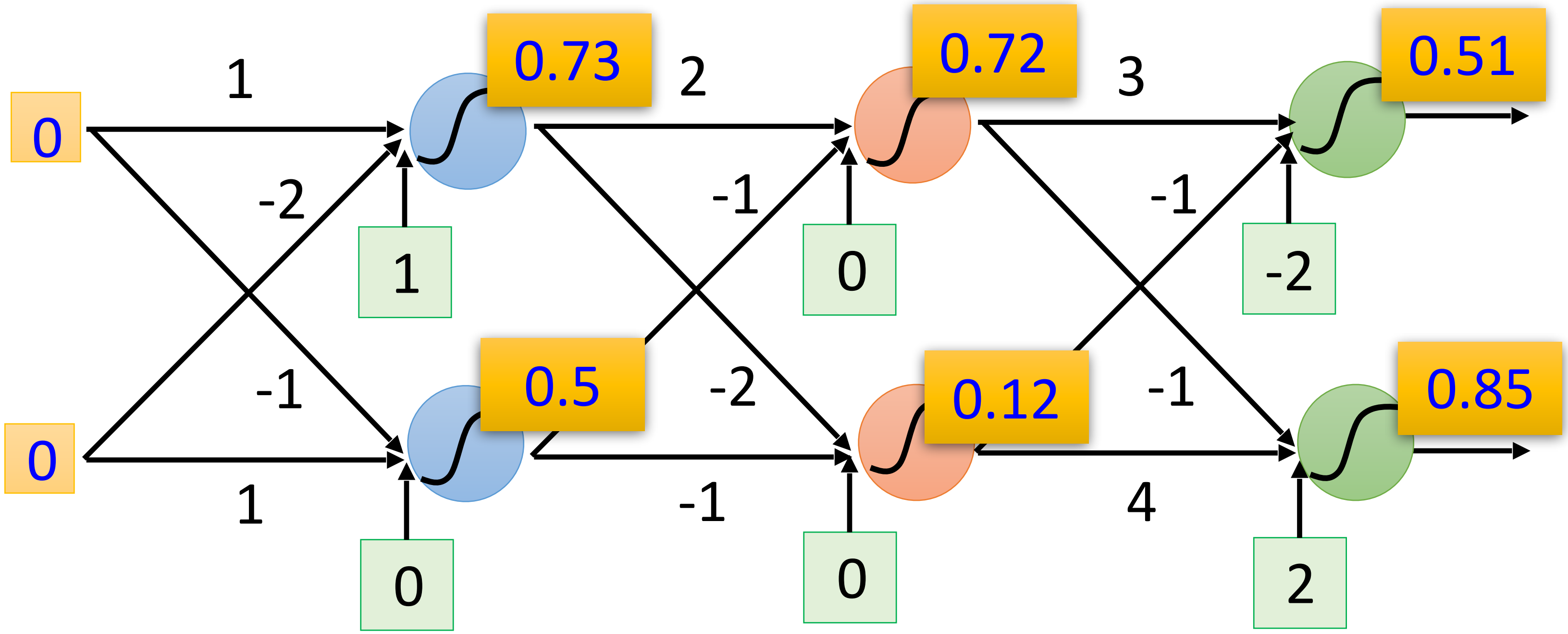


Example of Neural Network



$f: R^2 \to R^2$

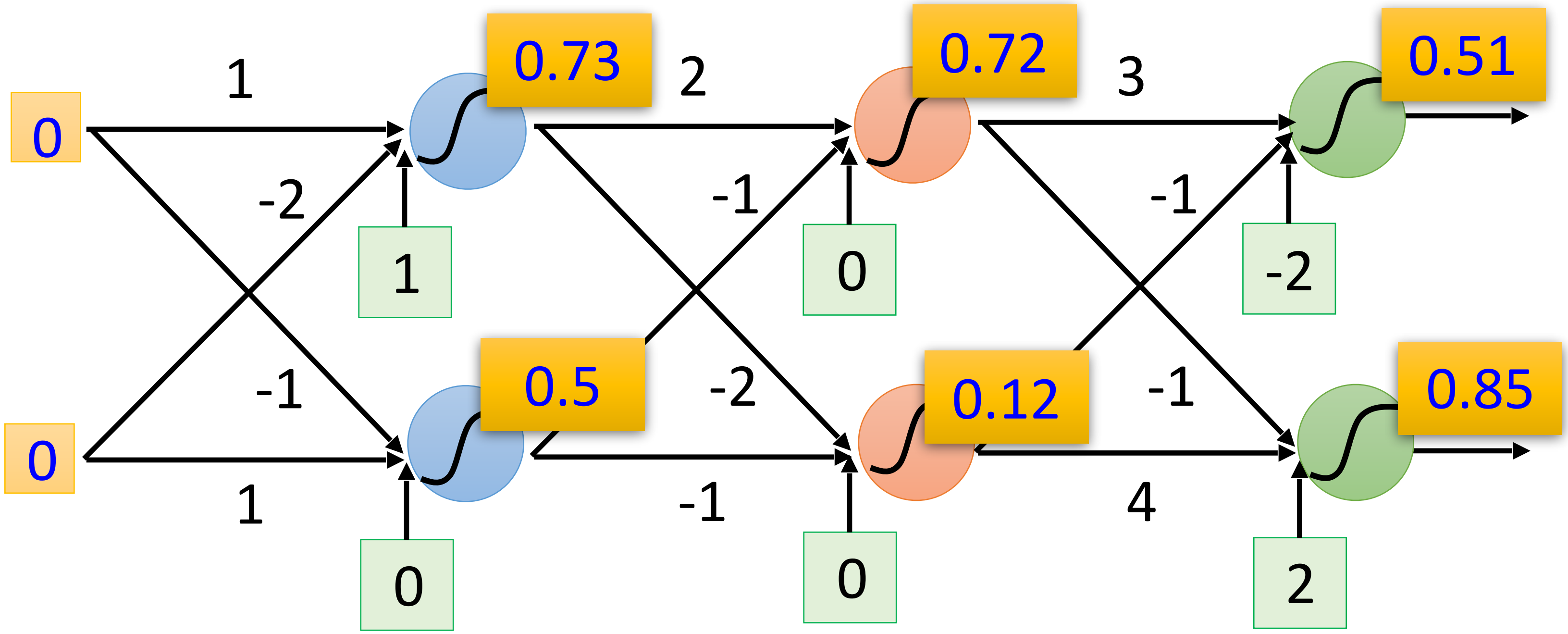
Example of Neural Network



$f: R^2 \to R^2$

$$f\left(\begin{bmatrix} 1 \\ -1 \end{bmatrix}\right) = \begin{bmatrix} 0.62 \\ 0.83 \end{bmatrix}$$

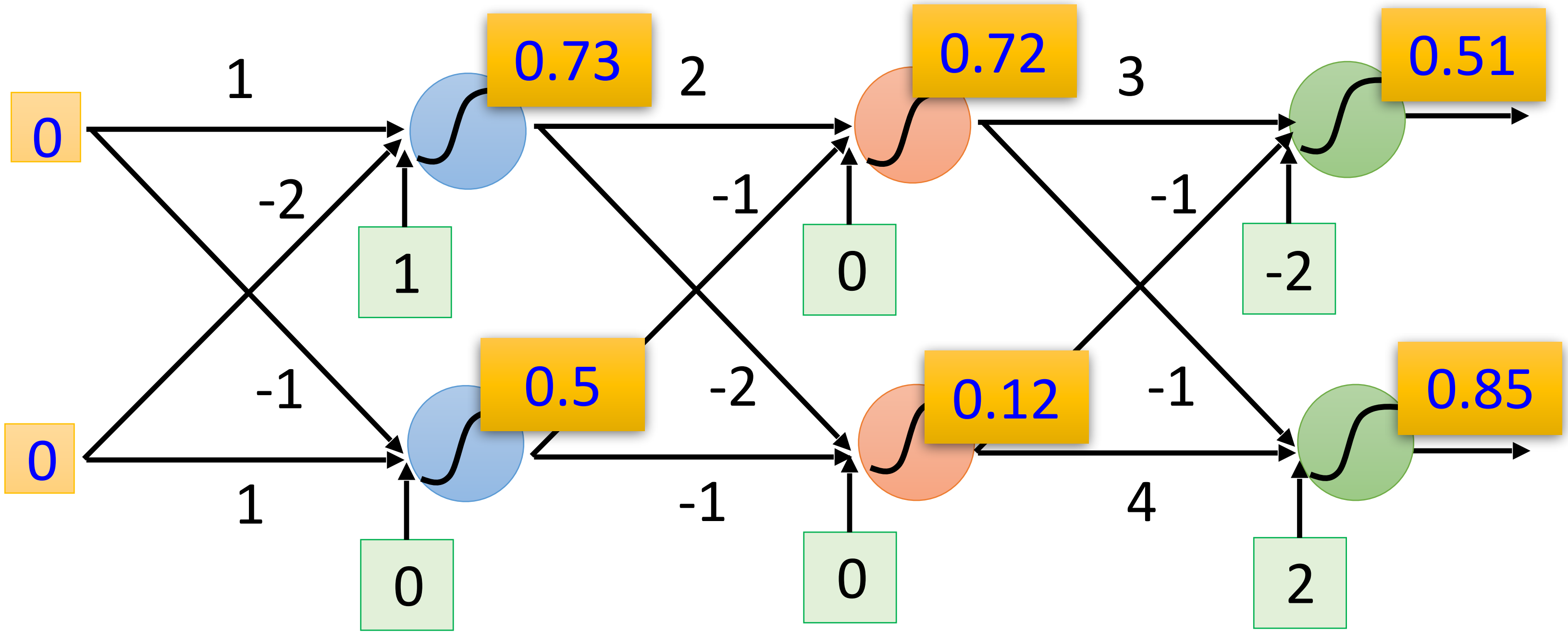
Example of Neural Network



$f: R^2 \rightarrow R^2$

$$f\left(\begin{bmatrix} 1 \\ -1 \end{bmatrix}\right) = \begin{bmatrix} 0.62 \\ 0.83 \end{bmatrix} \quad f\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}\right) = \begin{bmatrix} 0.51 \\ 0.85 \end{bmatrix}$$

Example of Neural Network

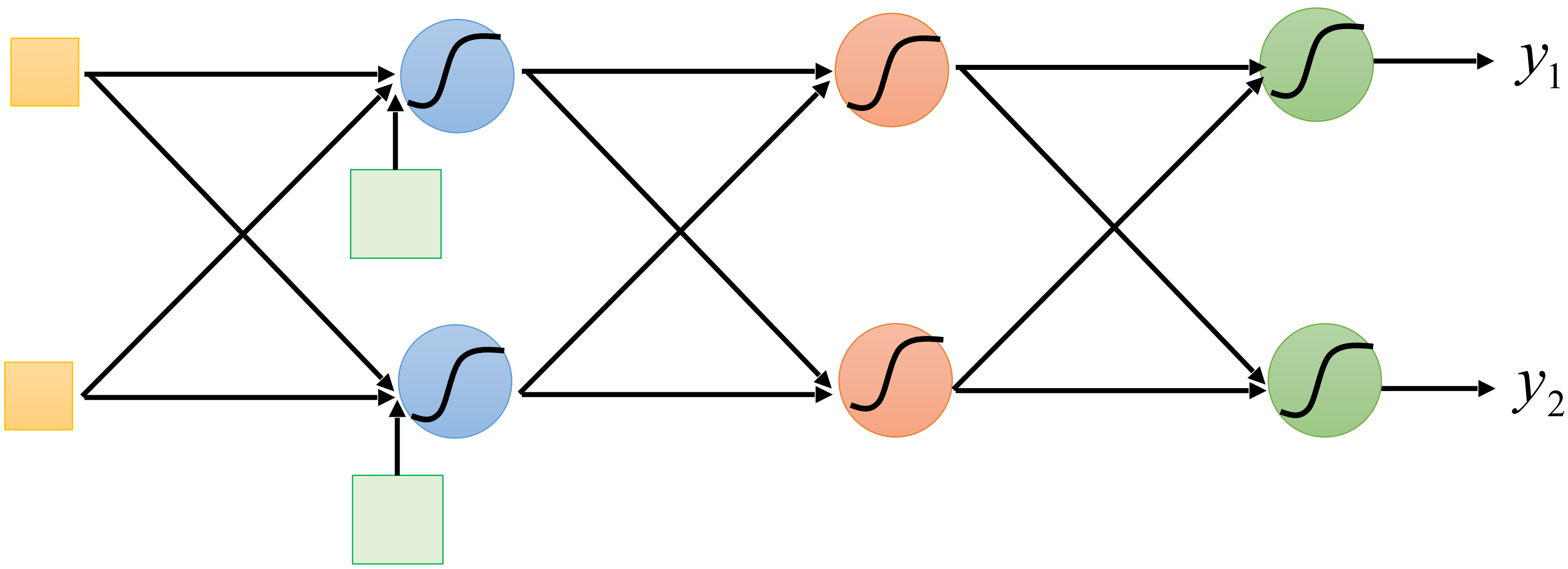


$f: R^2 \to R^2$

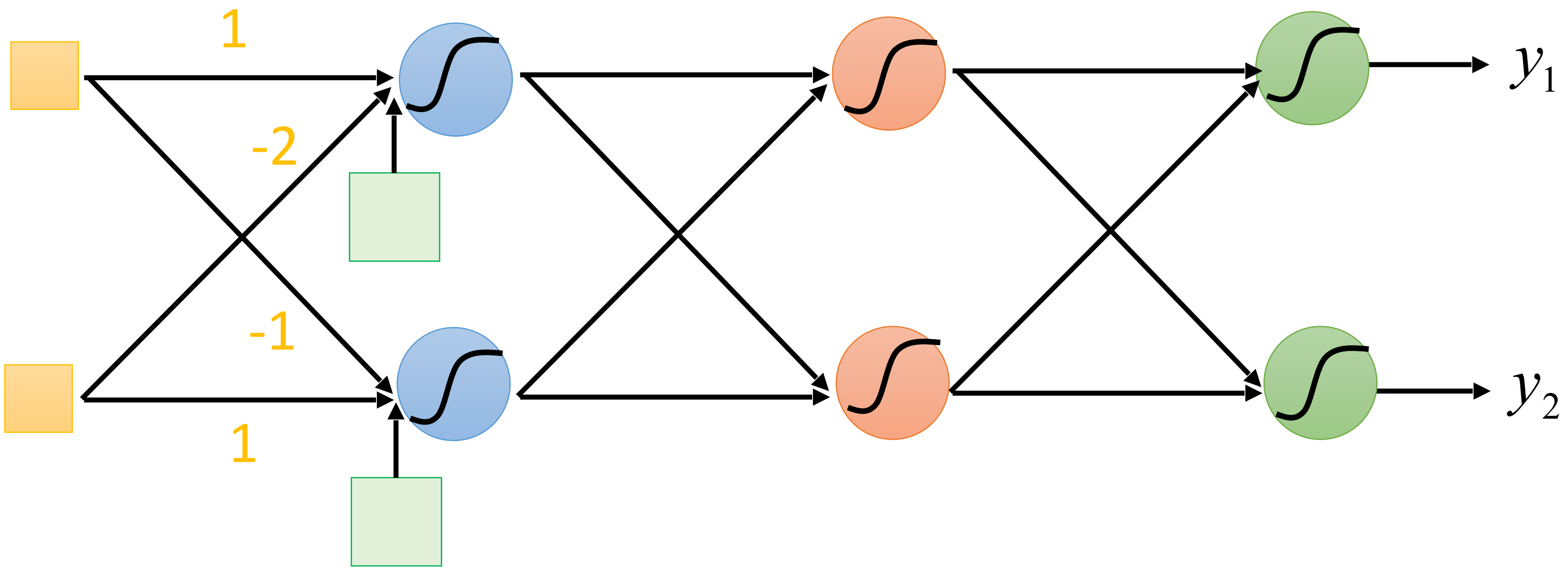
$$f\left(\begin{bmatrix} 1 \\ -1 \end{bmatrix}\right) = \begin{bmatrix} 0.62 \\ 0.83 \end{bmatrix} \quad f\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}\right) = \begin{bmatrix} 0.51 \\ 0.85 \end{bmatrix}$$

Different parameters define different function

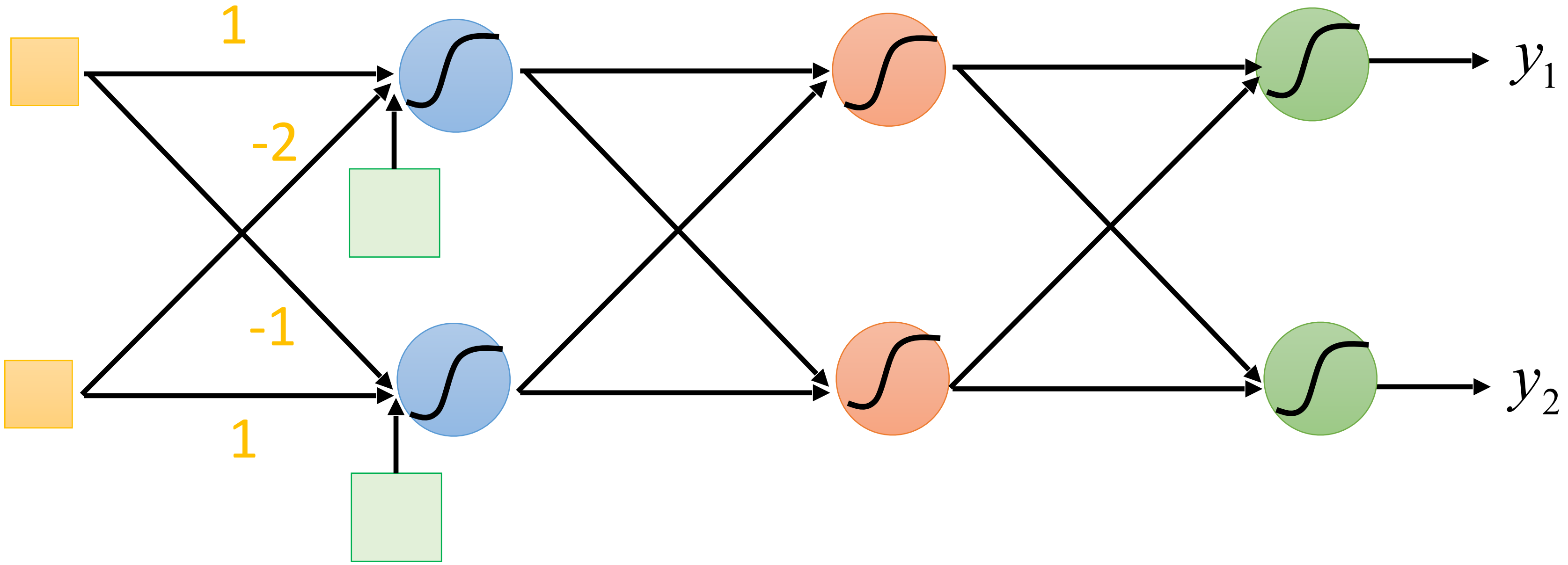
Matrix Operation



Matrix Operation

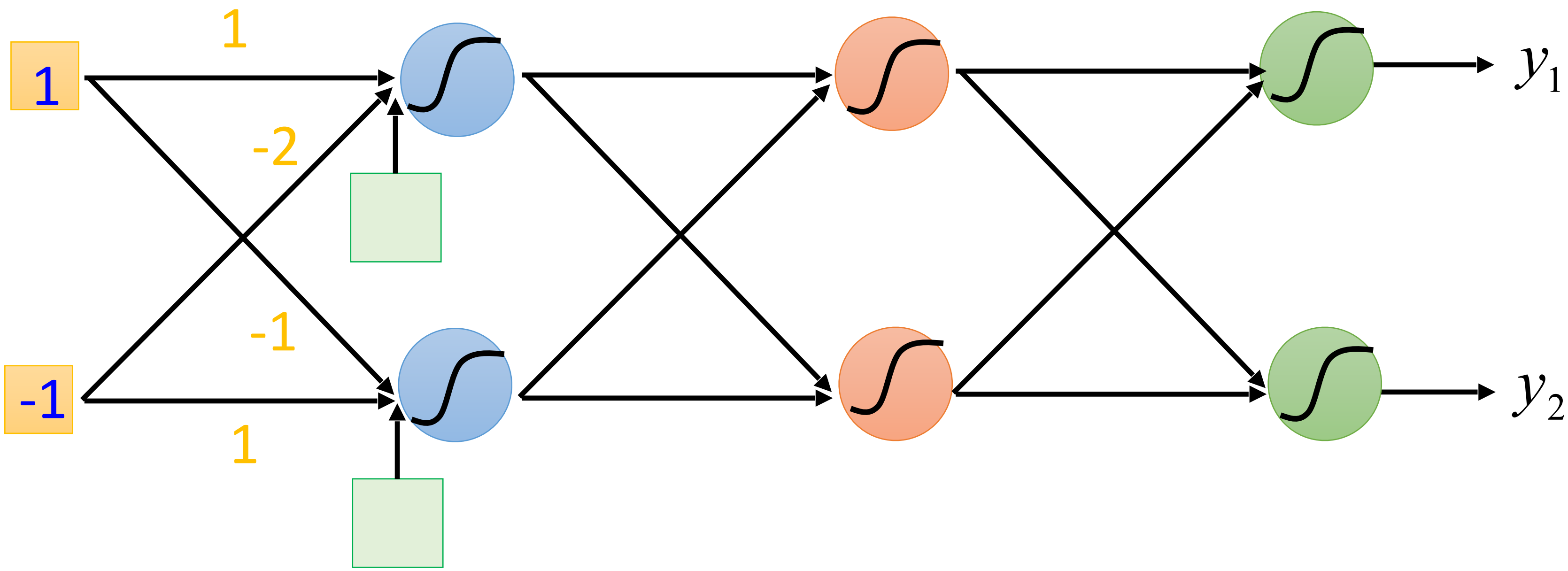


Matrix Operation



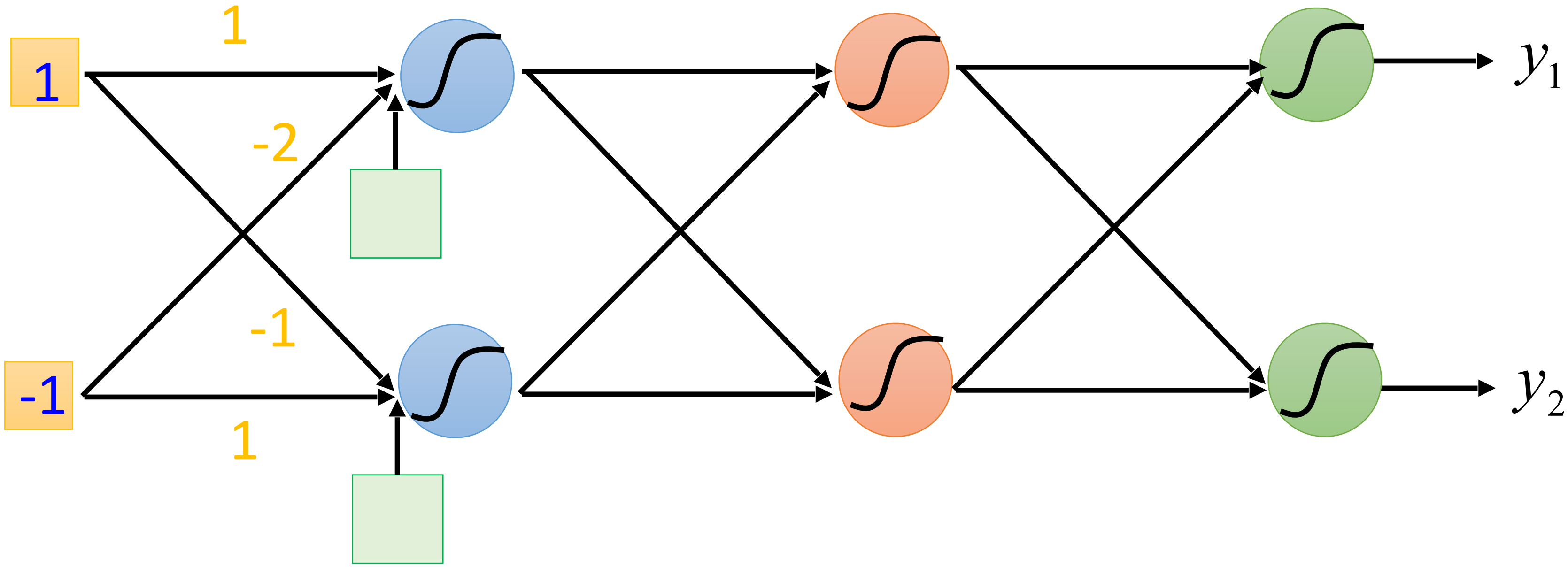
$$\begin{bmatrix} 1 & -2 \\ -1 & 1 \end{bmatrix}$$

Matrix Operation



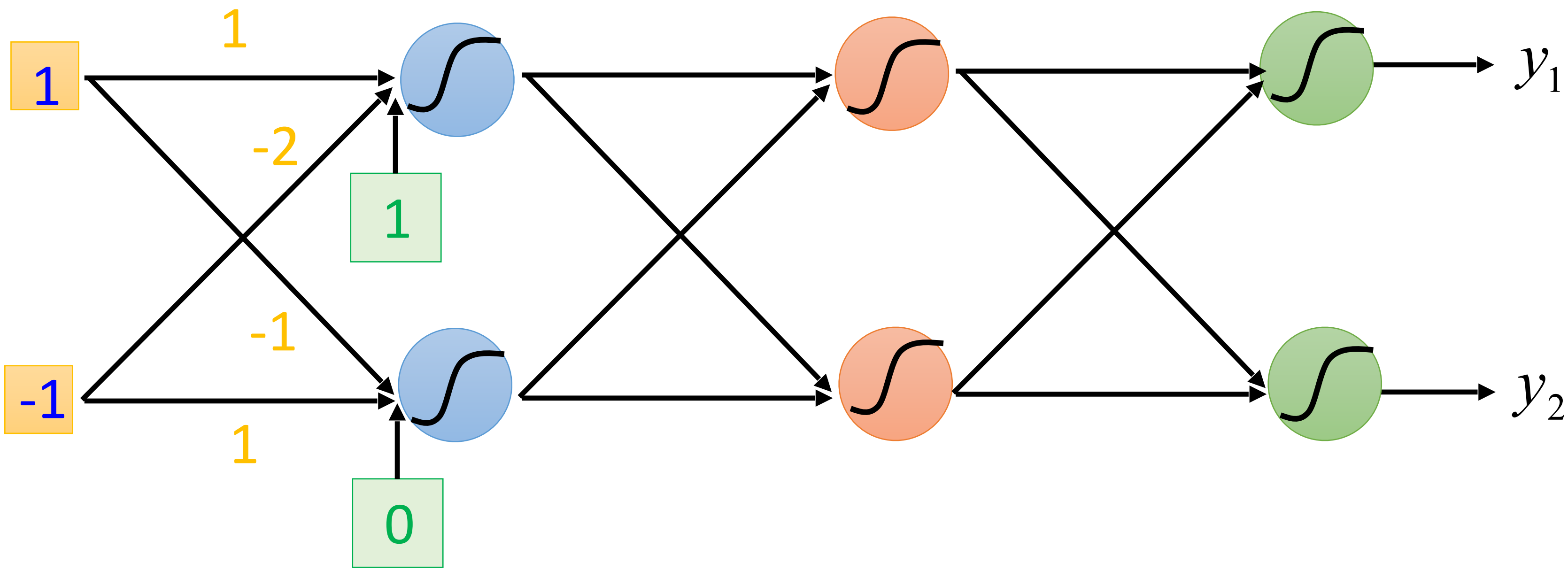
$$\begin{bmatrix} 1 & -2 \\ -1 & 1 \end{bmatrix}$$

Matrix Operation



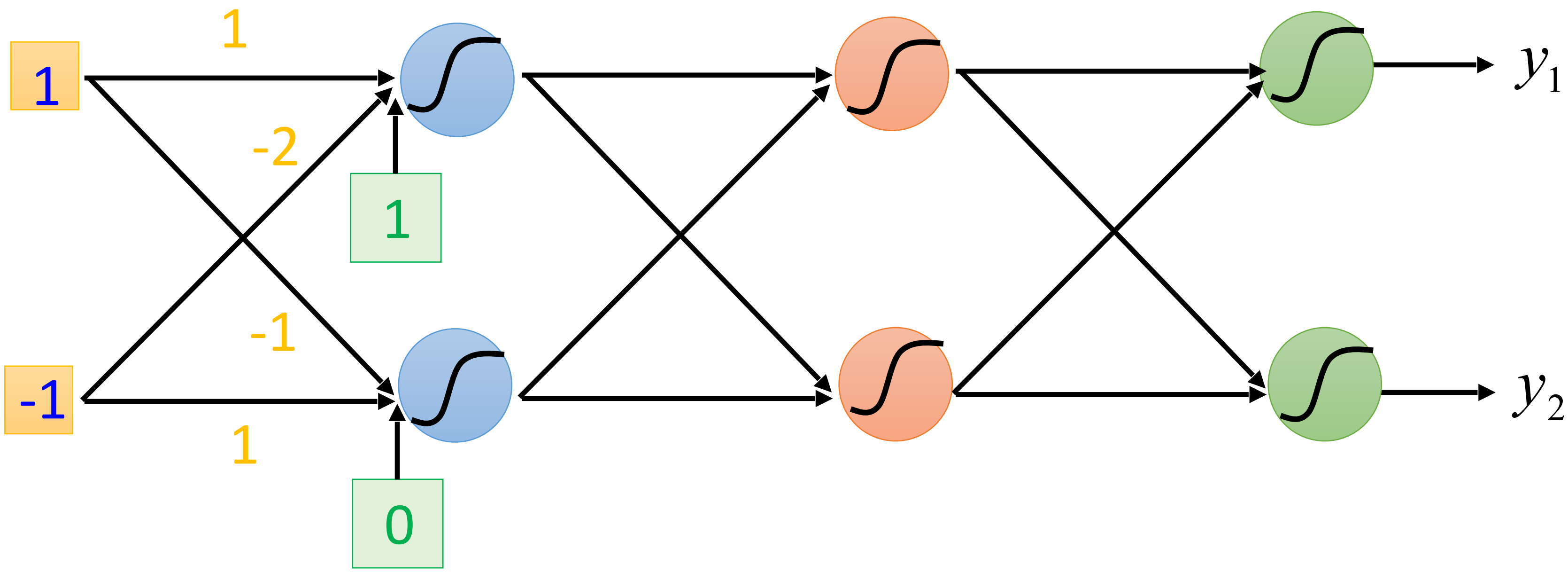
$$\begin{bmatrix} 1 & -2 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

Matrix Operation



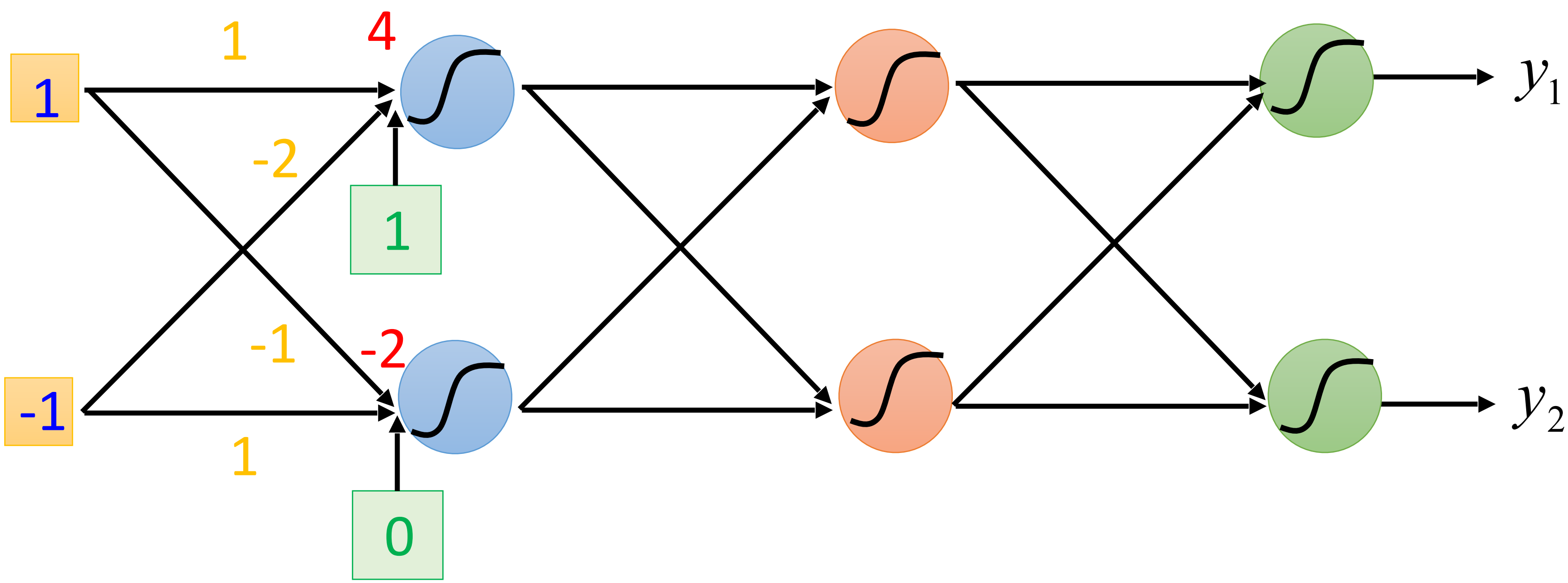
$$\begin{bmatrix} 1 & -2 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

Matrix Operation



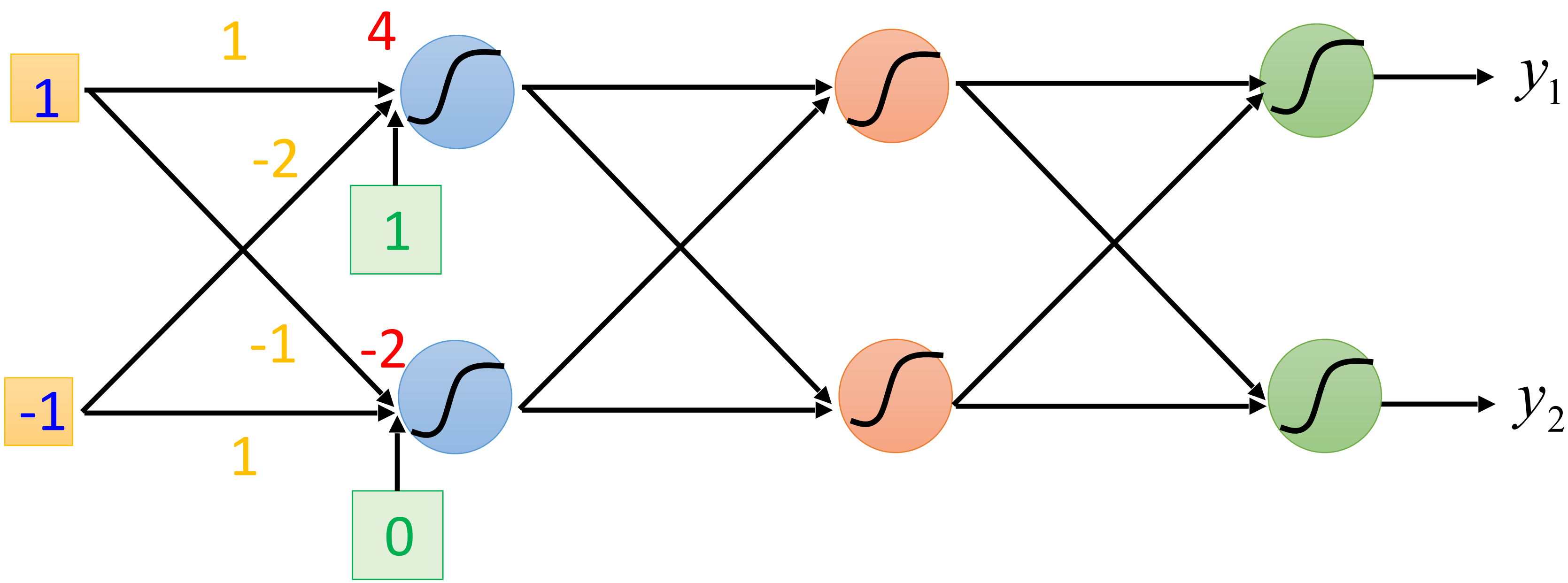
$$\begin{bmatrix} 1 & -2 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Matrix Operation



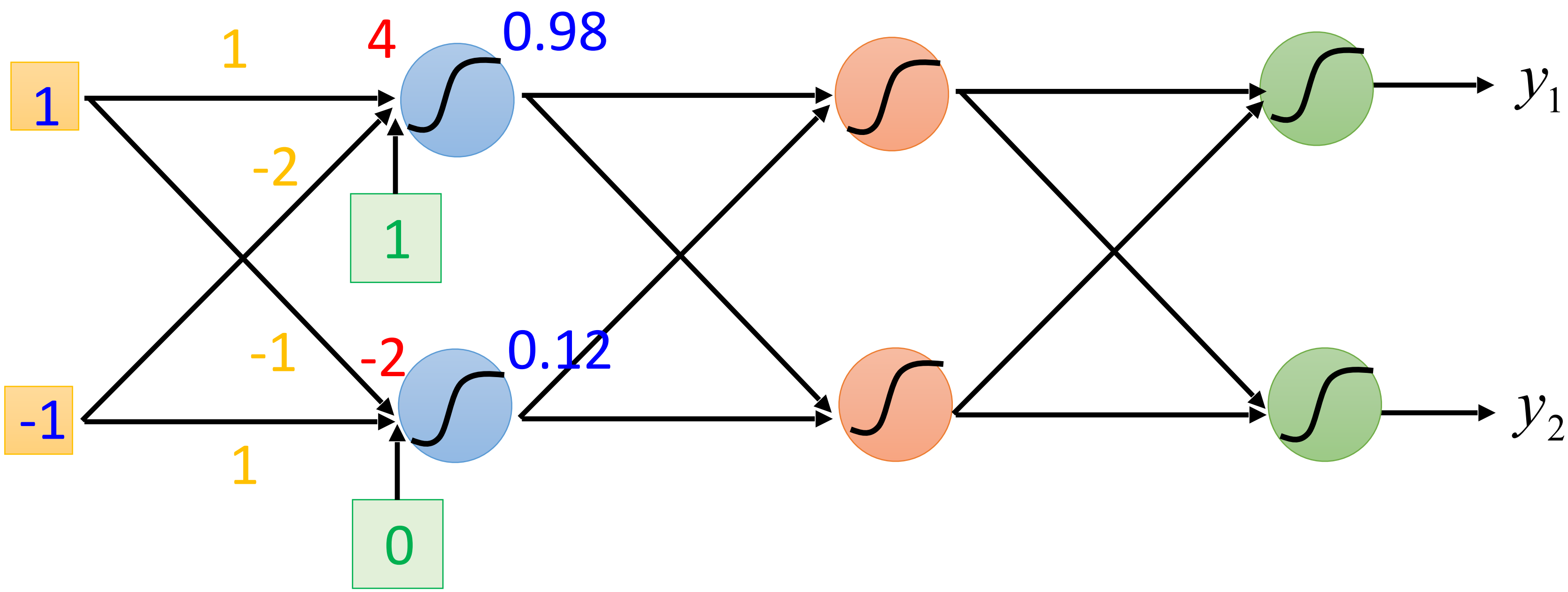
$$\underbrace{\begin{bmatrix} 1 & -2 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix}}_{\begin{bmatrix} 4 \\ -2 \end{bmatrix}}$$

Matrix Operation



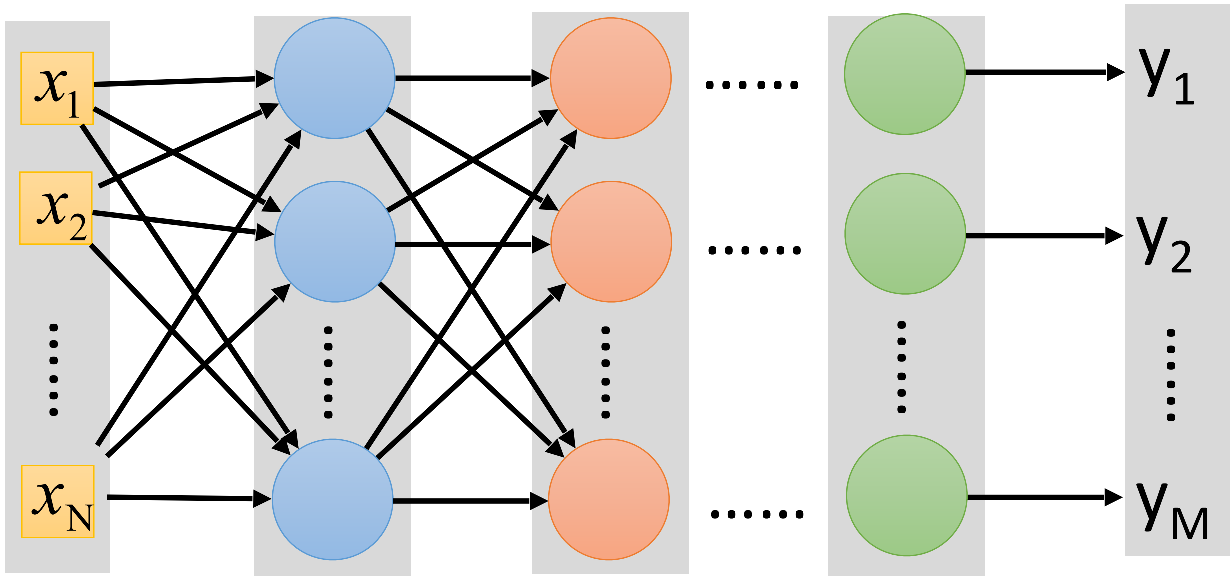
$$\underbrace{\sigma\left(\begin{bmatrix} 1 & -2 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right)}_{\begin{bmatrix} 4 \\ -2 \end{bmatrix}}$$

Matrix Operation

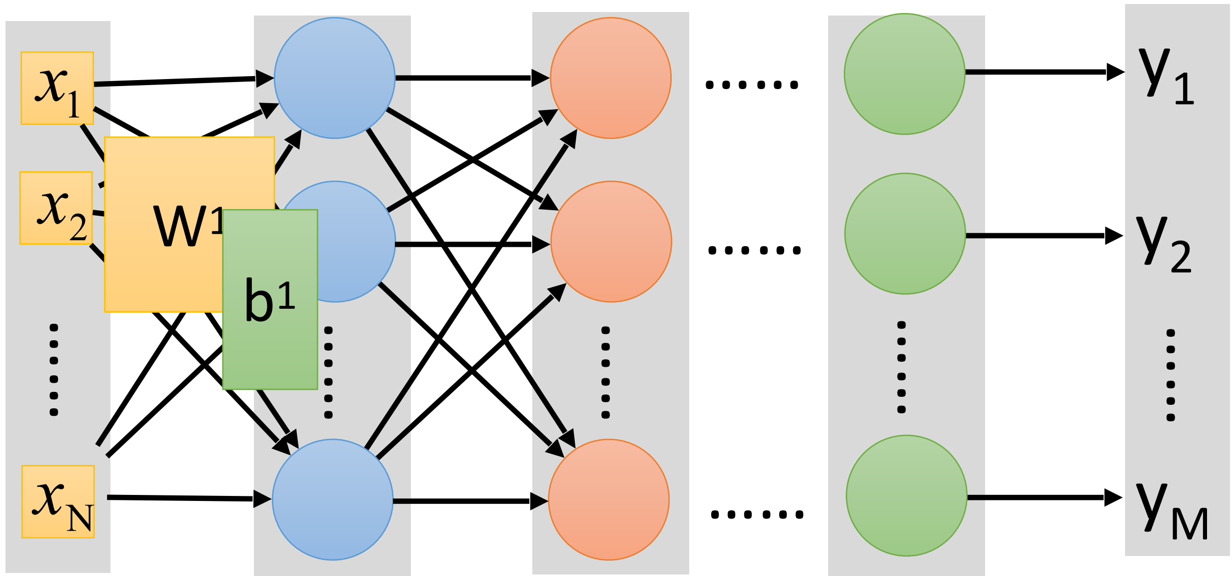


$$\sigma\left(\underbrace{\begin{bmatrix}1 & -2\\ -1 & 1\end{bmatrix}\begin{bmatrix}1\\ -1\end{bmatrix}+\begin{bmatrix}1\\ 0\end{bmatrix}}_{\begin{bmatrix}4\\ -2\end{bmatrix}}\right)=\begin{bmatrix}0.98\\ 0.12\end{bmatrix}$$

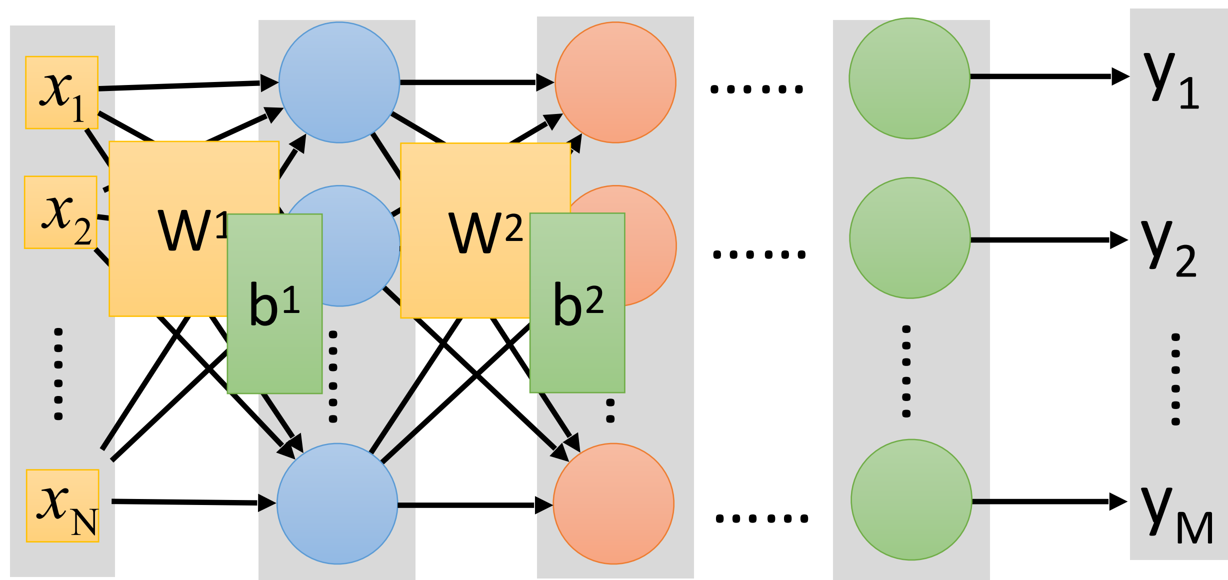
Neural Network



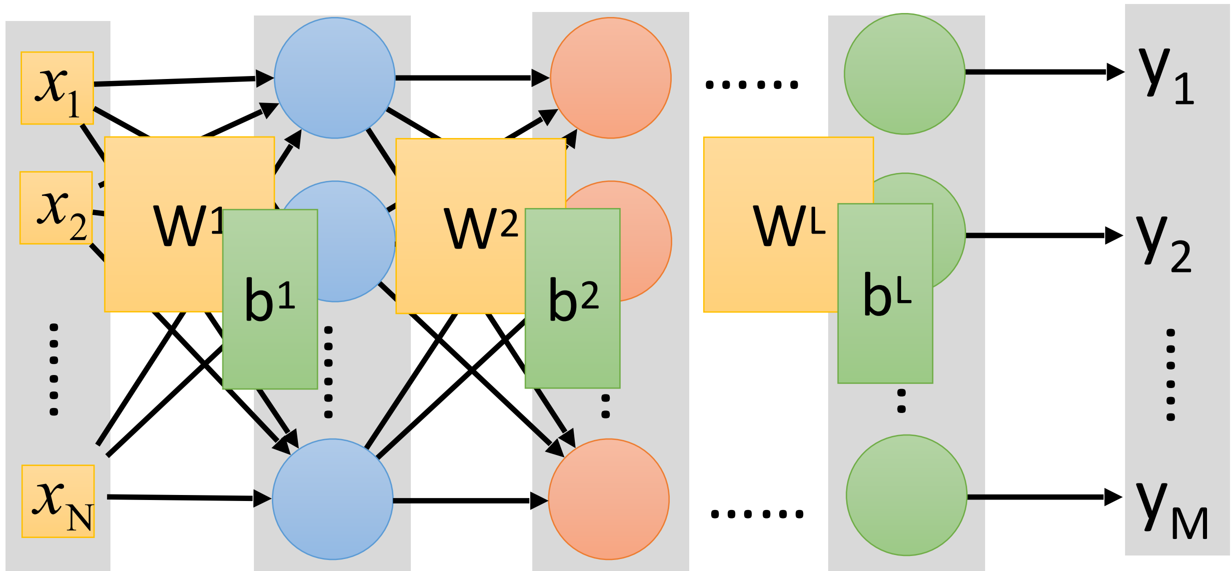
Neural Network



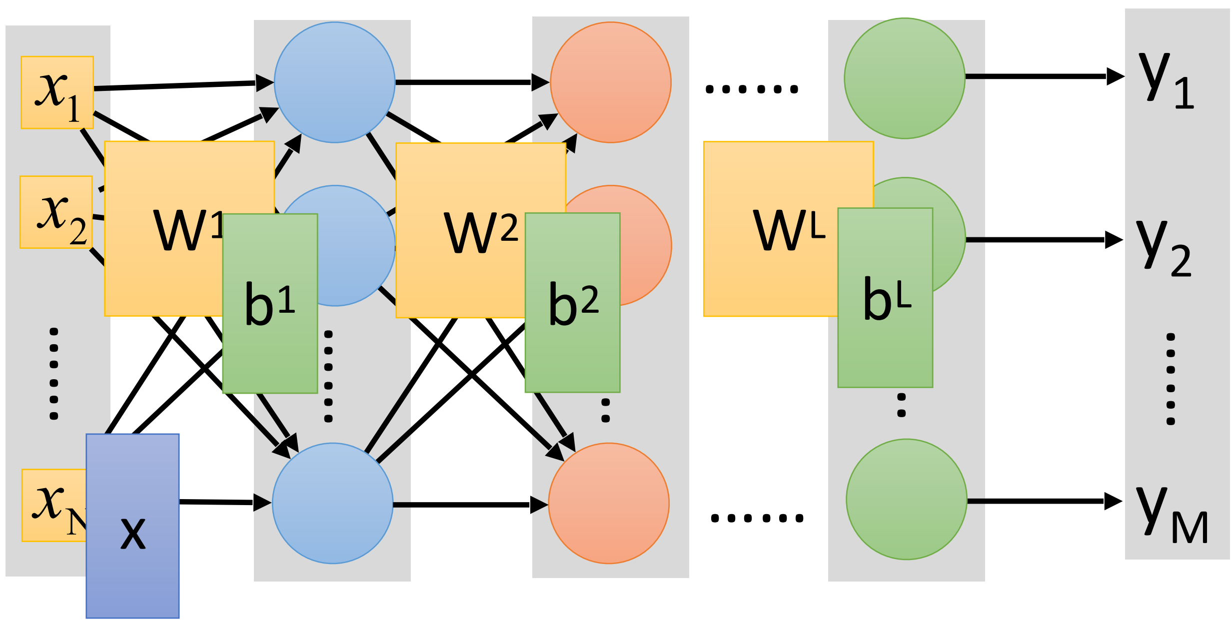
Neural Network



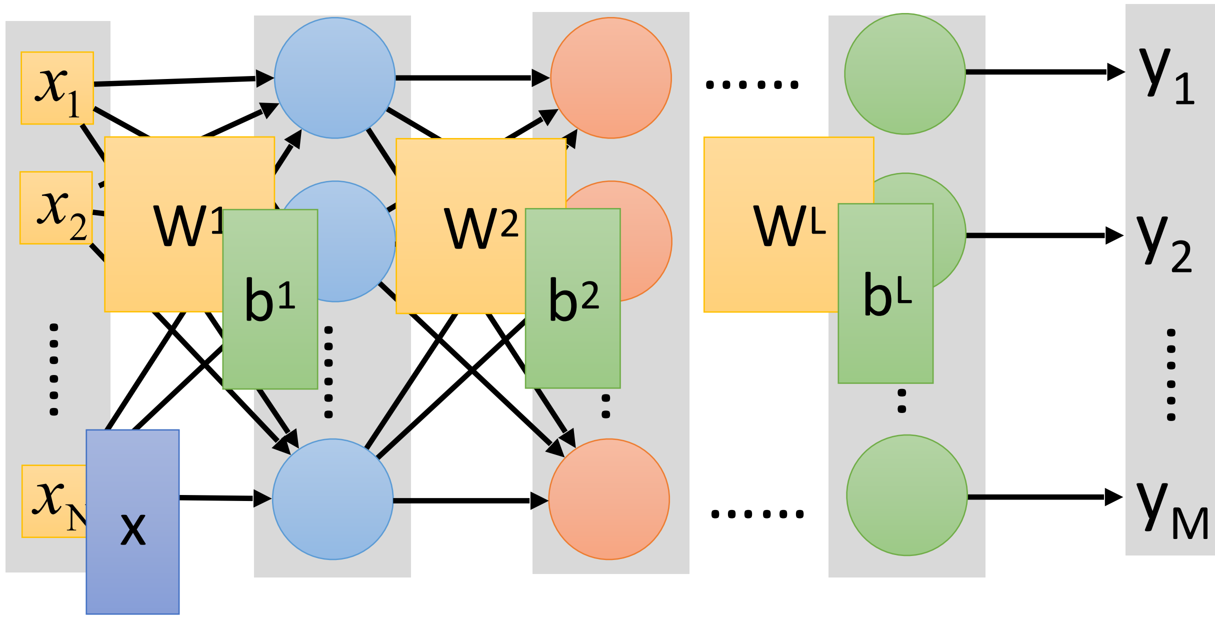
Neural Network



Neural Network

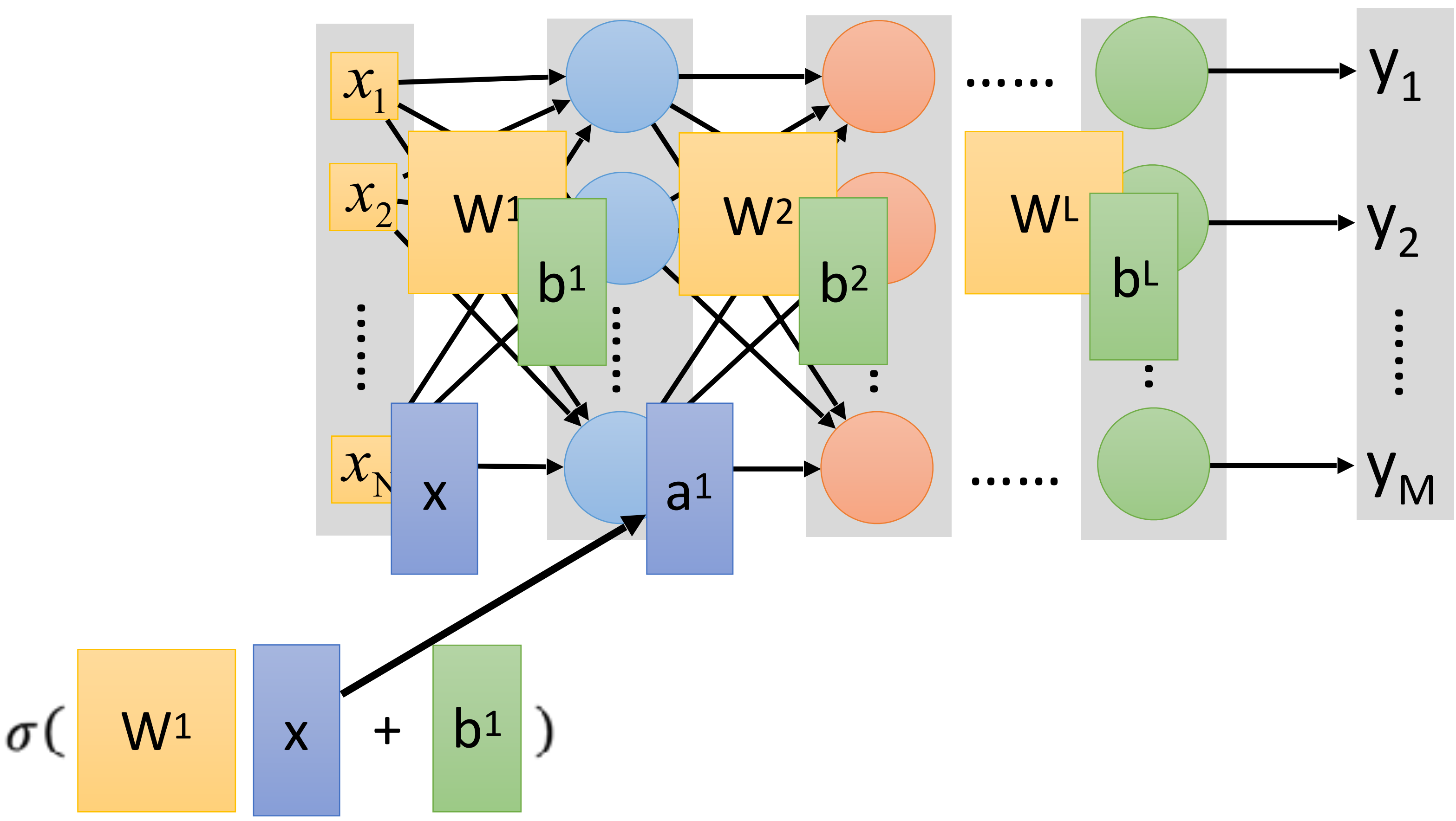


Neural Network

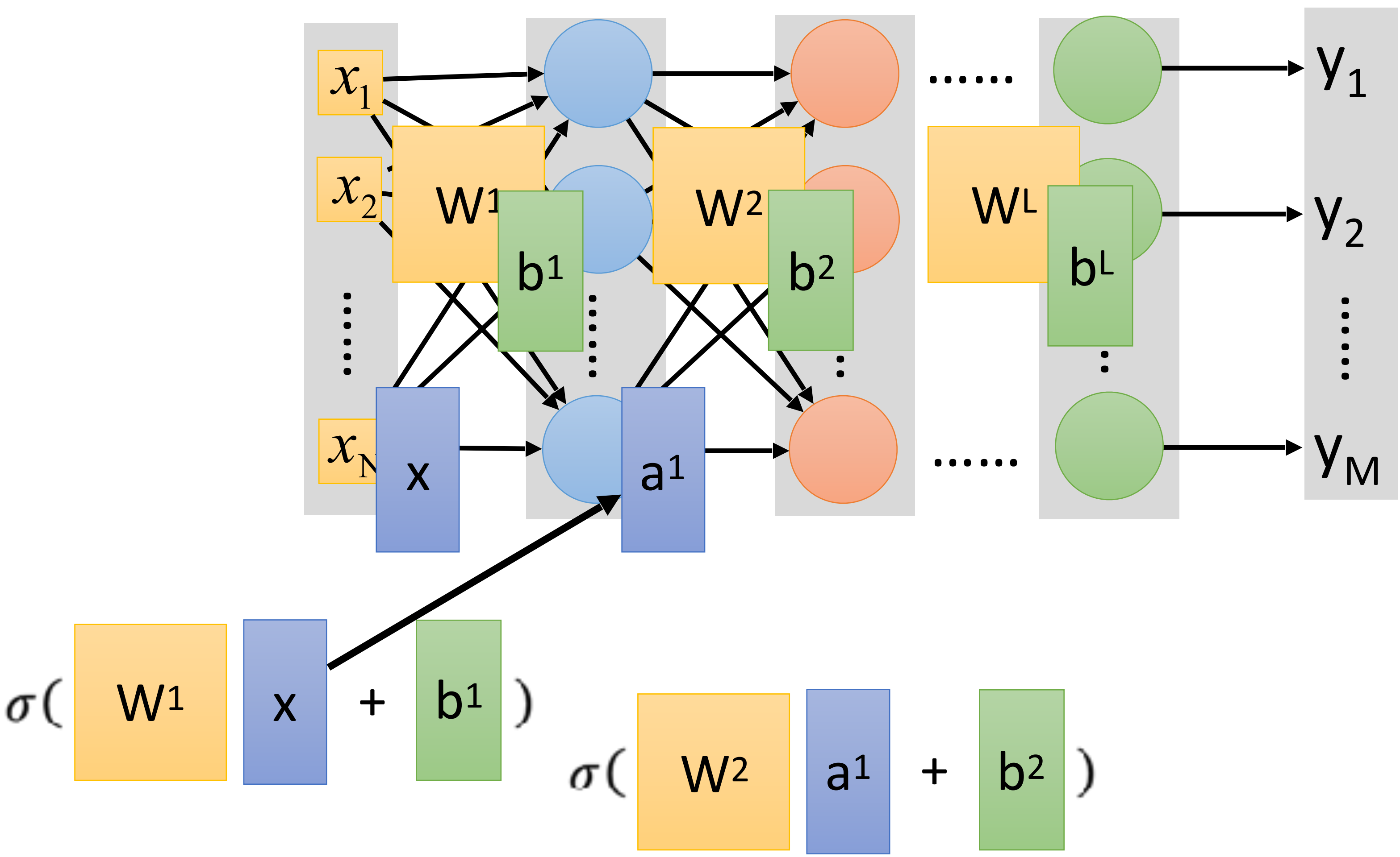


$$\sigma(W^1 \; x + b^1)$$

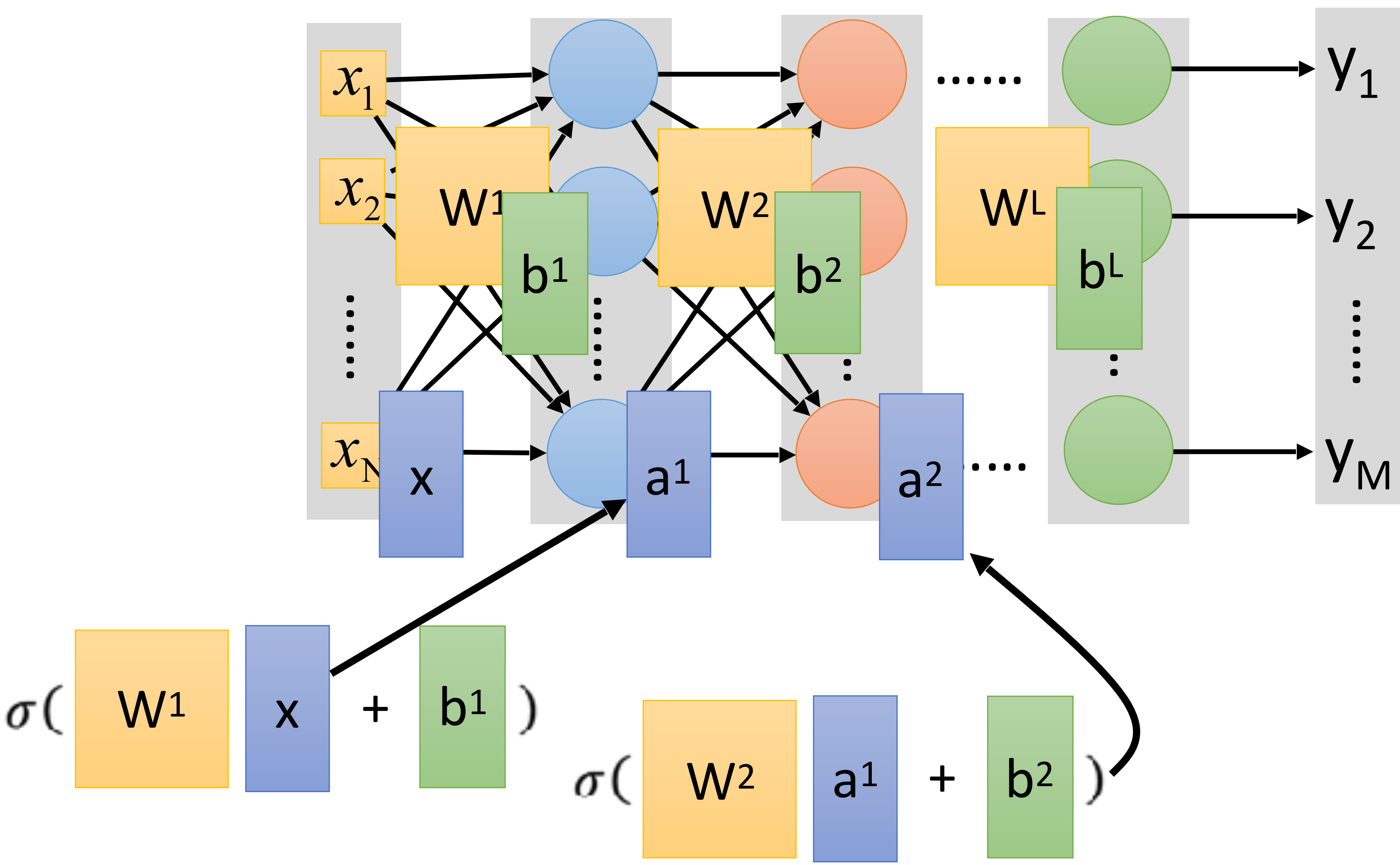
Neural Network



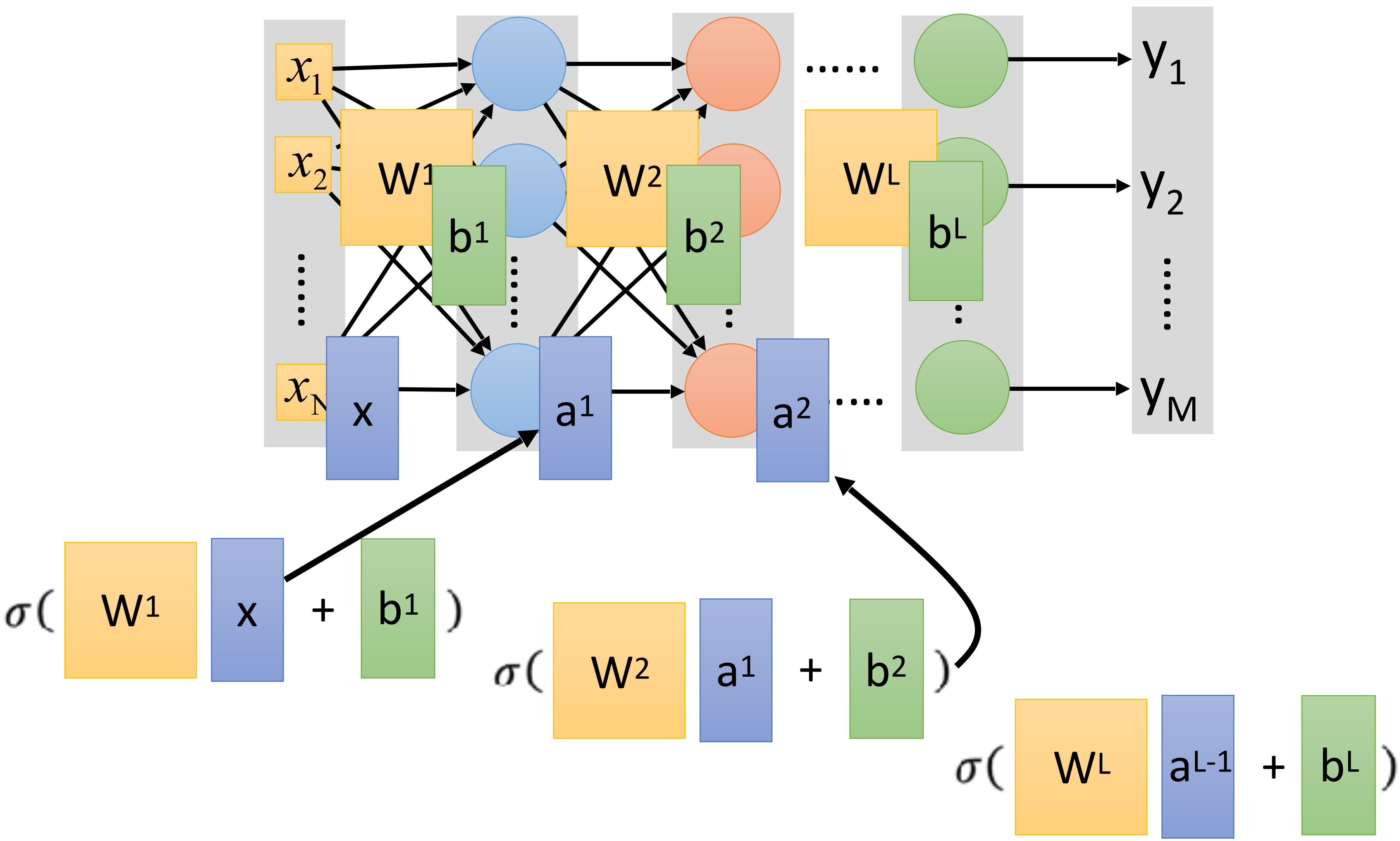
Neural Network



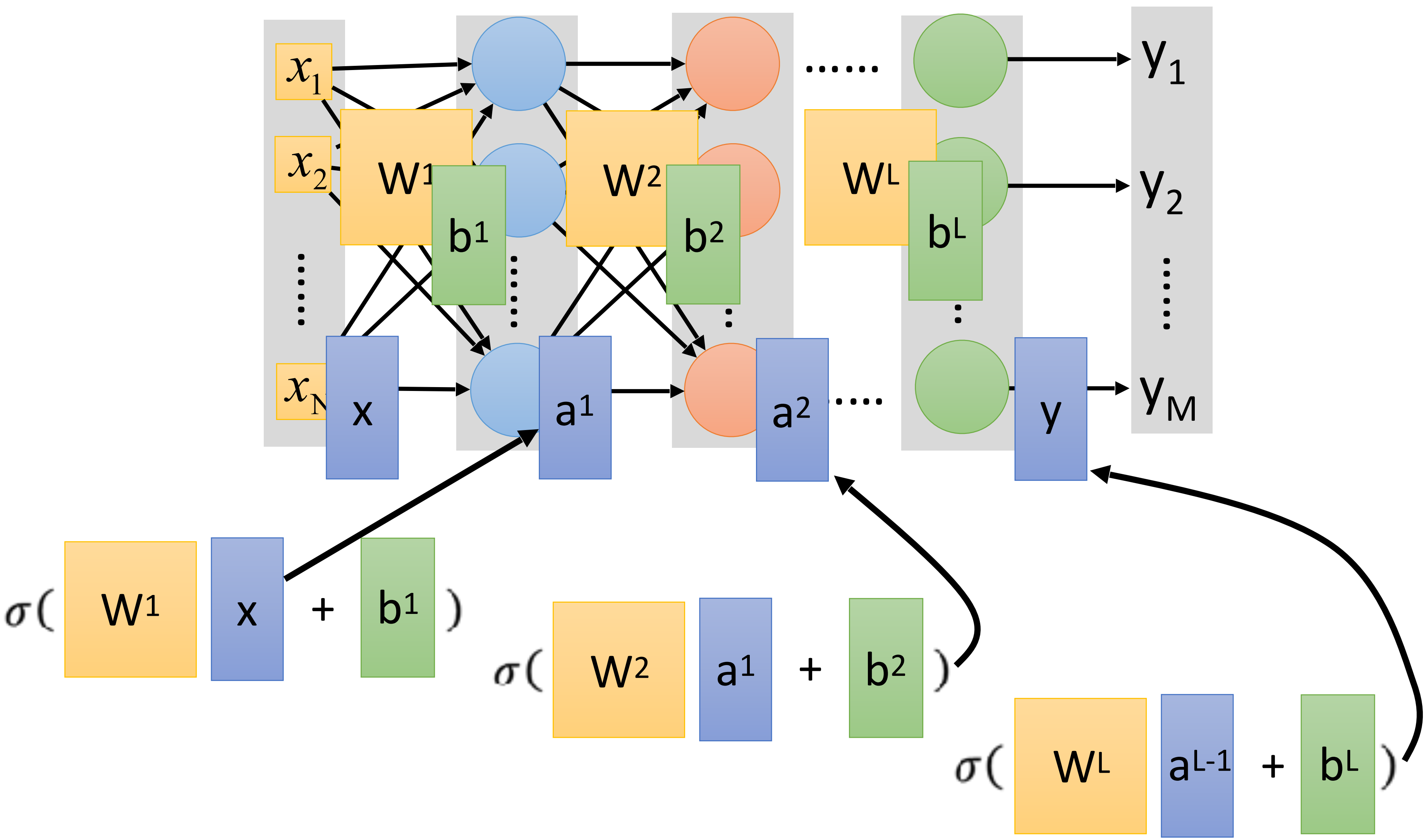
Neural Network



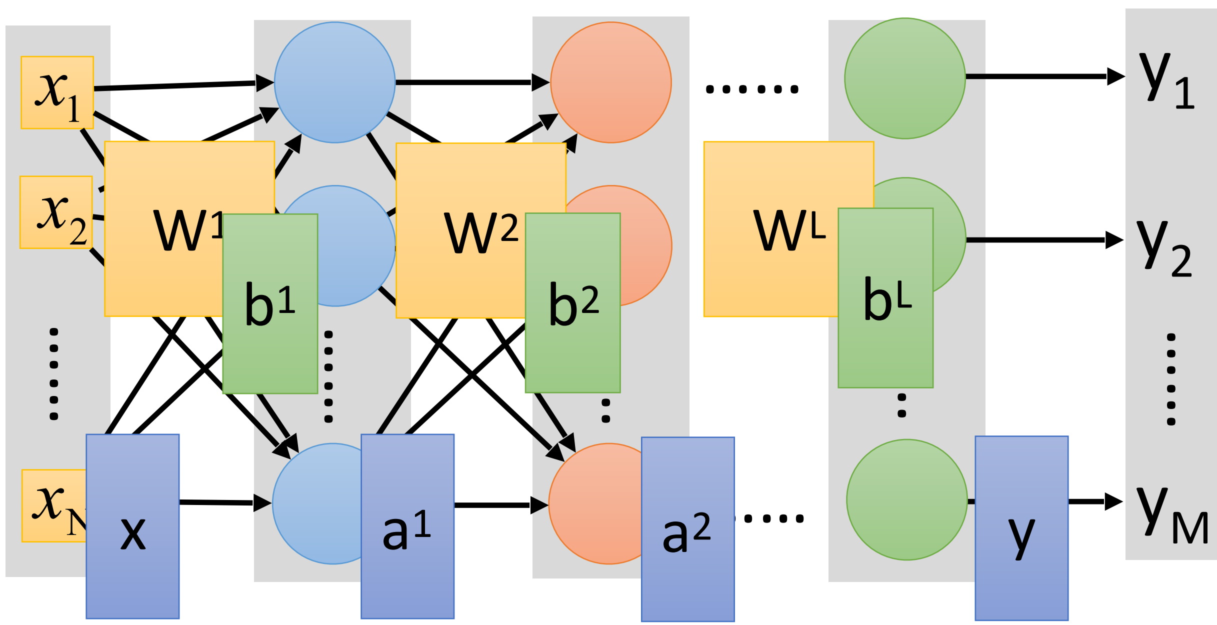
Neural Network



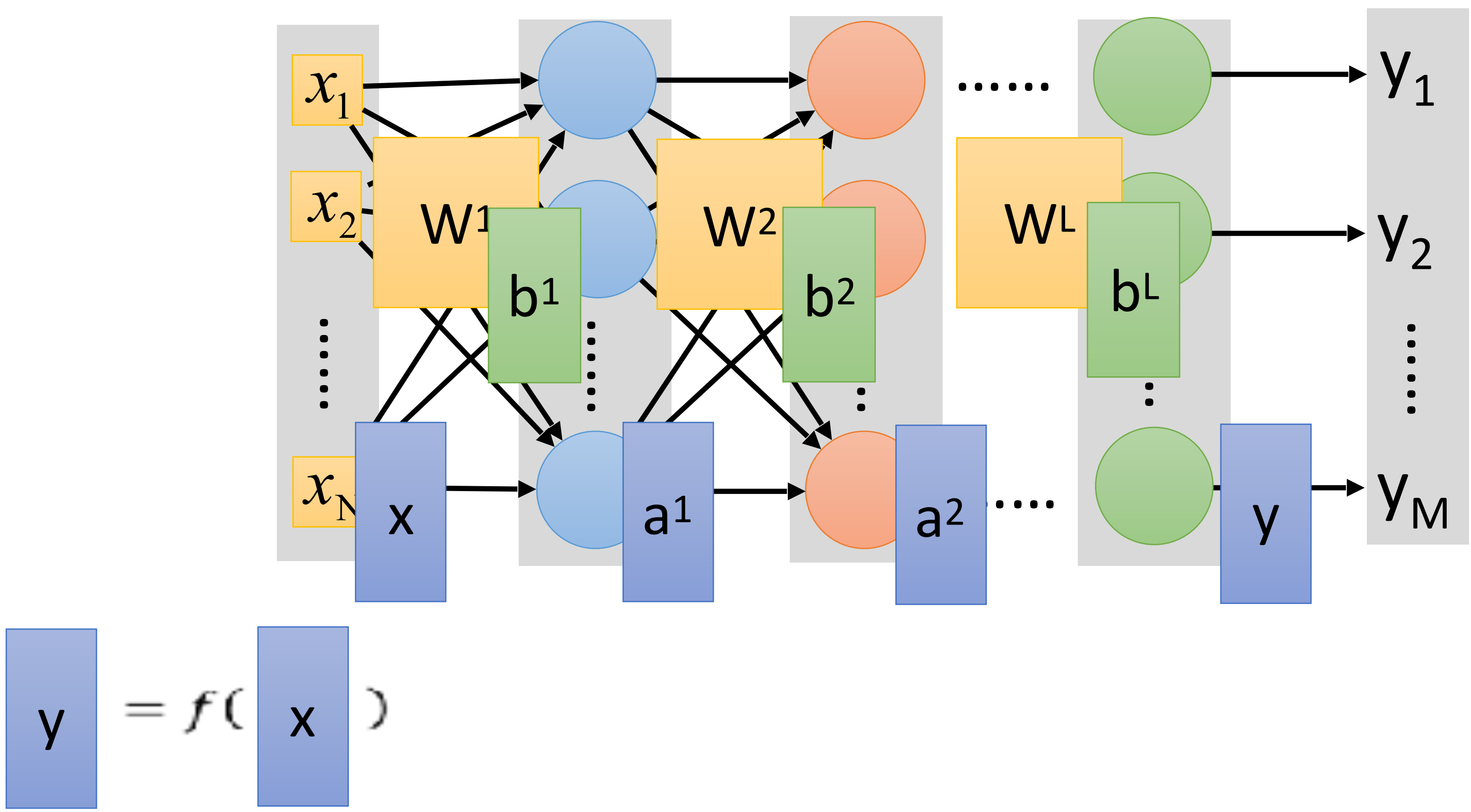
Neural Network



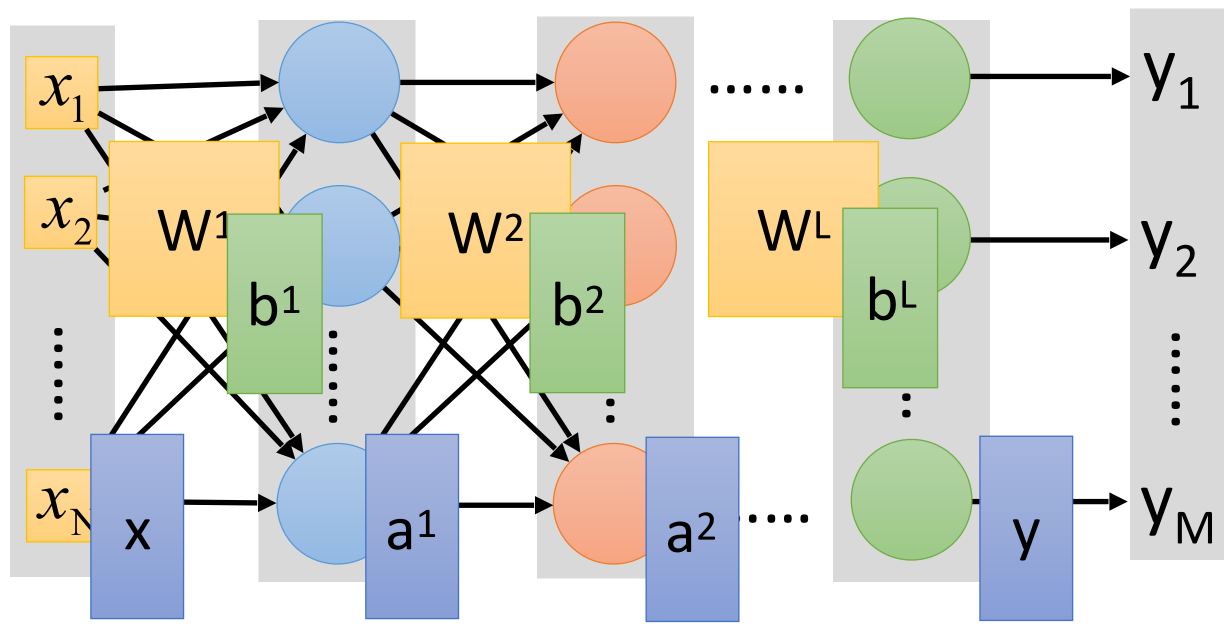
Neural Network



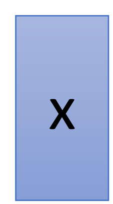
Neural Network



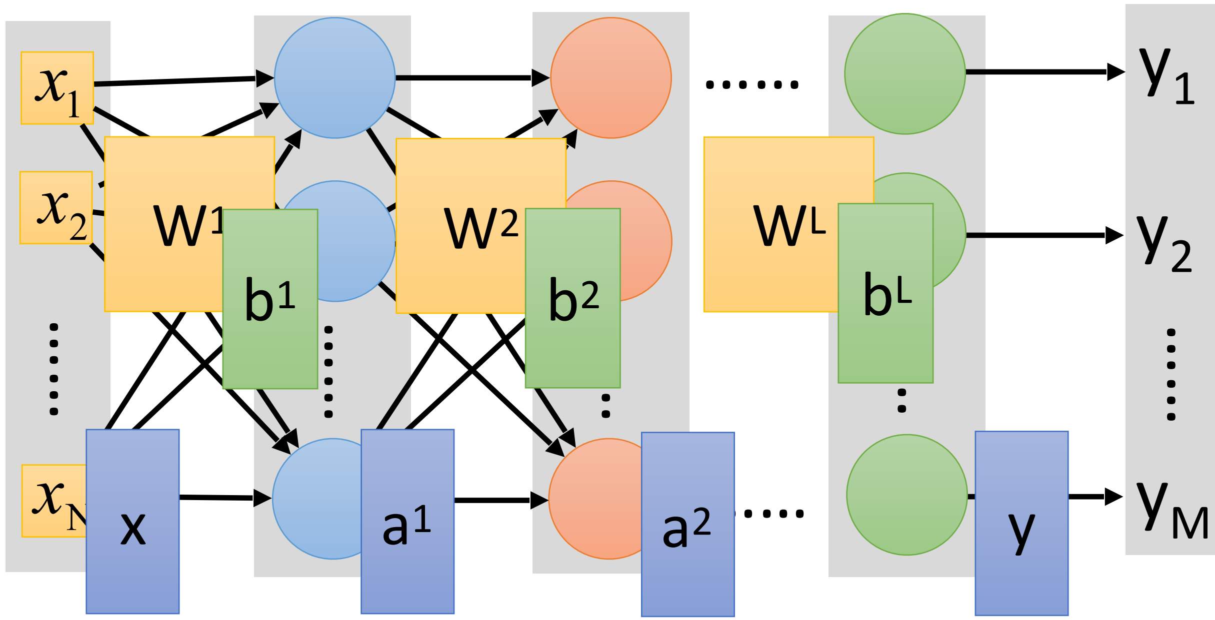
Neural Network



$$y = f(x)$$



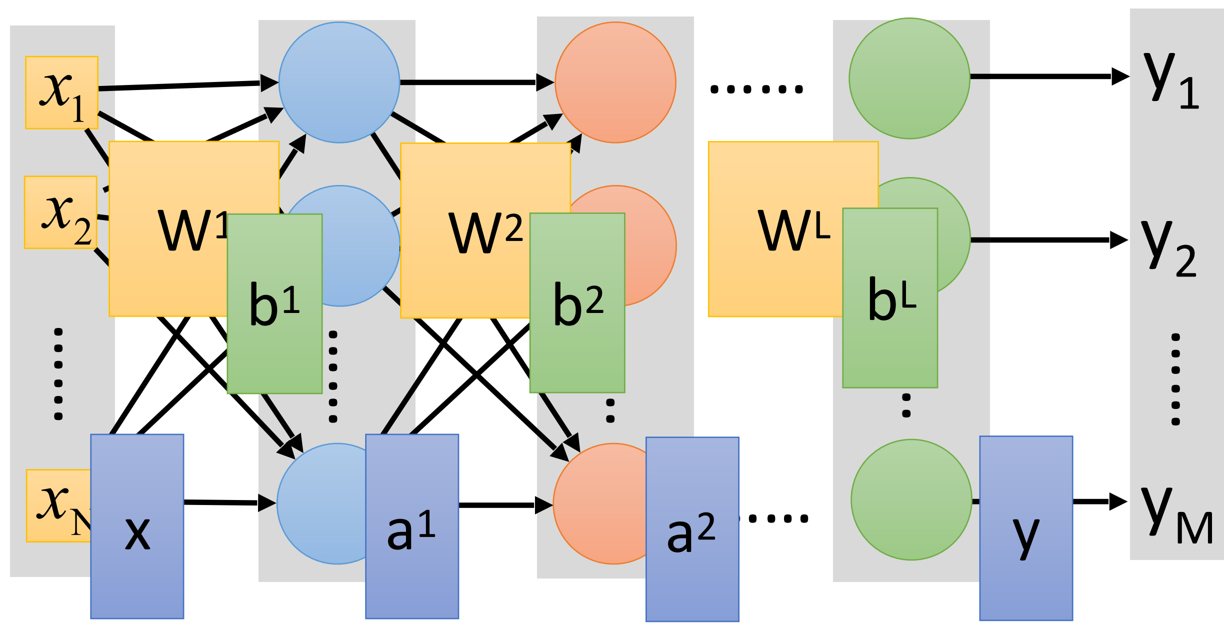
Neural Network



$$y = f(x)$$

$$\sigma(W^1 \; x + b^1)$$

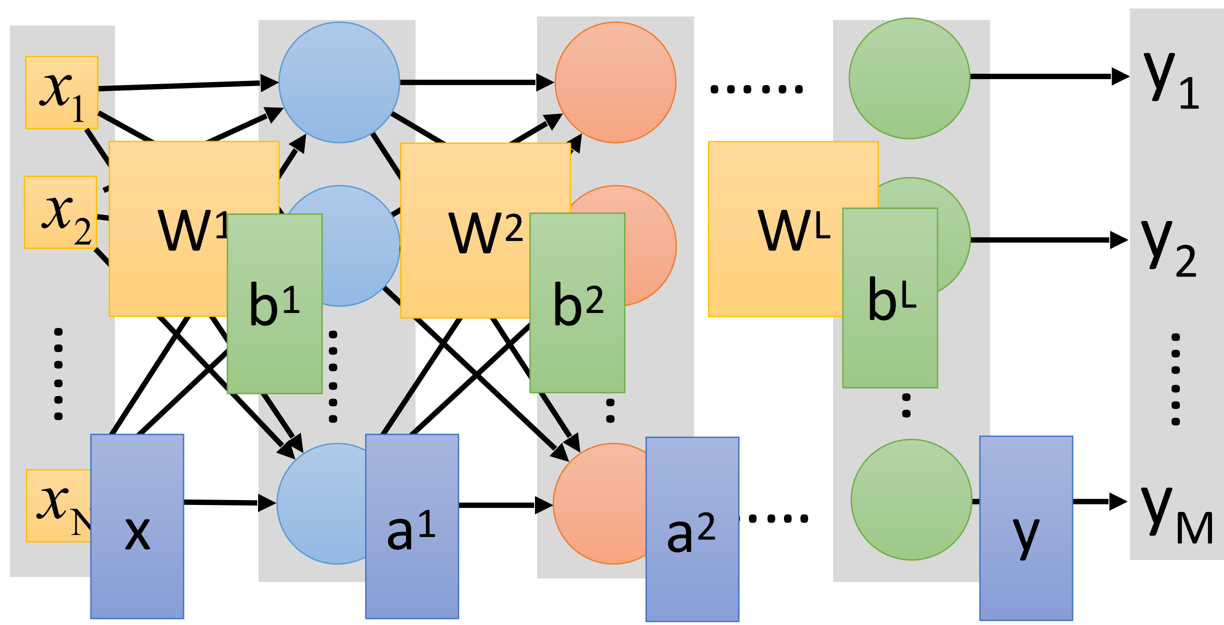
Neural Network



$$y = f(x)$$

$$\sigma(W^2 \; \sigma(W^1 \; x + b^1) + b^2)$$

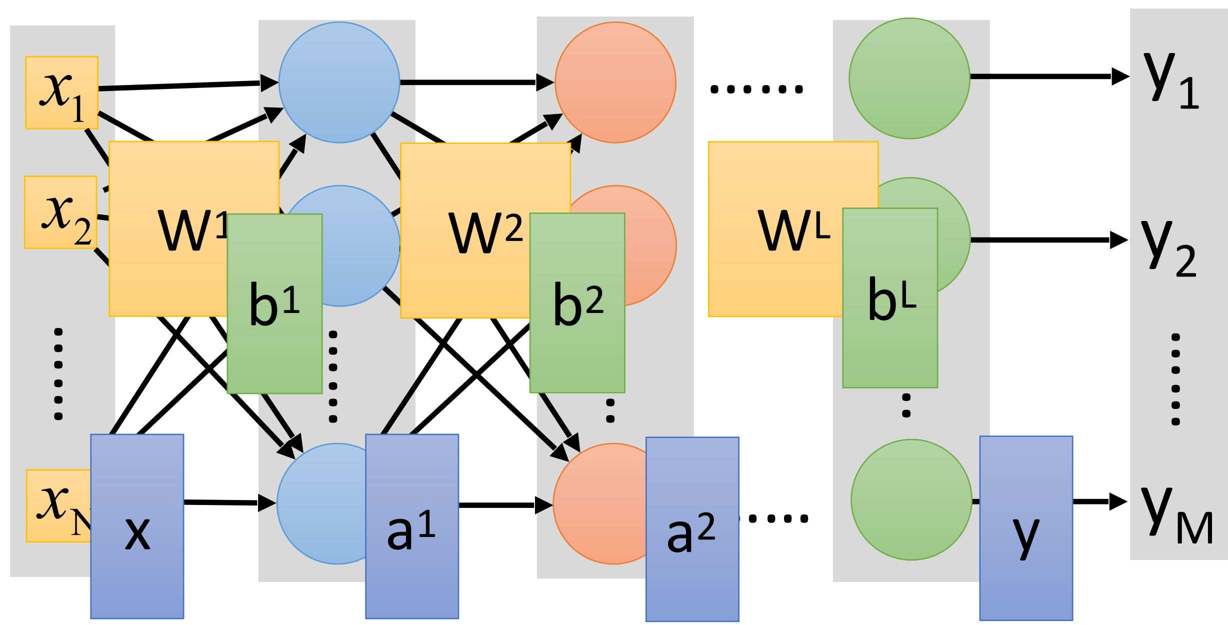
Neural Network



$$y = f(x)$$

$$= \sigma(W^L \ldots \sigma(W^2 \, \sigma(W^1 x + b^1) + b^2) \ldots + b^L)$$

Neural Network



$$y = f(x)$$

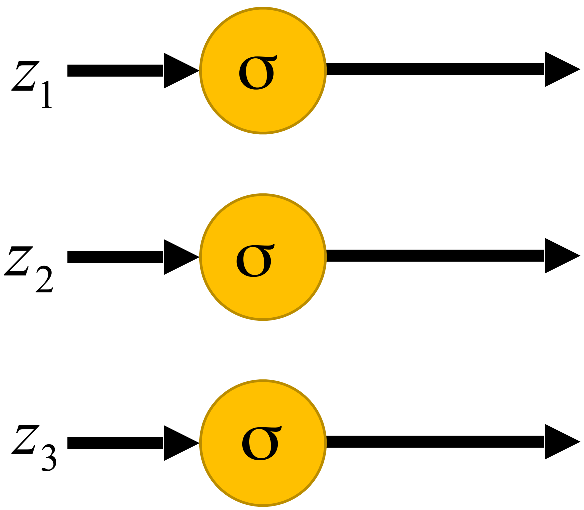
Using parallel computing techniques to speed up matrix operation

$$= \sigma(W^L \ldots \sigma(W^2 \, \sigma(W^1 x + b^1) + b^2) \ldots + b^L)$$

Softmax

- Softmax layer as the output layer

Ordinary Layer



$$y_1 = \sigma(z_1)$$

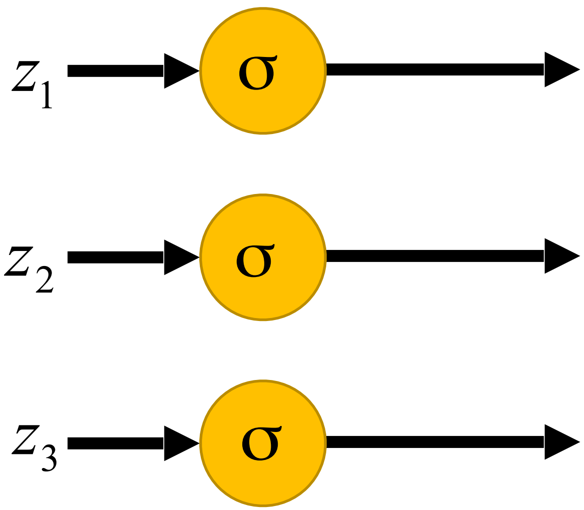
$$y_2 = \sigma(z_2)$$

$$y_3 = \sigma(z_3)$$

Softmax

- Softmax layer as the output layer

__Ordinary Layer__



$y_1 = \sigma(z_1)$

$y_2 = \sigma(z_2)$

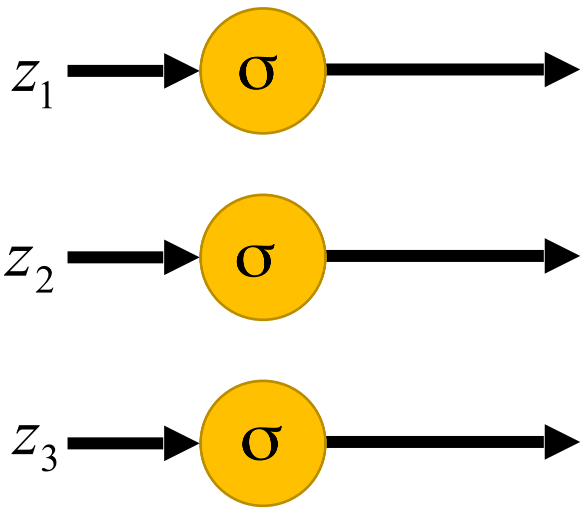
$y_3 = \sigma(z_3)$

In general, the output of network can be any value.

Softmax

- Softmax layer as the output layer

<u>Ordinary Layer</u>



$$y_1 = \sigma(z_1)$$

$$y_2 = \sigma(z_2)$$

$$y_3 = \sigma(z_3)$$

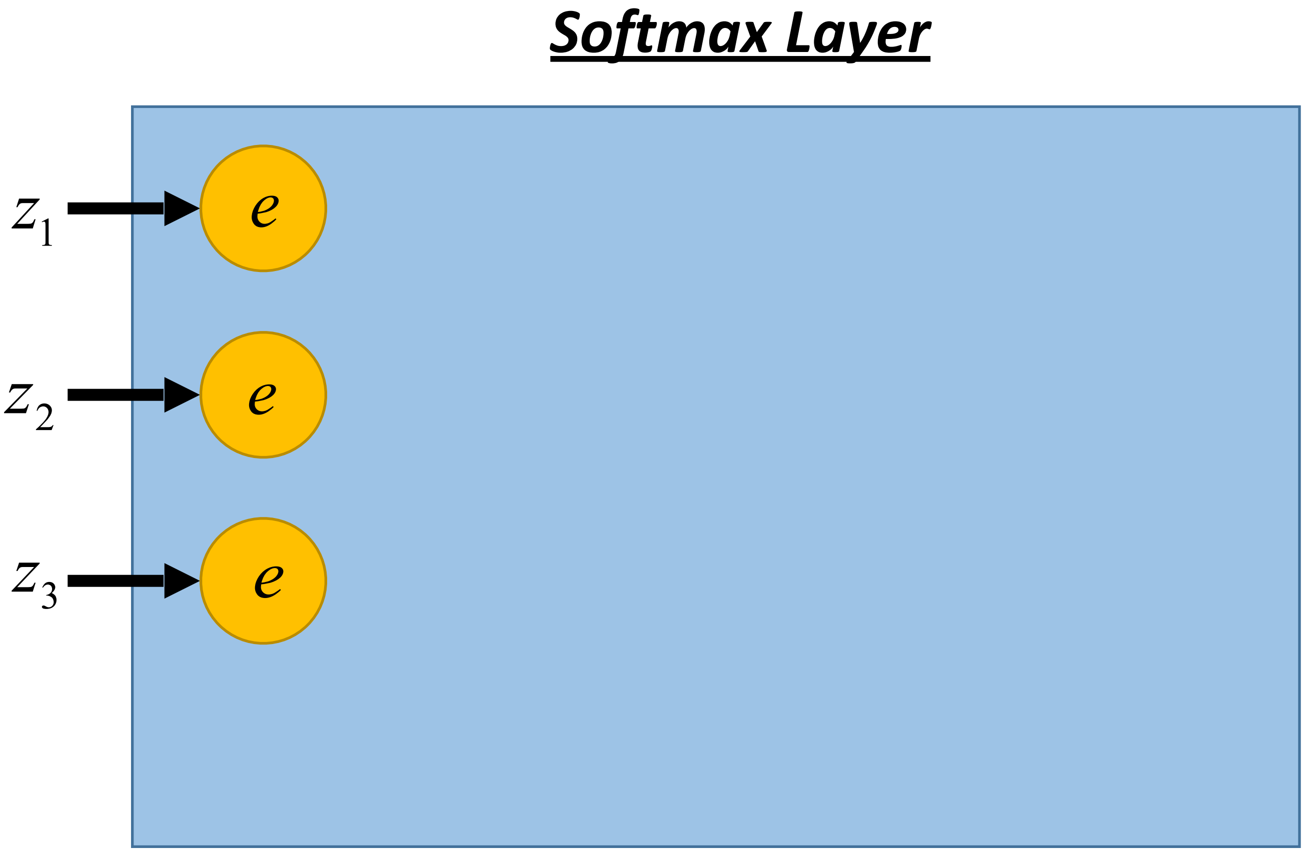
In general, the output of network can be any value.

May not be easy to interpret

Softmax

Probability:
- $1 > y_i > 0$
- $\sum_i y_i = 1$

- Softmax layer as the output layer

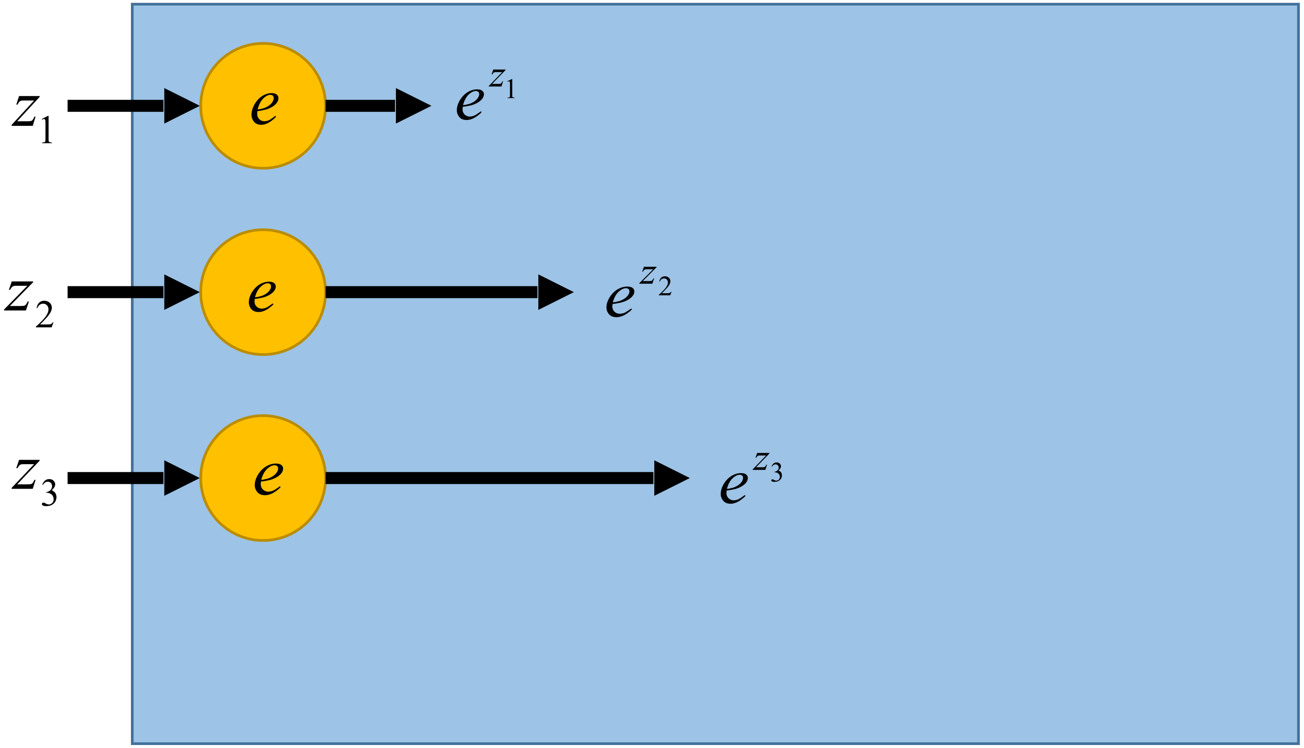


Softmax

Probability:
- $1 > y_i > 0$
- $\sum_i y_i = 1$

- Softmax layer as the output layer

Softmax Layer

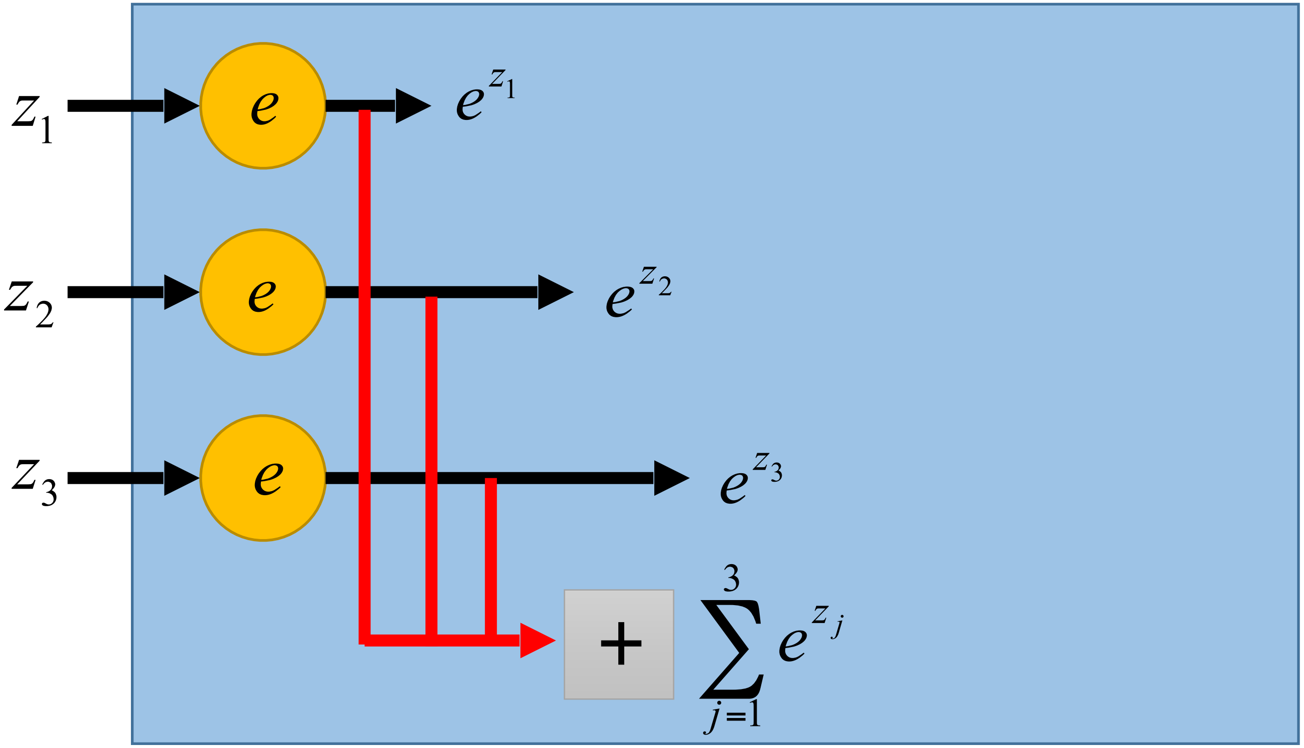


Softmax

Probability:
- $1 > y_i > 0$
- $\sum_i y_i = 1$

- Softmax layer as the output layer

Softmax Layer

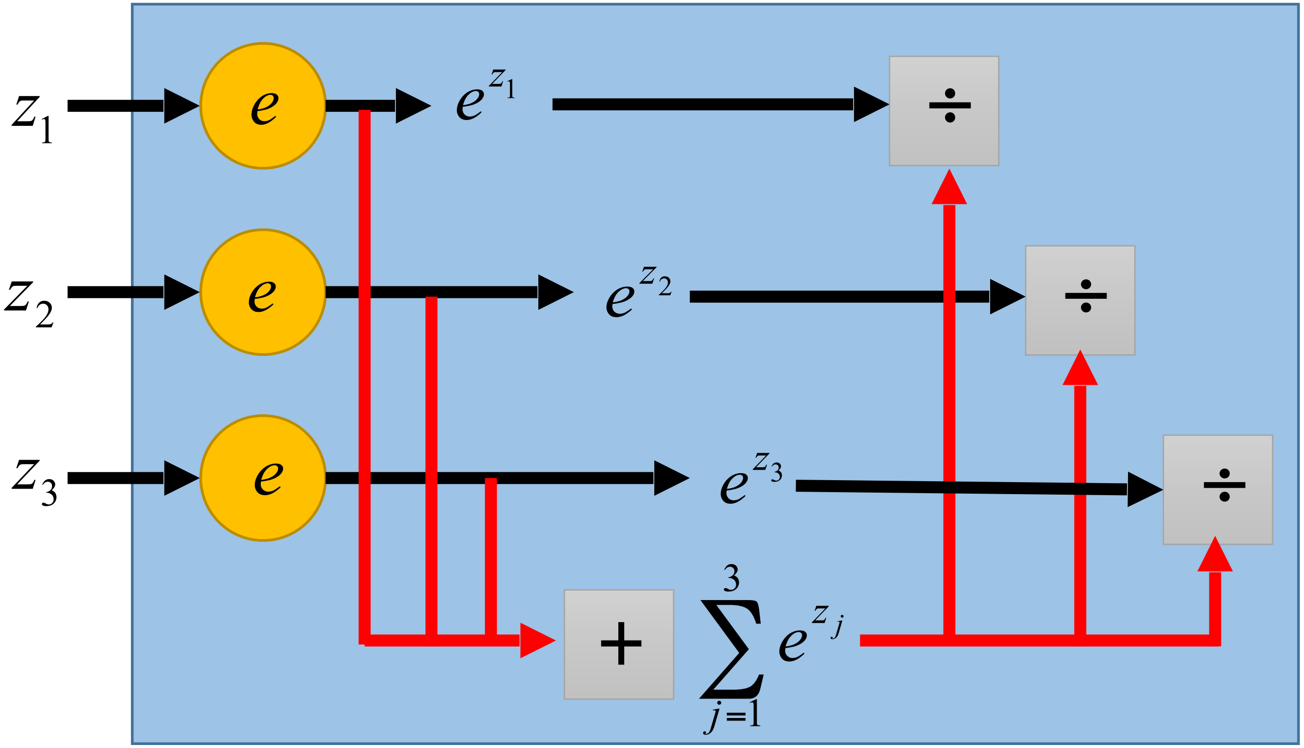


Softmax

Probability:
- $1 > y_i > 0$
- $\sum_i y_i = 1$

- Softmax layer as the output layer

Softmax Layer

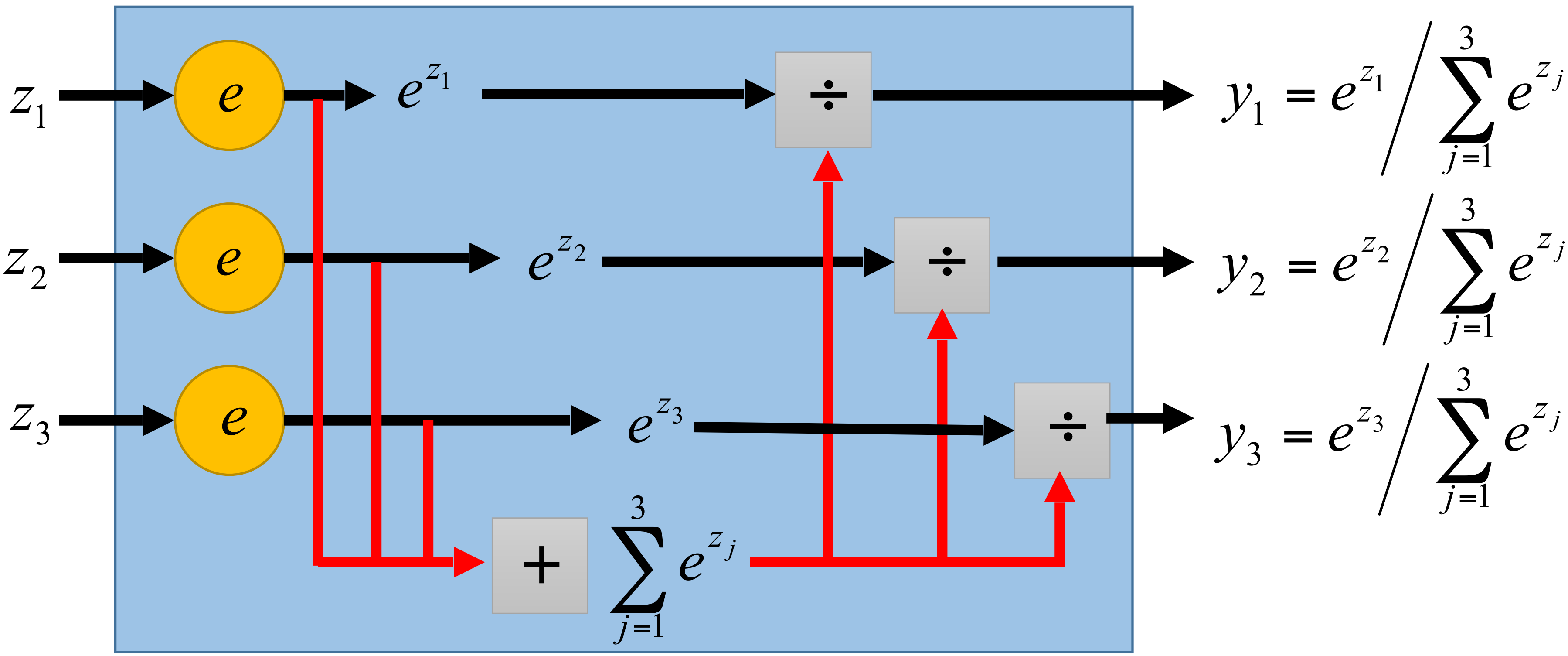


Softmax

***Probability*:**
- $1 > y_i > 0$
- $\sum_i y_i = 1$

- Softmax layer as the output layer

Softmax Layer

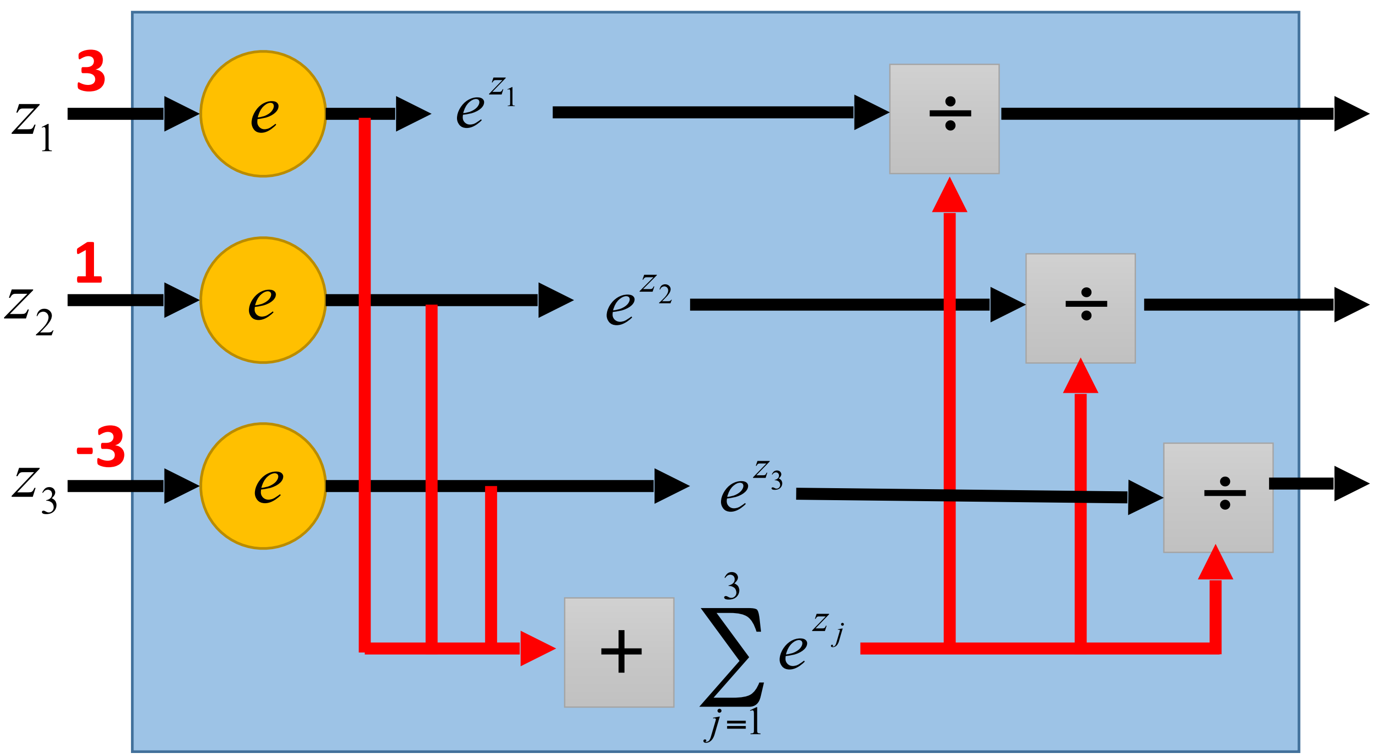


Softmax

Probability:
- $1 > y_i > 0$
- $\sum_i y_i = 1$

- Softmax layer as the output layer

Softmax Layer



$$y_1 = e^{z_1} \Big/ \sum_{j=1}^{3} e^{z_j}$$

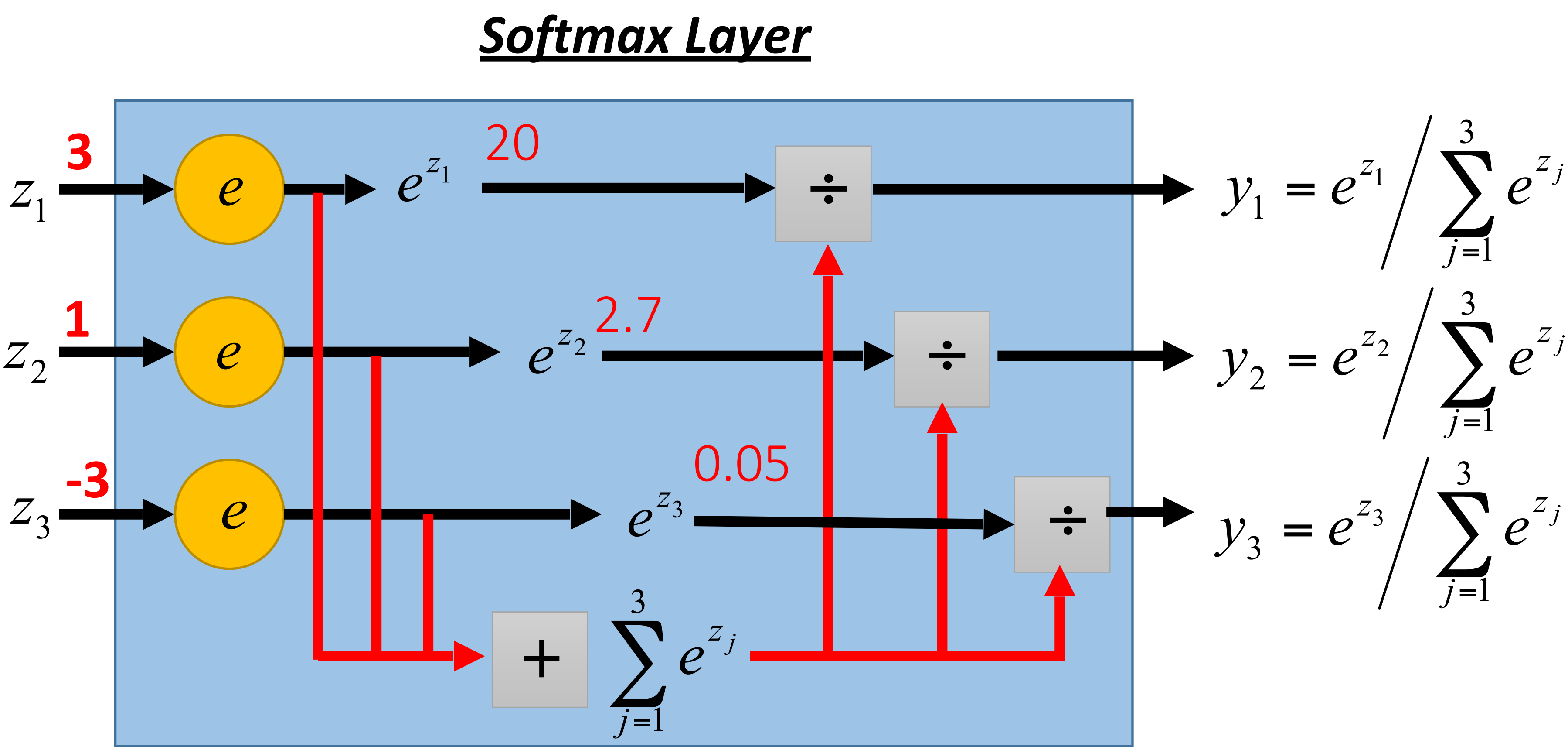
$$y_2 = e^{z_2} \Big/ \sum_{j=1}^{3} e^{z_j}$$

$$y_3 = e^{z_3} \Big/ \sum_{j=1}^{3} e^{z_j}$$

Softmax

• Softmax layer as the output layer

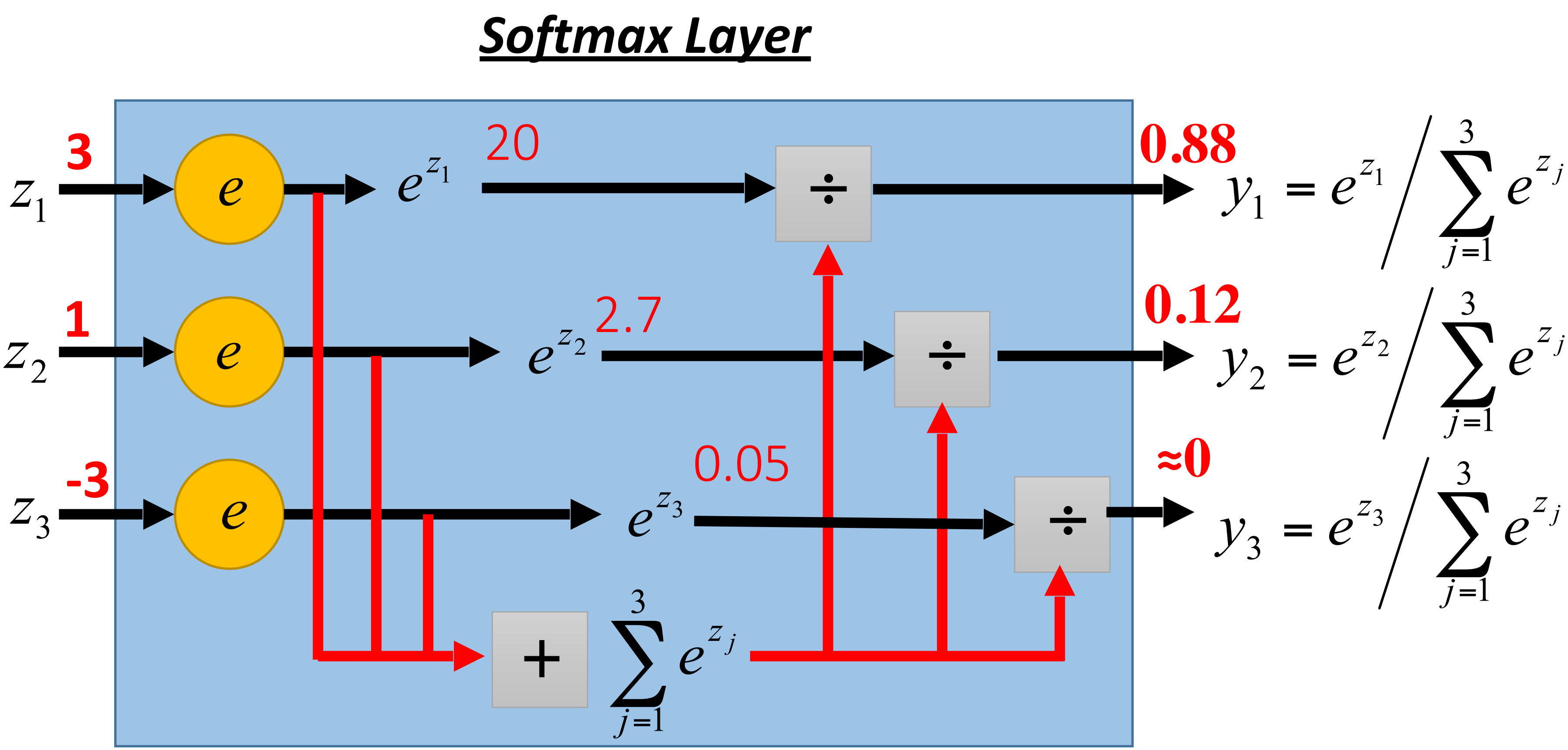
Probability:
- $1 > y_i > 0$
- $\sum_i y_i = 1$



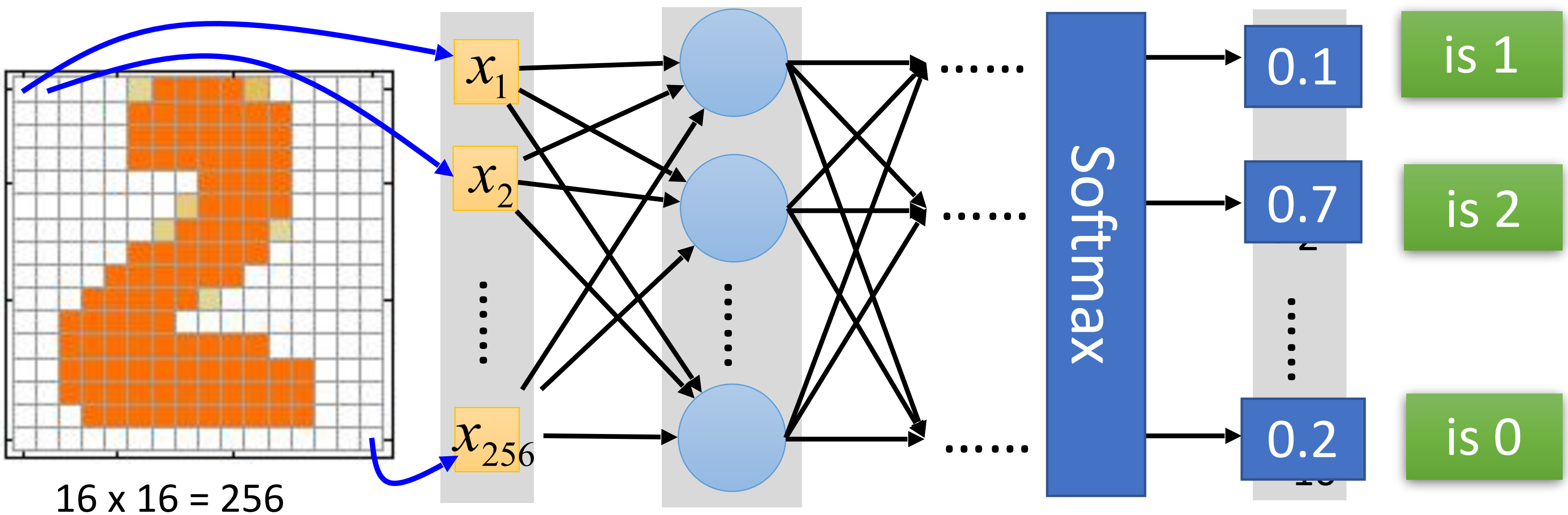
Softmax

- Softmax layer as the output layer

Probability:
- $1 > y_i > 0$
- $\sum_i y_i = 1$



How to set network parameters

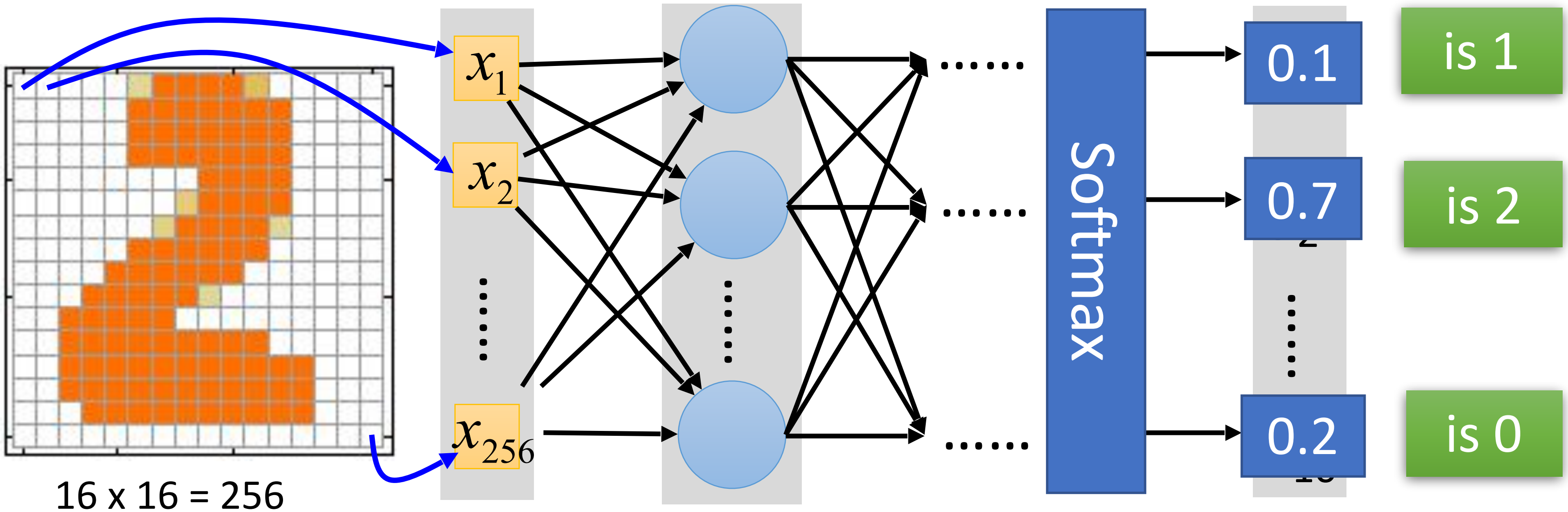


16 x 16 = 256

Ink → 1

No ink → 0

How to set network parameters



16 x 16 = 256

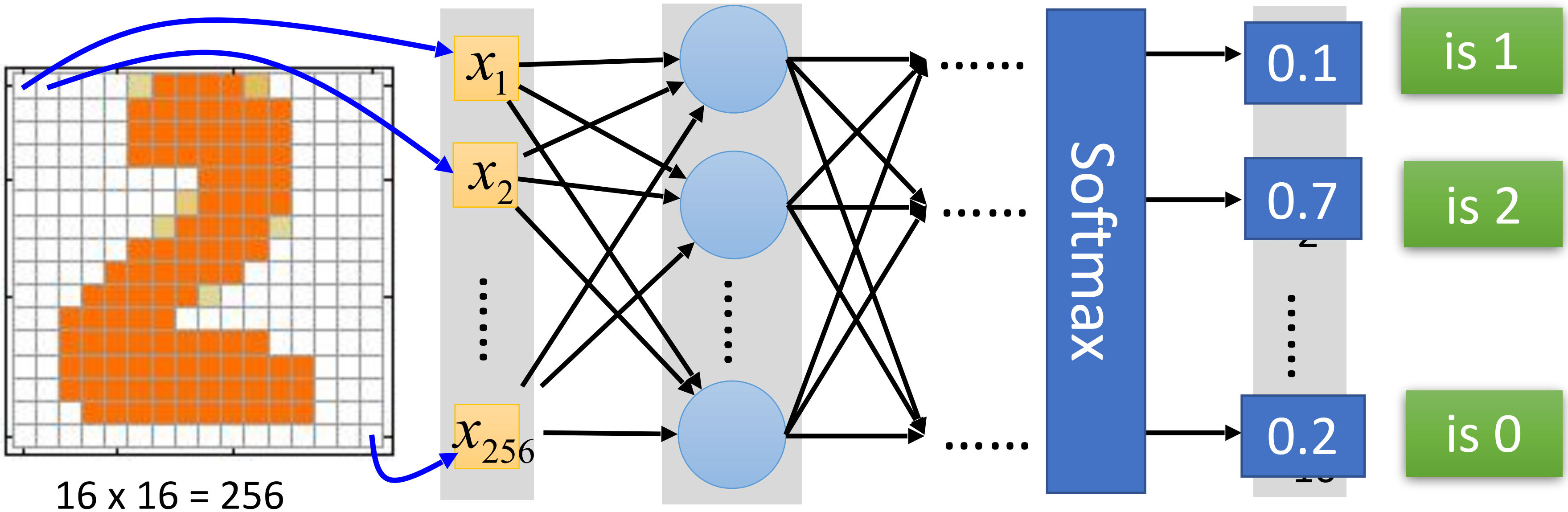
Ink → 1

No ink → 0

Set the network parameters θ such that ……

How to set network parameters

$$\theta = \{W^1, b^1, W^2, b^2, \cdots W^L, b^L\}$$



16 x 16 = 256

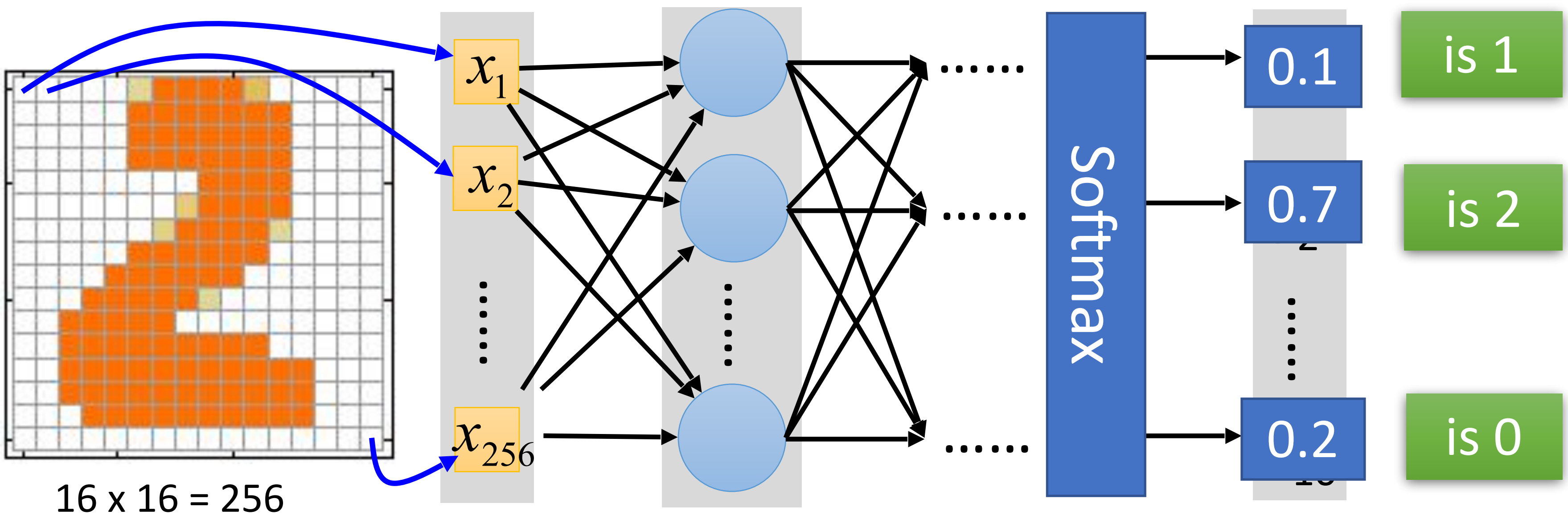
Ink → 1

No ink → 0

Set the network parameters θ such that ……

How to set network parameters

$$\theta = \{W^1, b^1, W^2, b^2, \cdots W^L, b^L\}$$

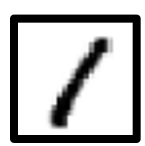


Ink → 1

No ink → 0

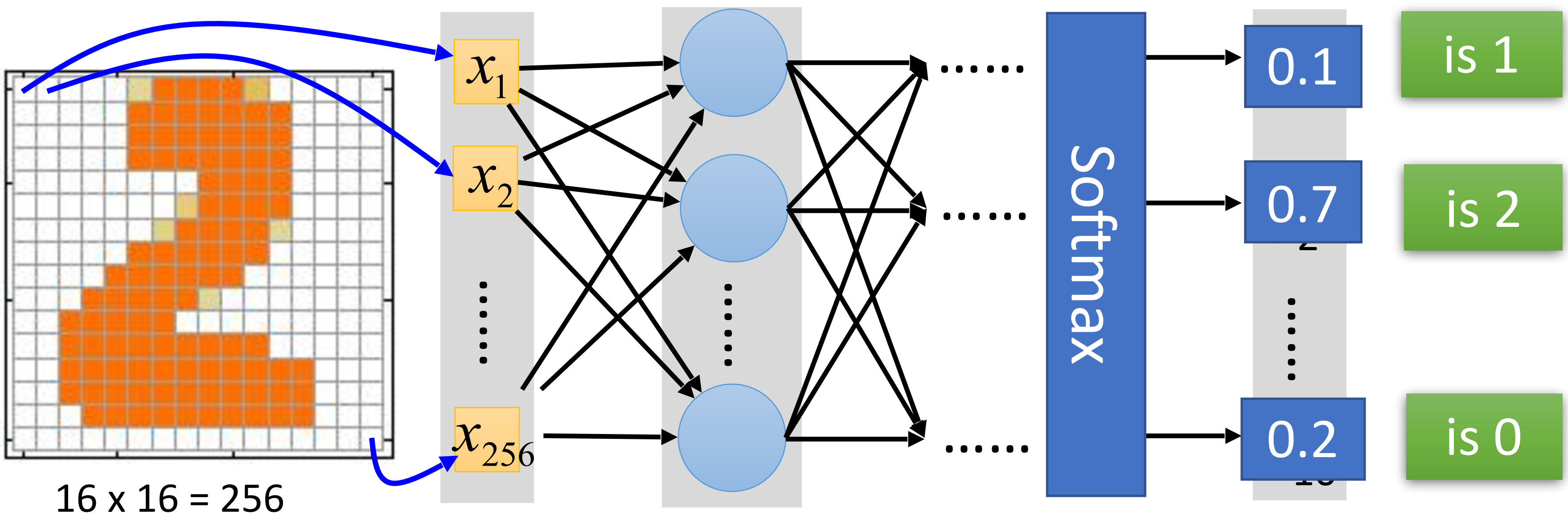
Set the network parameters θ such that ......

Input:



How to set network parameters

$$\theta = \{W^1, b^1, W^2, b^2, \cdots W^L, b^L\}$$

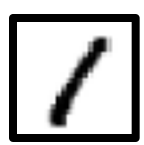


16 x 16 = 256

Ink → 1

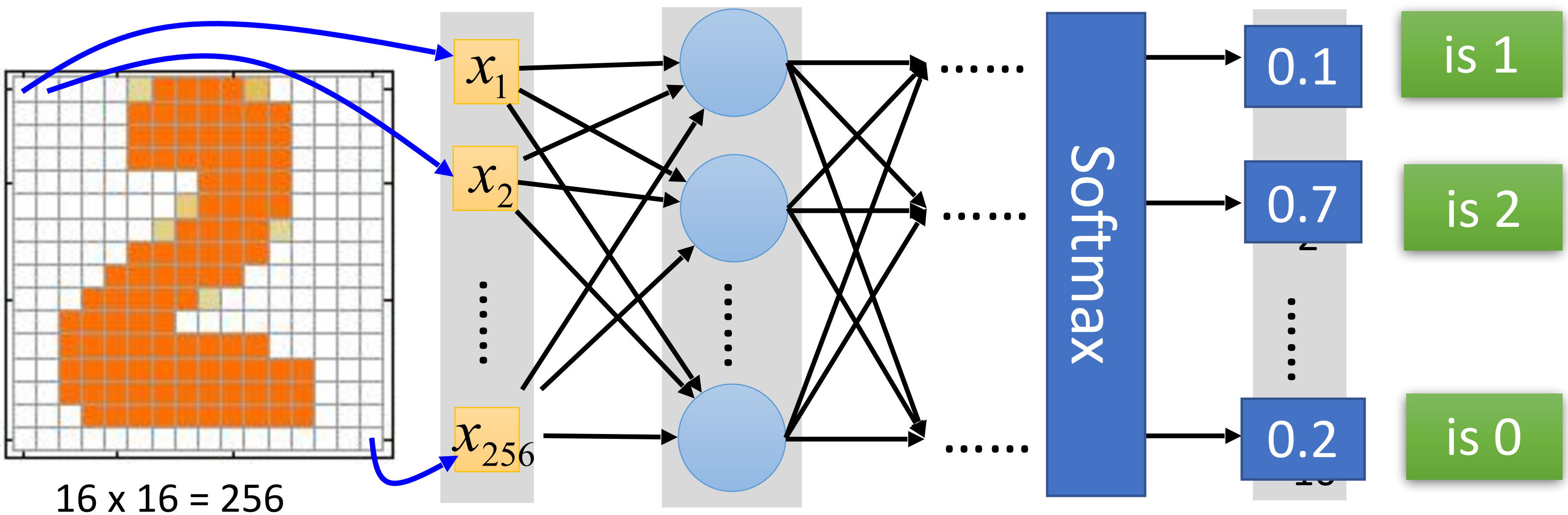
No ink → 0

Set the network parameters θ such that ......

Input:  y_1 has the maximum value

How to set network parameters

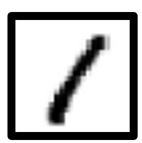
$$\theta = \{W^1, b^1, W^2, b^2, \cdots W^L, b^L\}$$



Ink → 1

No ink → 0

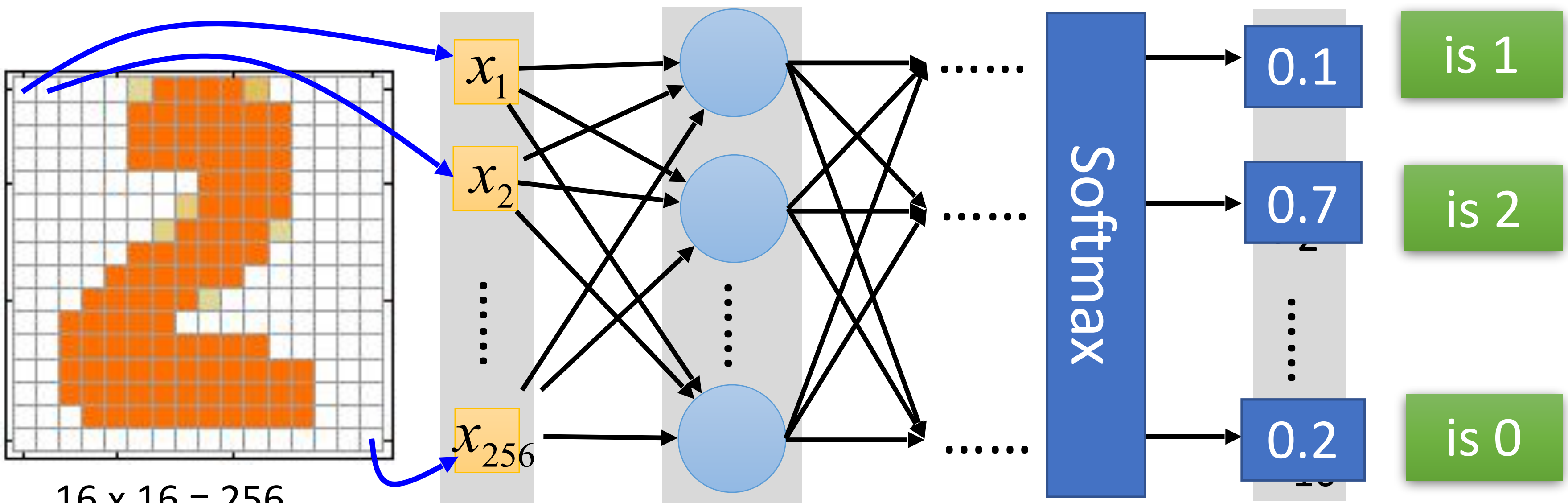
Set the network parameters θ such that ……

Input:  → y_1 has the maximum value

Input: 

How to set network parameters

$$\theta = \{W^1, b^1, W^2, b^2, \cdots W^L, b^L\}$$

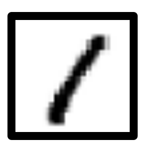


16 x 16 = 256

Ink → 1

No ink → 0

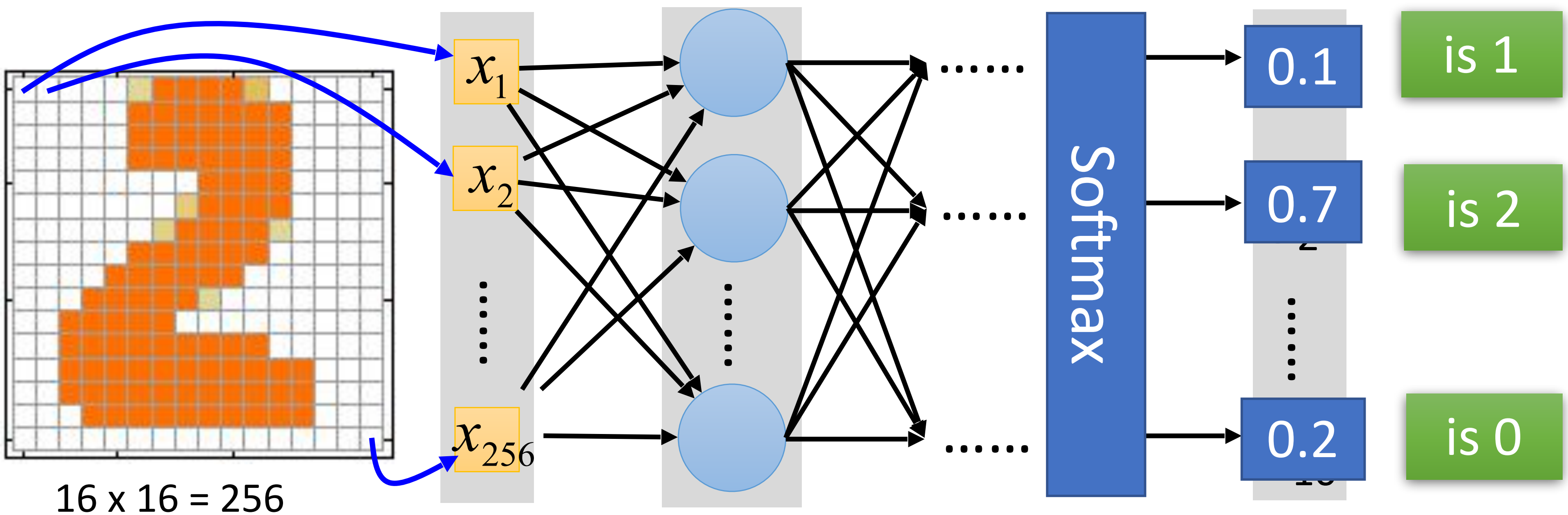
Set the network parameters θ such that ......

Input:  → y_1 has the maximum value

Input:  → y_2 has the maximum value

How to set network parameters

$$\theta = \{W^1, b^1, W^2, b^2, \cdots W^L, b^L\}$$



Ink → 1

No ink → 0

Set the network parameters θ such that ……

Input: … n value

How to let the neural network achieve this

Input:  y_2 has the maximum value

Training Data

- Preparing training data: images and their labels

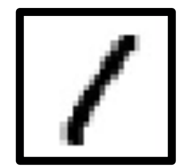
Training Data

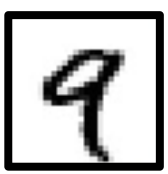
- Preparing training data: images and their labels


“5”

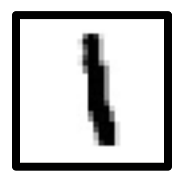

“0”

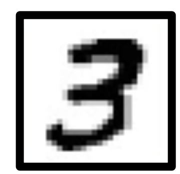

“4”


“1”


“9”

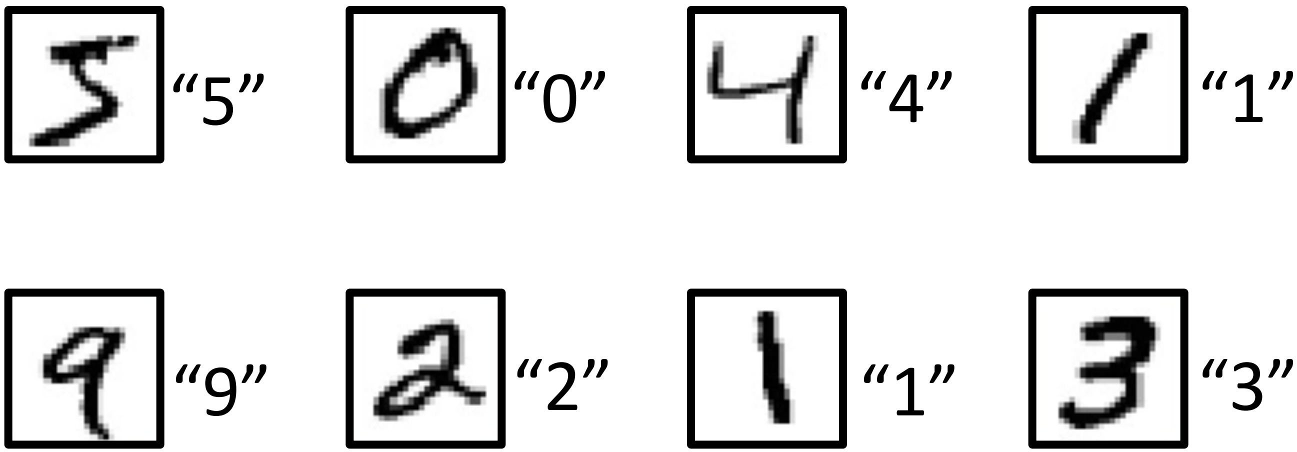

“2”


“1”


“3”

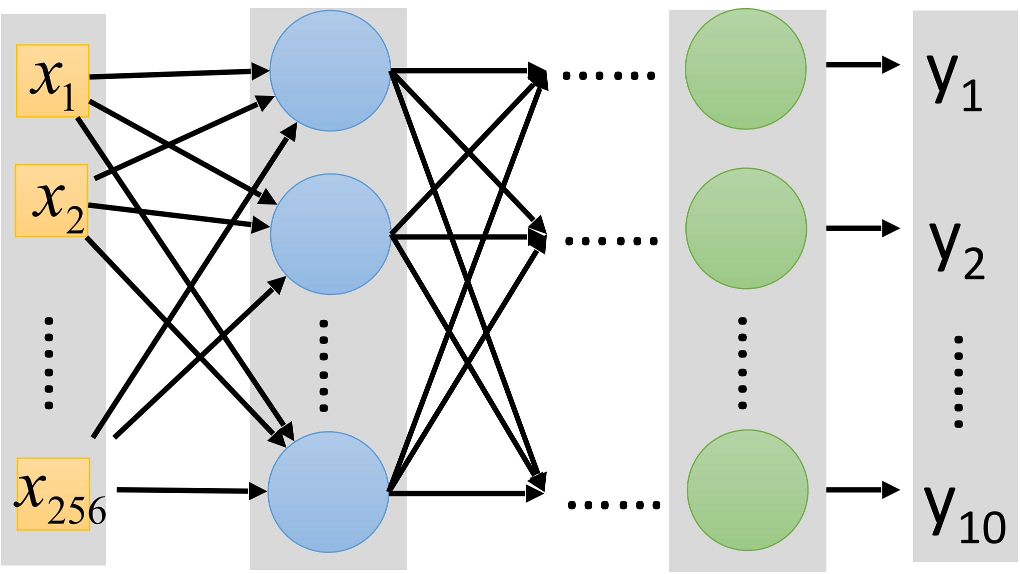
Training Data

- Preparing training data: images and their labels



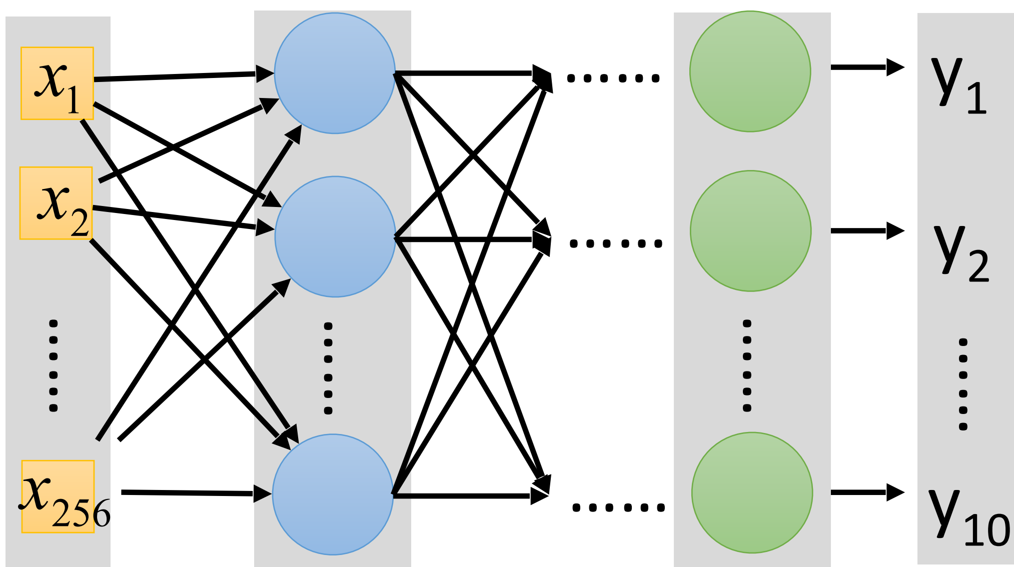
Using the training data to find the network parameters.

Cost



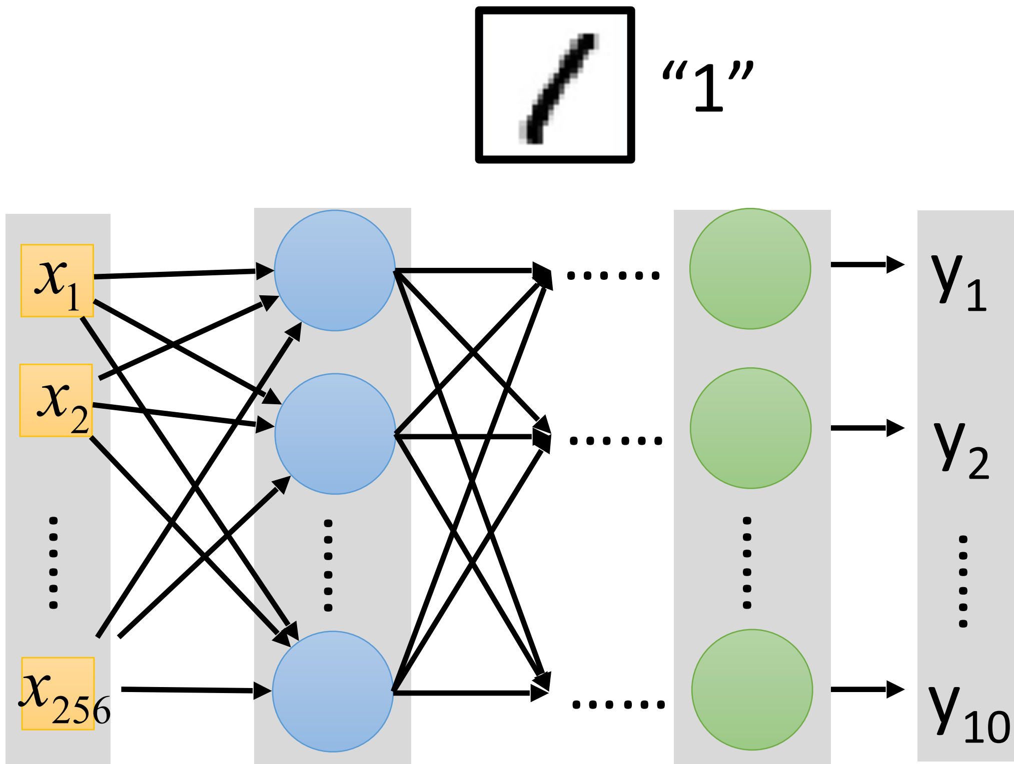
Cost

Given a set of network parameters θ, each example has a cost value.



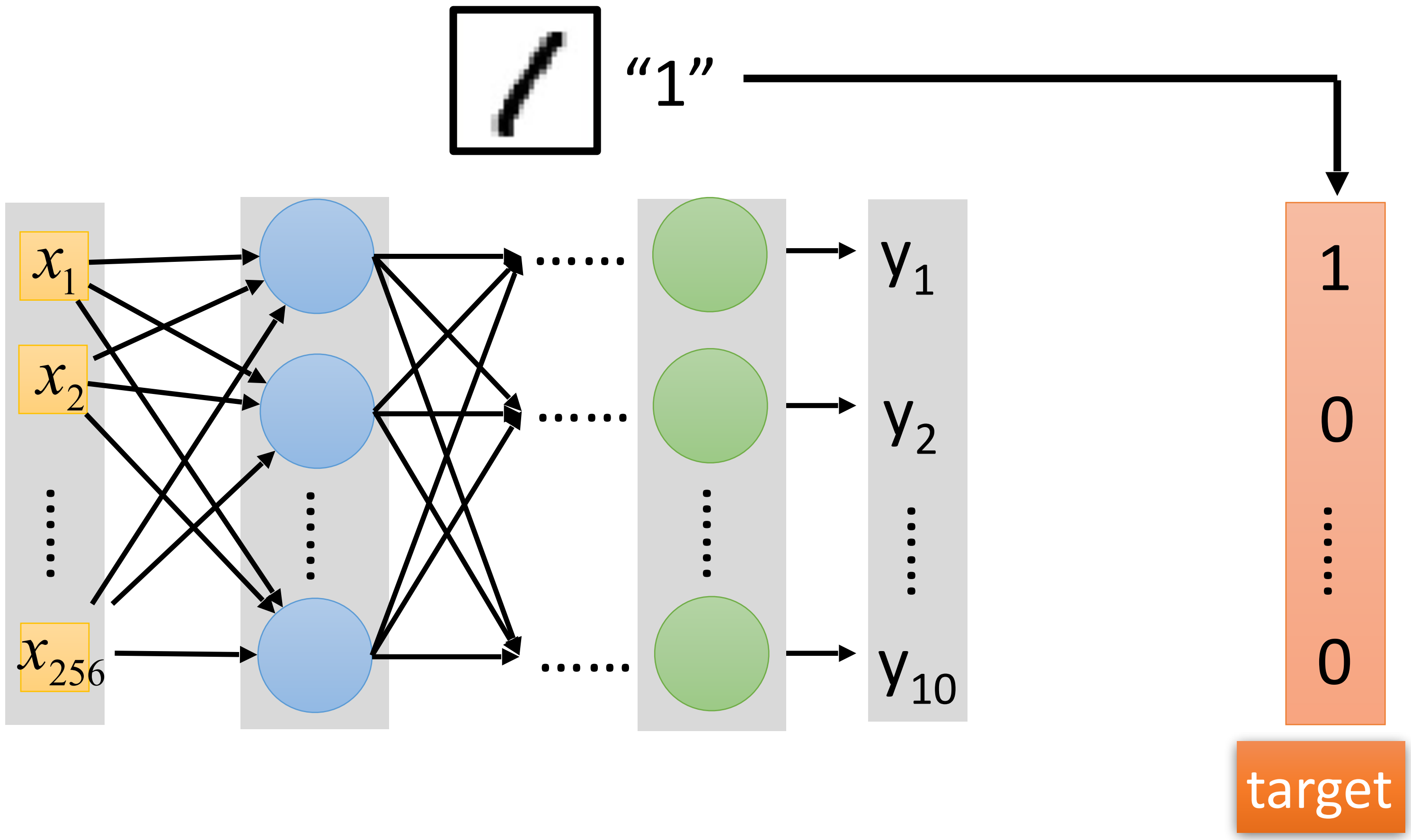
Cost

Given a set of network parameters θ, each example has a cost value.



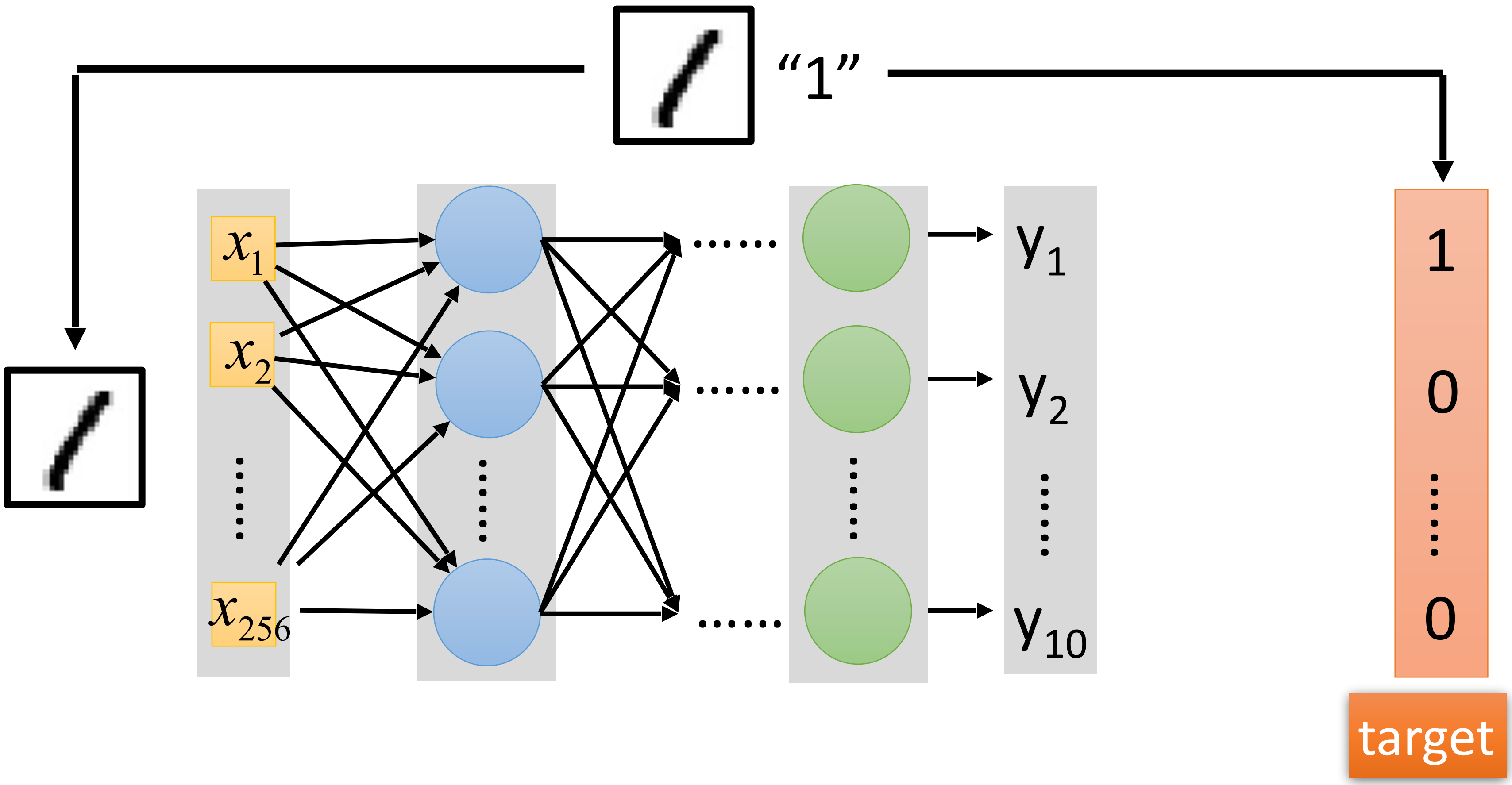
Cost

Given a set of network parameters θ, each example has a cost value.



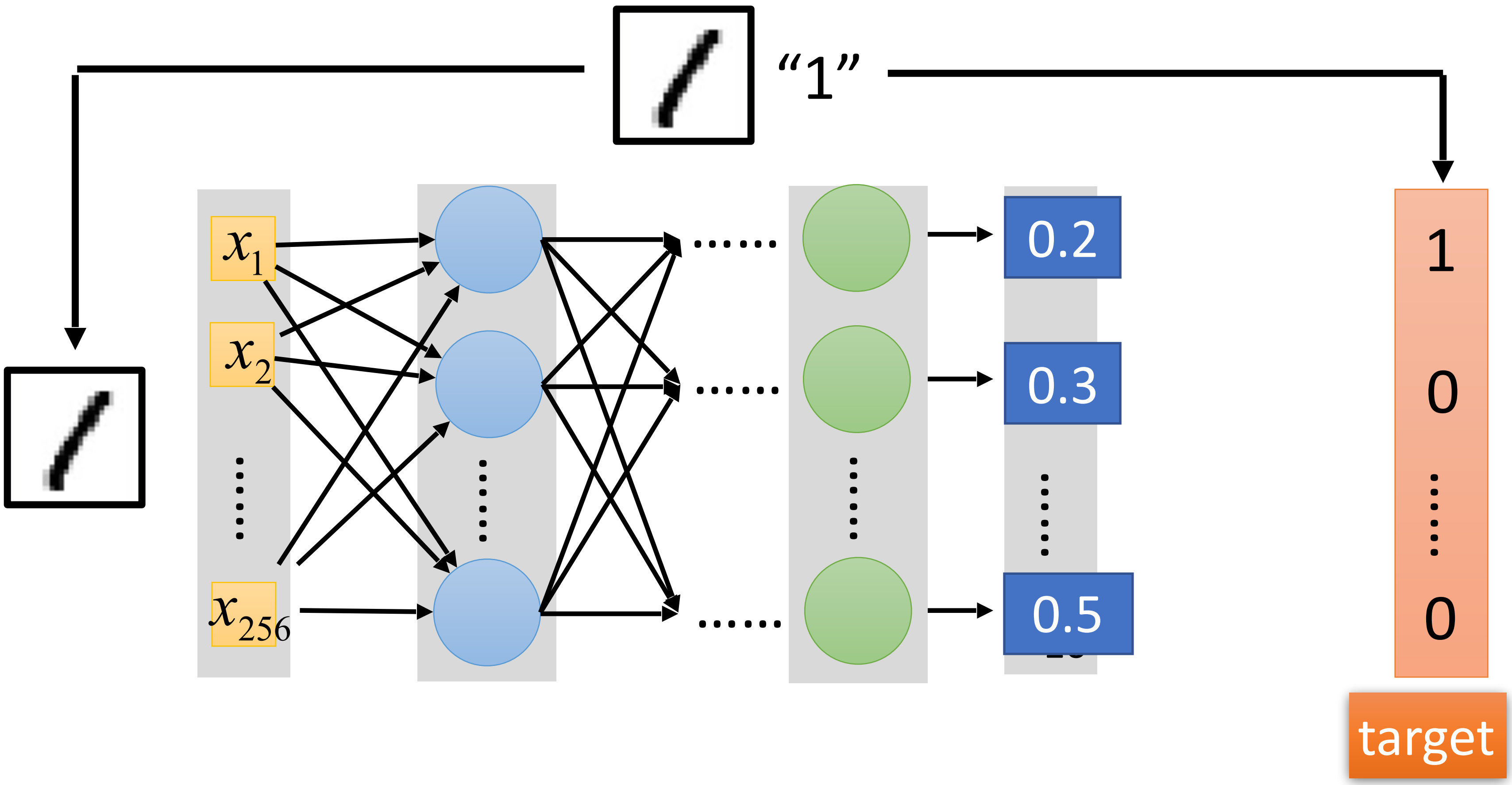
Cost

Given a set of network parameters θ, each example has a cost value.



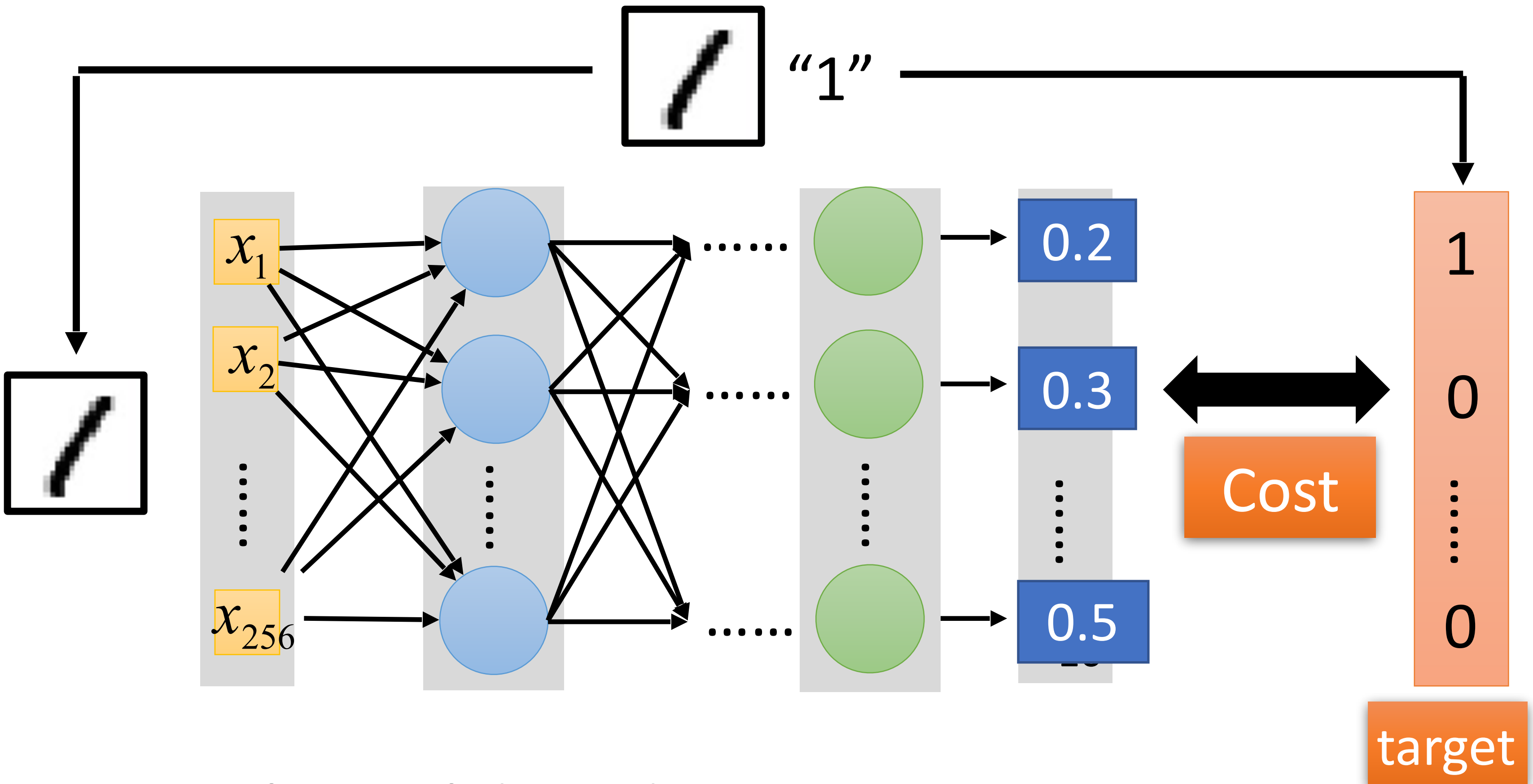
Cost

Given a set of network parameters θ, each example has a cost value.



Cost

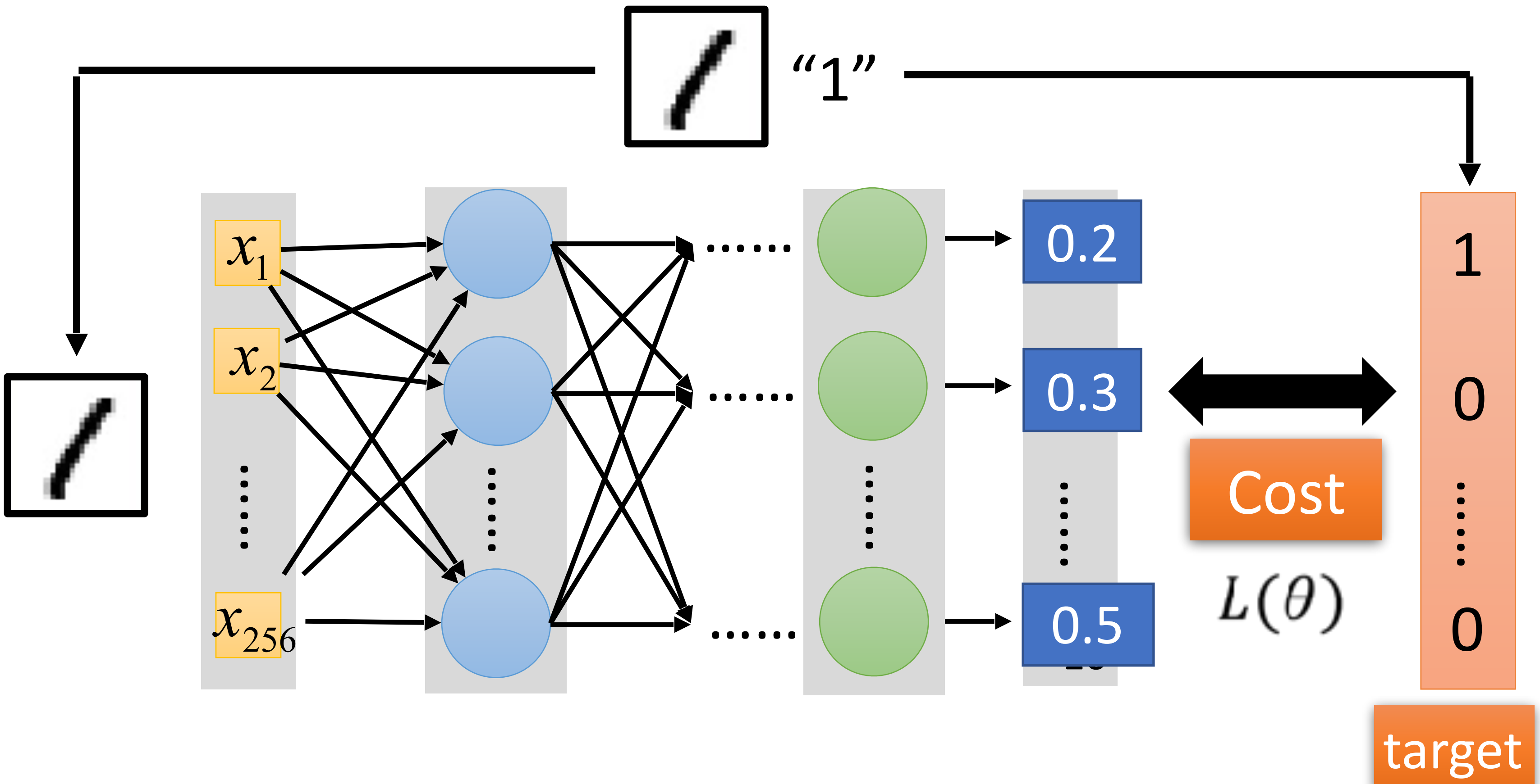
Given a set of network parameters θ, each example has a cost value.



Cost can be Euclidean distance or cross entropy of the network output and target

Cost

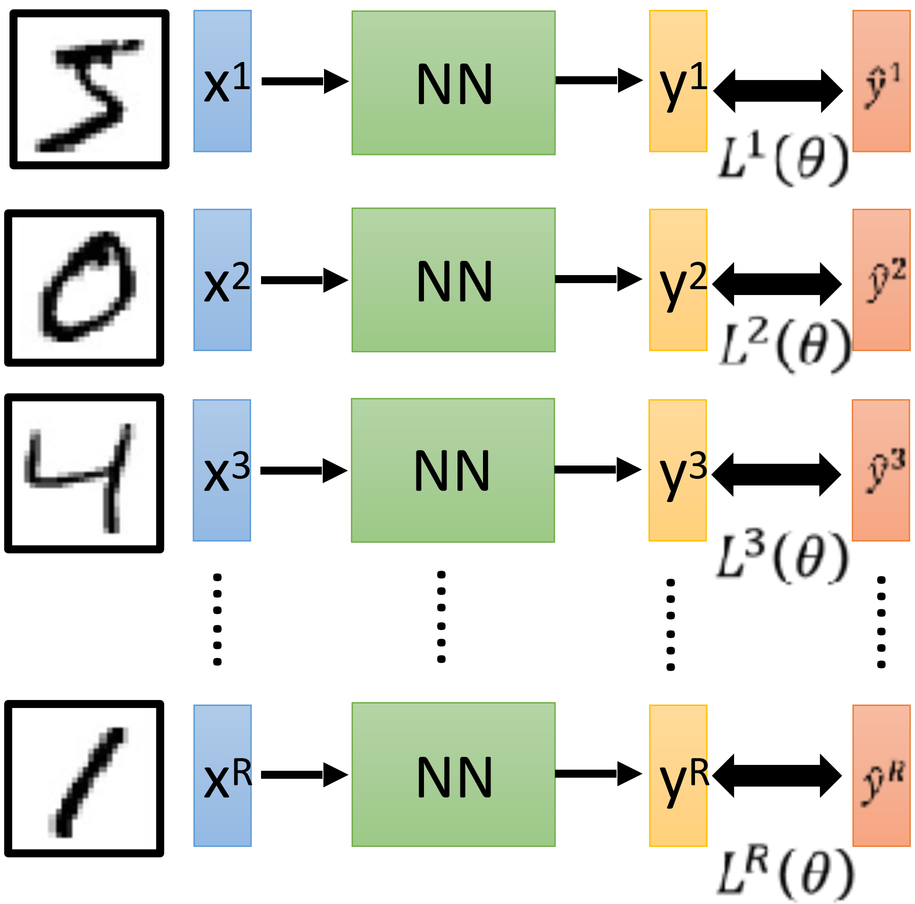
Given a set of network parameters θ, each example has a cost value.



Cost can be Euclidean distance or cross entropy of the network output and target

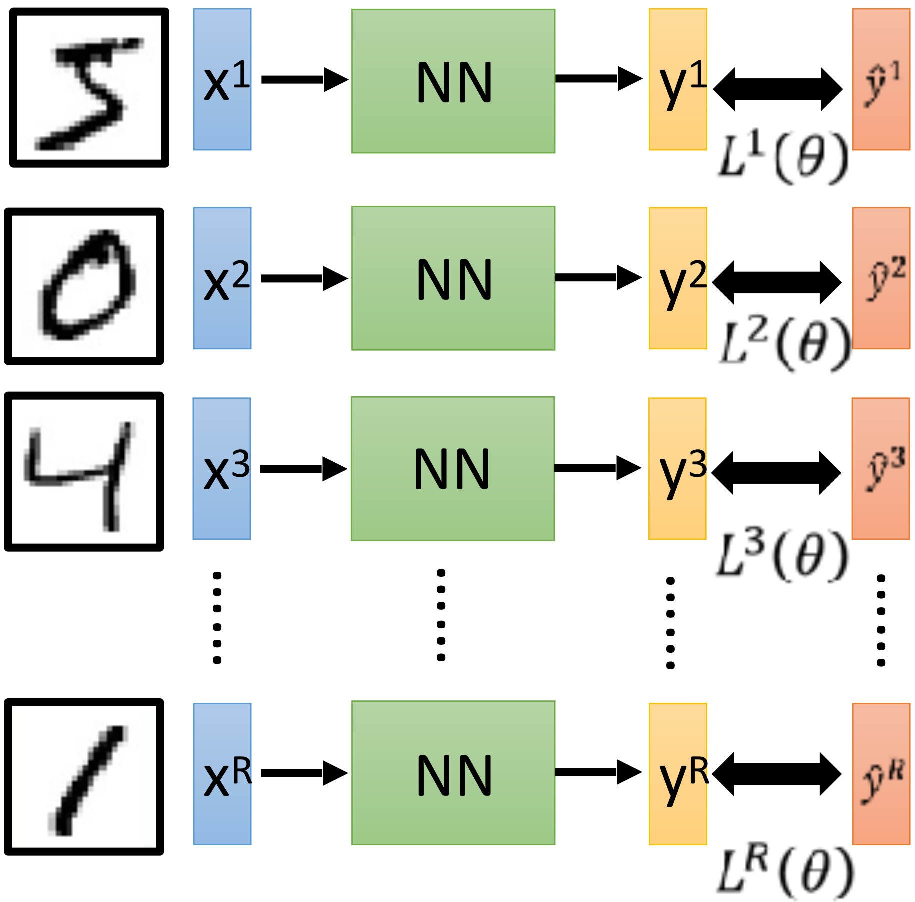
Total Cost

For all training data …



Total Cost

For all training data …

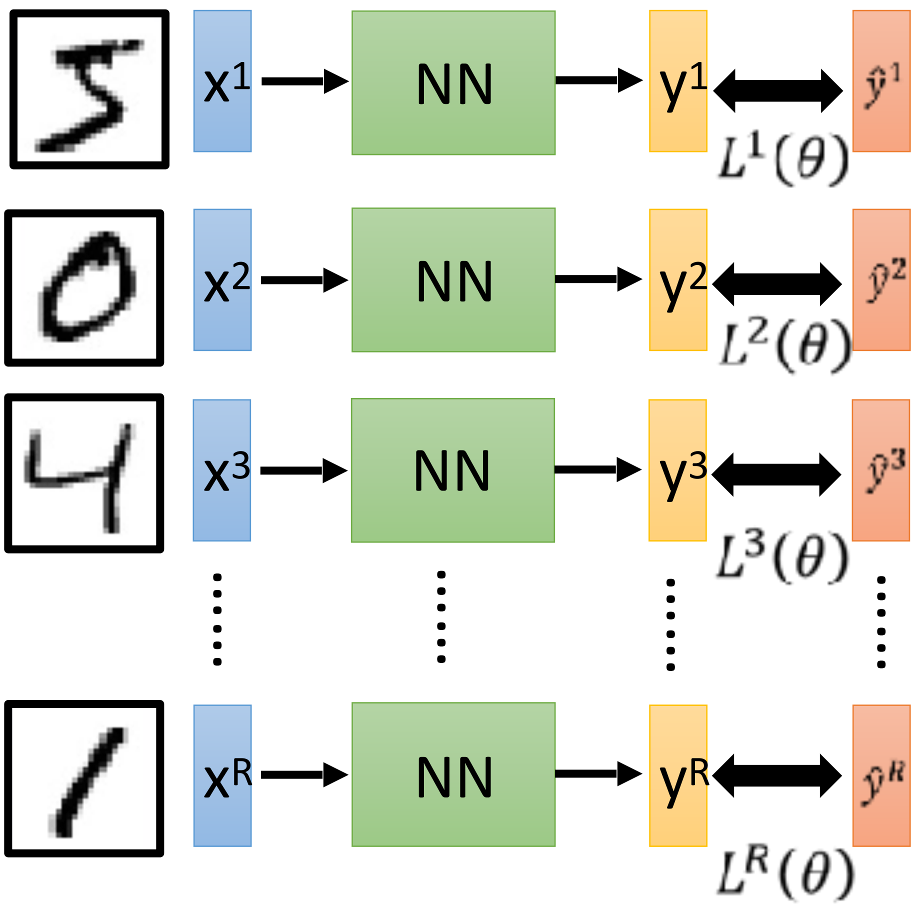


Total Cost:

$$C(\theta) = \sum_{r=1}^{R} L^r(\theta)$$

Total Cost

For all training data …



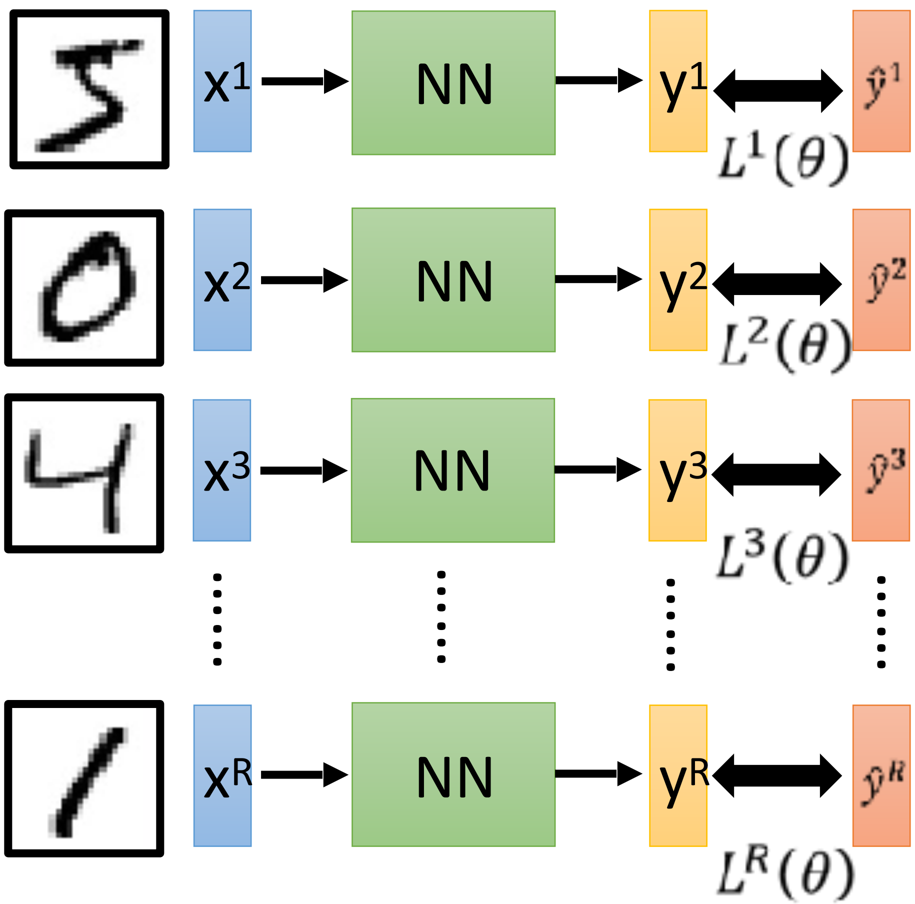
Total Cost:

$$C(\theta) = \sum_{r=1}^{R} L^r(\theta)$$

How bad the network parameters θ is on this task

Total Cost

For all training data …



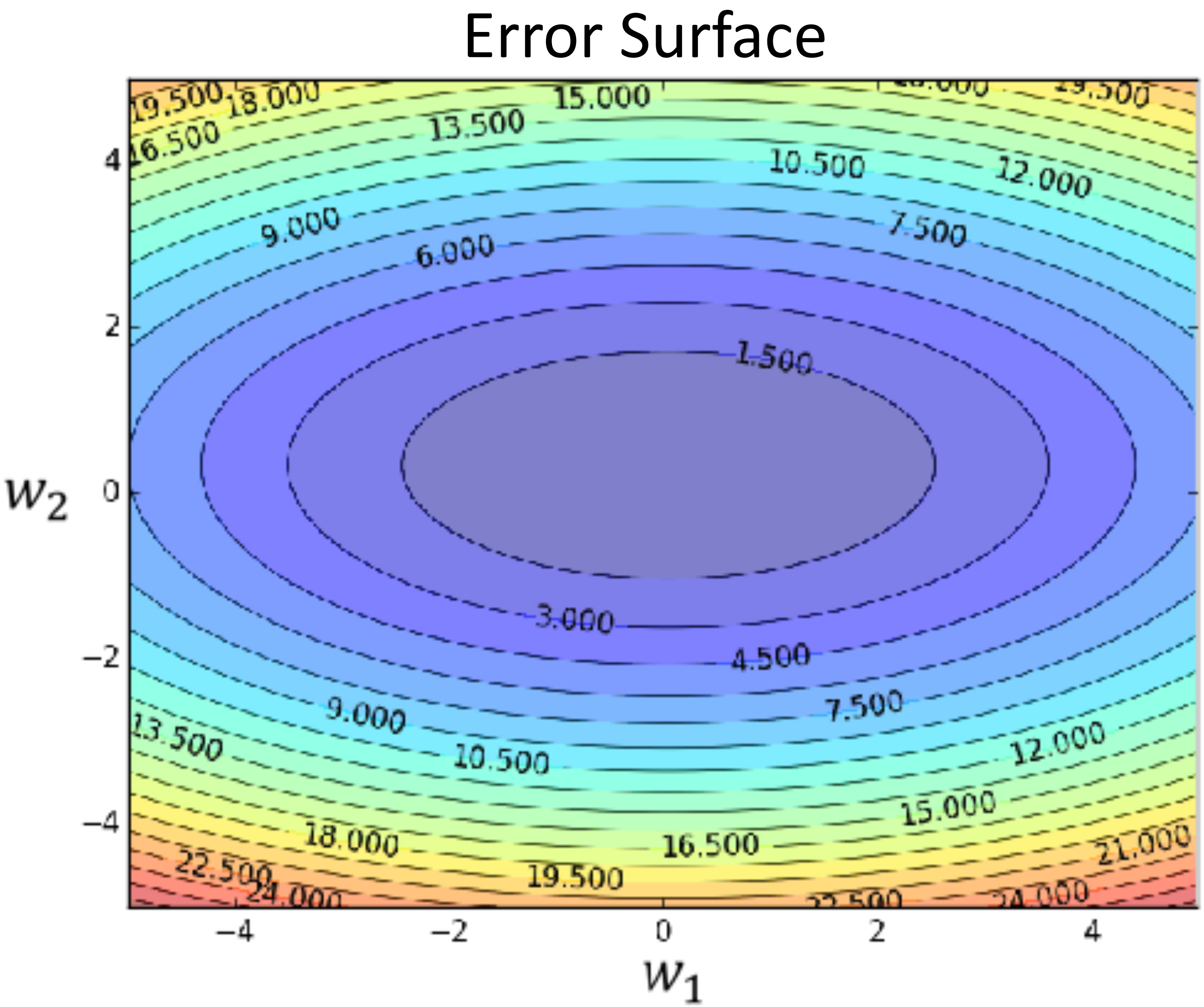
Total Cost:

$$C(\theta) = \sum_{r=1}^{R} L^r(\theta)$$

How bad the network parameters θ is on this task

Find the network parameters θ^* that minimize this value

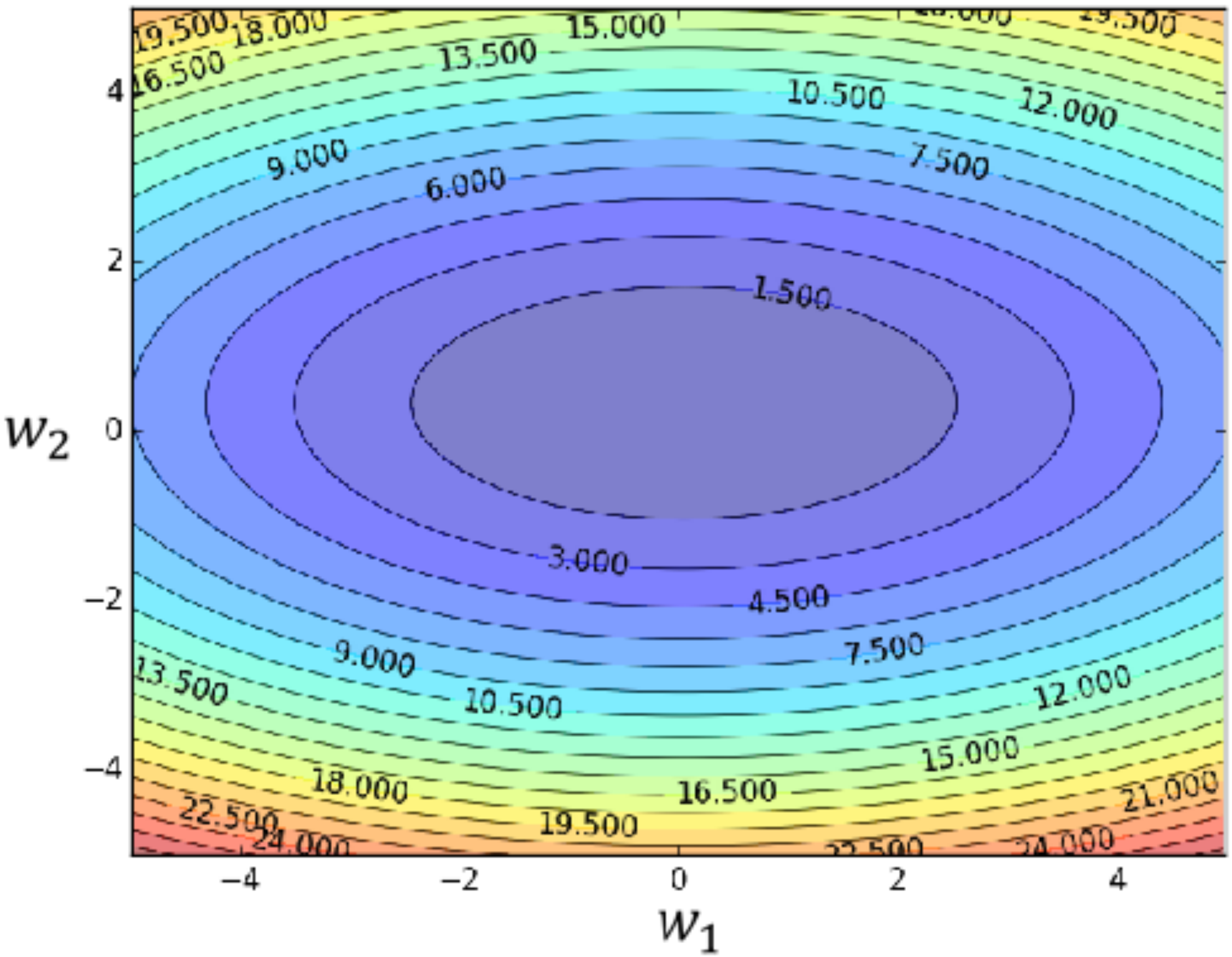
Gradient Descent



Gradient Descent

Assume there are only two parameters w_1 and w_2 in a network.

Error Surface

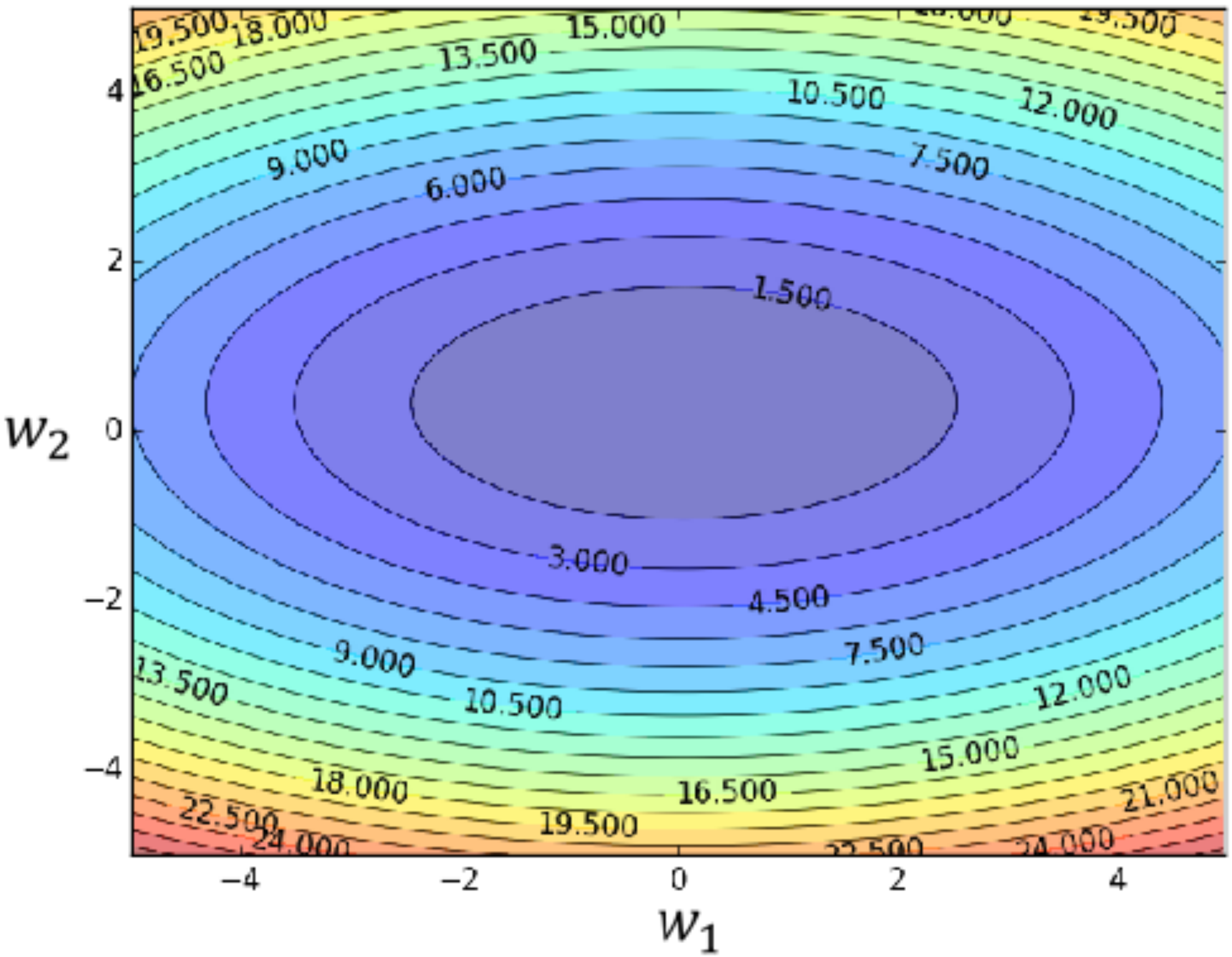


Gradient Descent

Assume there are only two parameters w_1 and w_2 in a network.

$$\theta = \{w_1, w_2\}$$

Error Surface

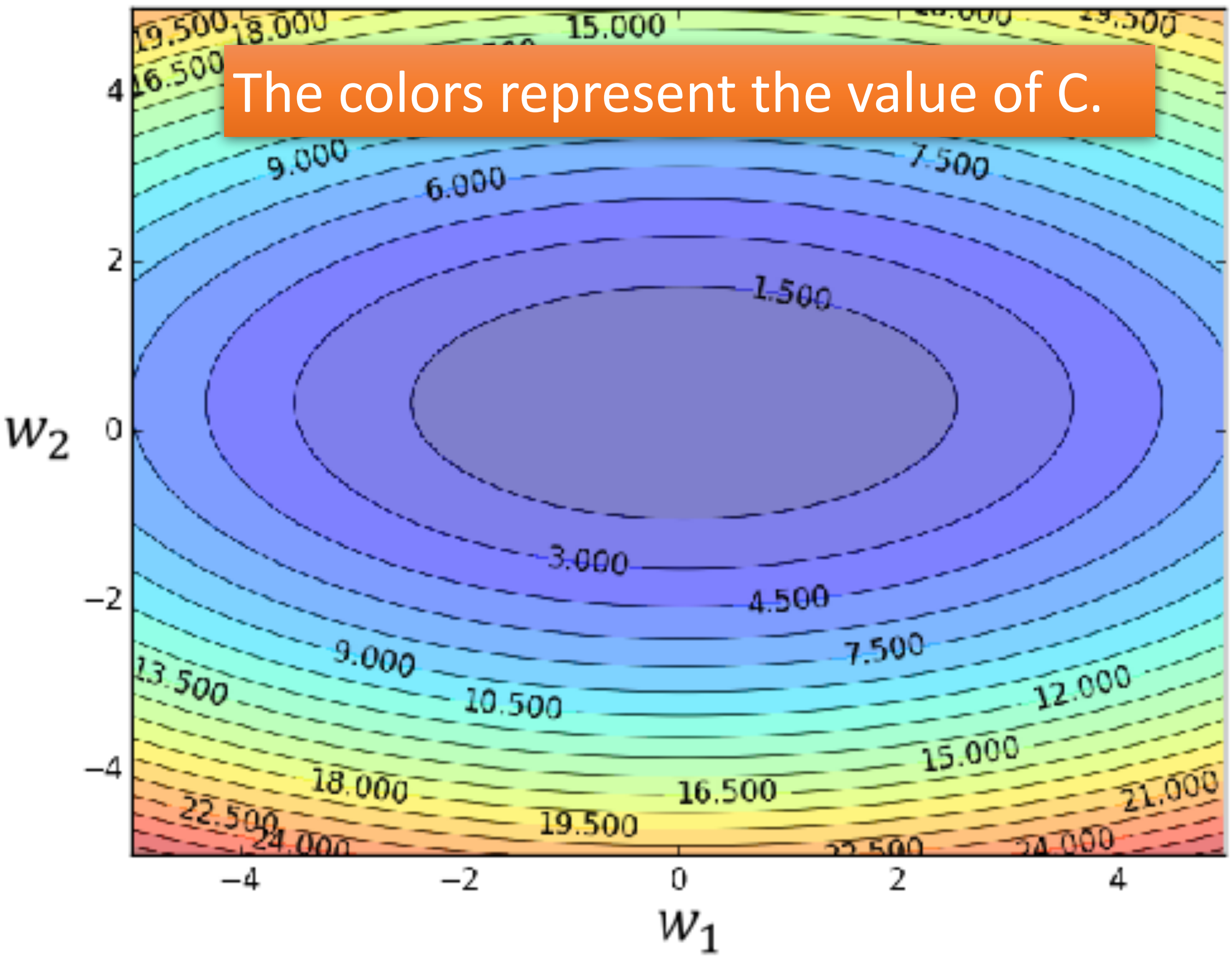


Gradient Descent

Assume there are only two parameters w_1 and w_2 in a network.

$$\theta = \{w_1, w_2\}$$

Error Surface

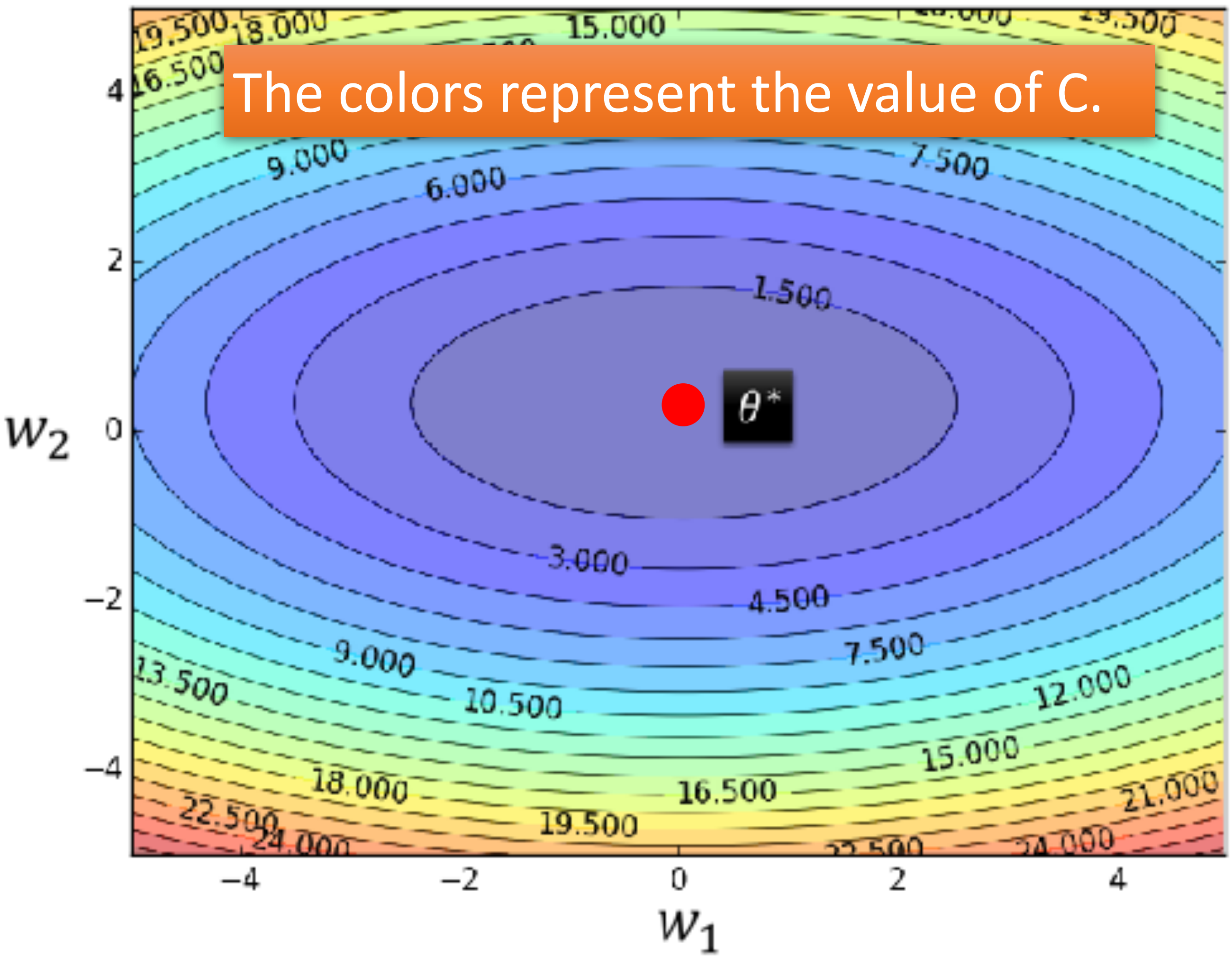


Gradient Descent

Assume there are only two parameters w_1 and w_2 in a network.

$$\theta = \{w_1, w_2\}$$

Error Surface

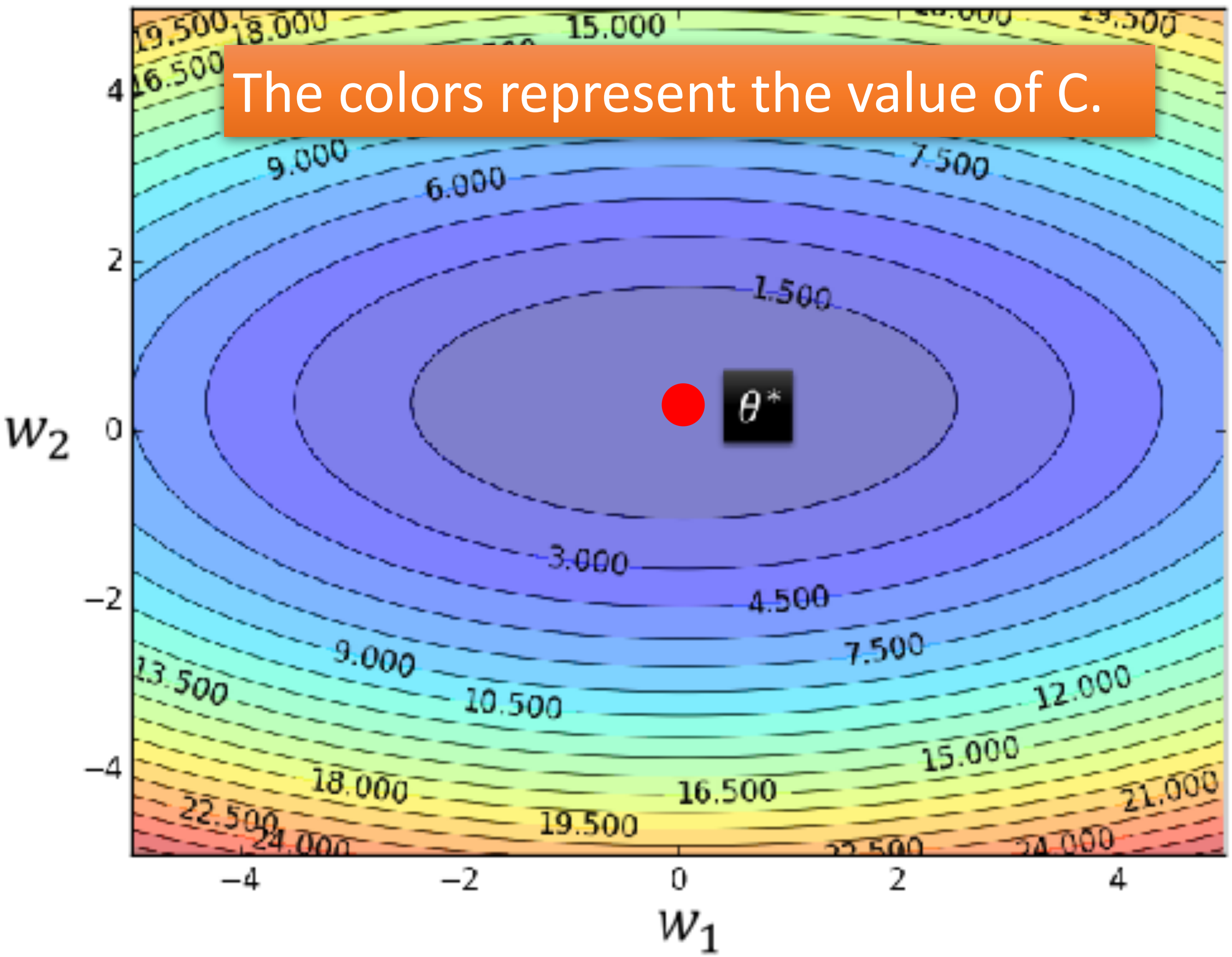


Gradient Descent

Assume there are only two parameters w_1 and w_2 in a network.

$$\theta = \{w_1, w_2\}$$

Error Surface



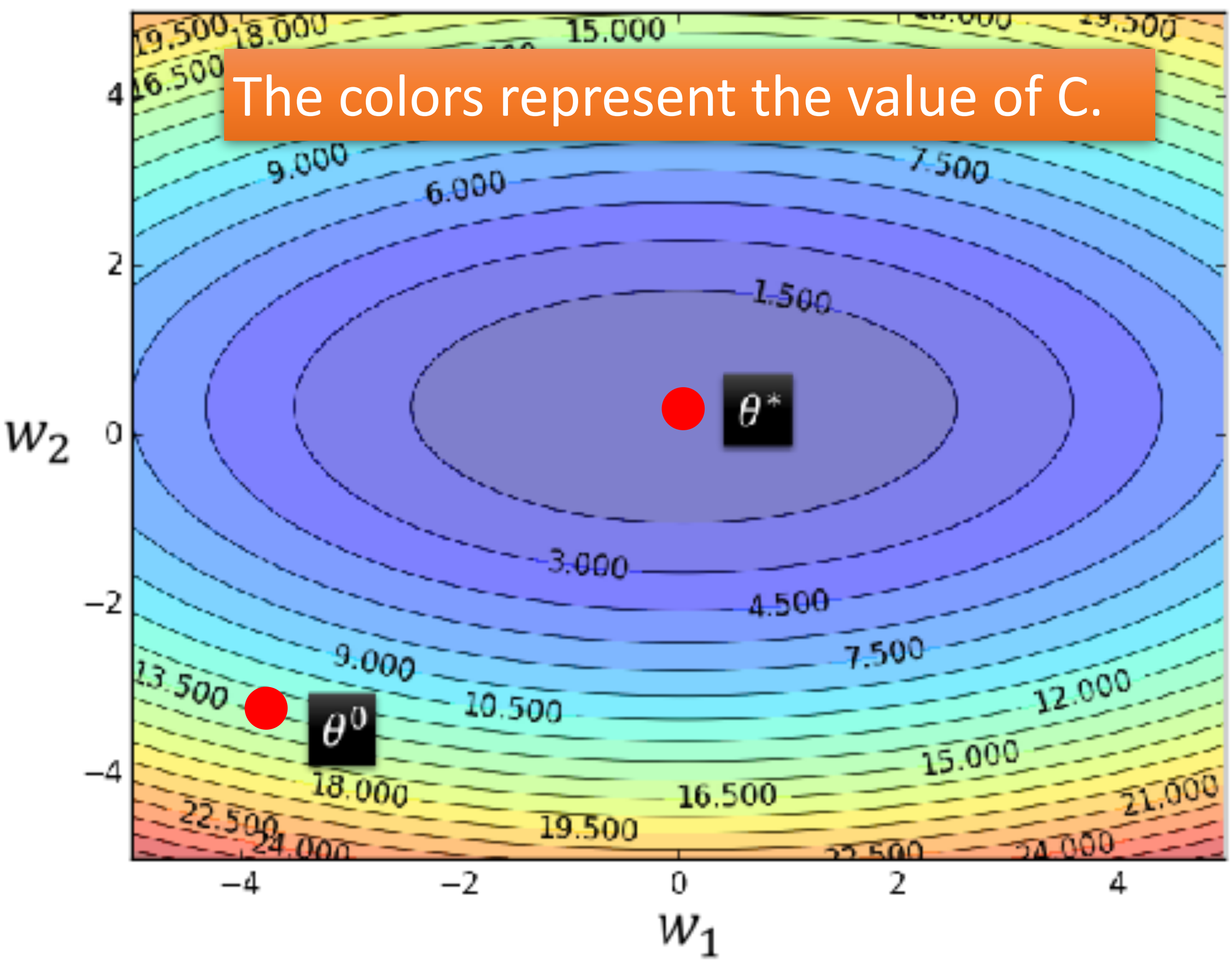
Randomly pick a starting point θ^0

Gradient Descent

Assume there are only two parameters w_1 and w_2 in a network.

$$\theta = \{w_1, w_2\}$$

Error Surface



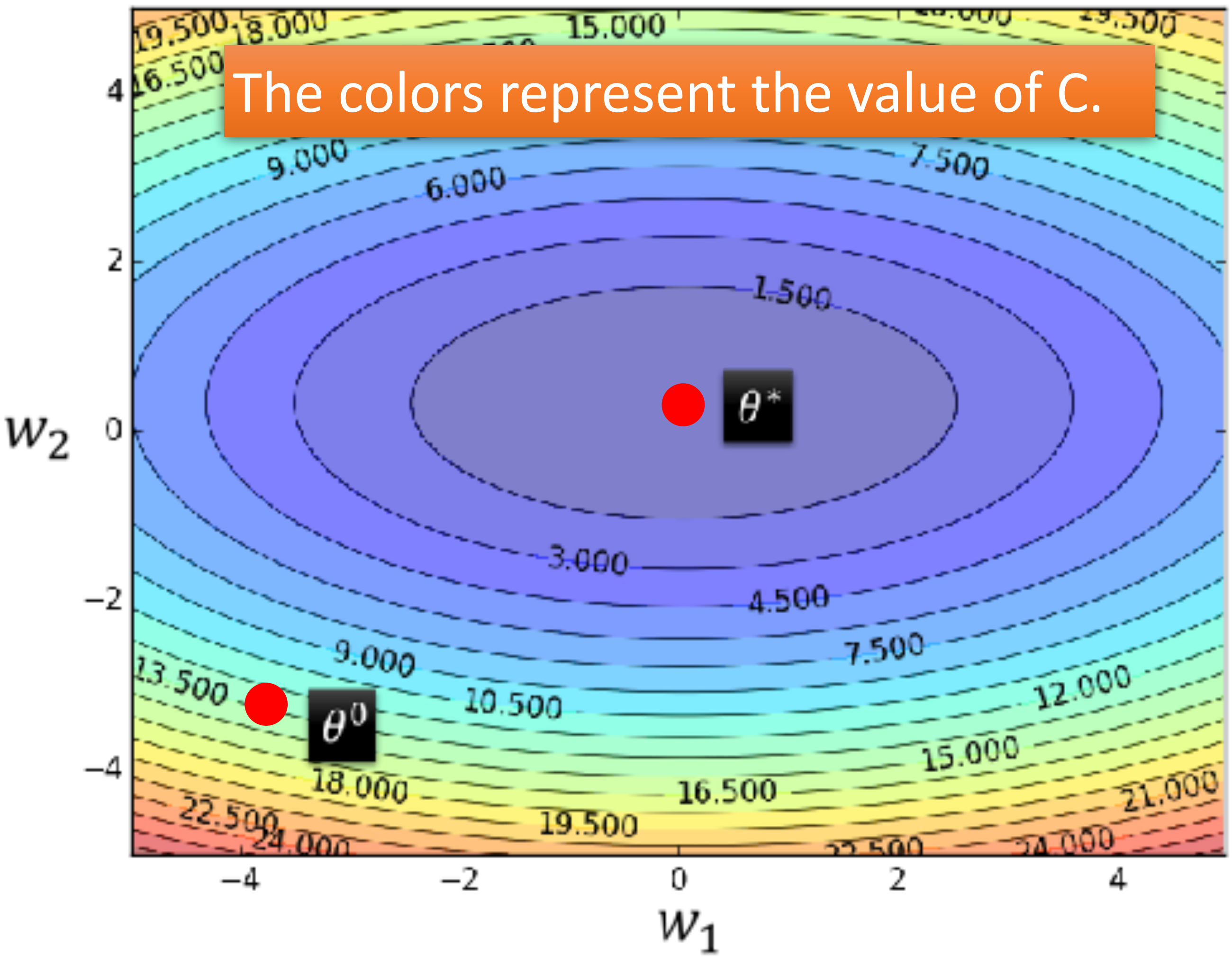
Randomly pick a starting point θ^0

Gradient Descent

Assume there are only two parameters w_1 and w_2 in a network.

$$\theta = \{w_1, w_2\}$$

Error Surface



Randomly pick a starting point θ^0

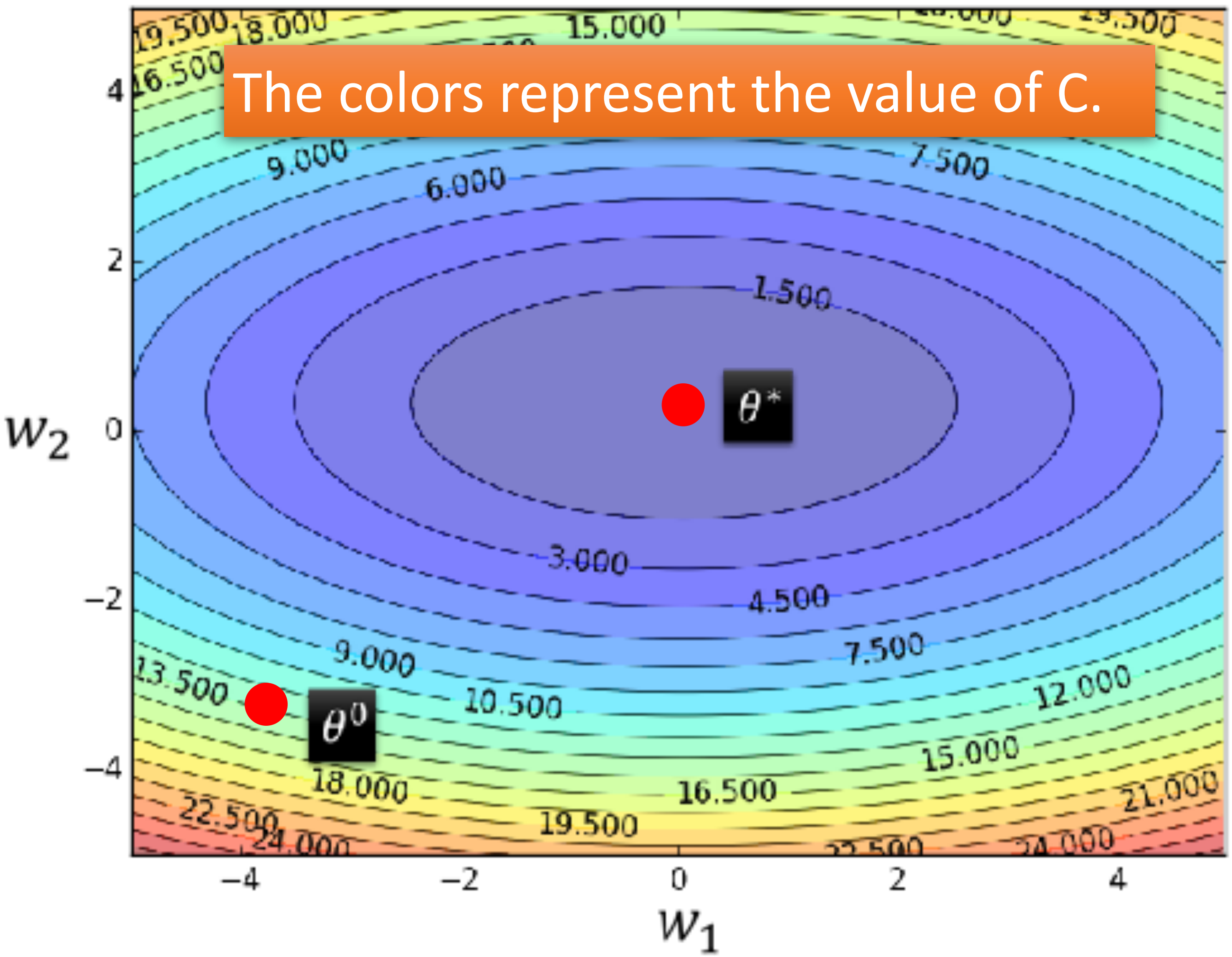
Compute the negative gradient at θ^0

Gradient Descent

Assume there are only two parameters w_1 and w_2 in a network.

$$\theta = \{w_1, w_2\}$$

Error Surface



Randomly pick a starting point θ^0

Compute the negative gradient at θ^0

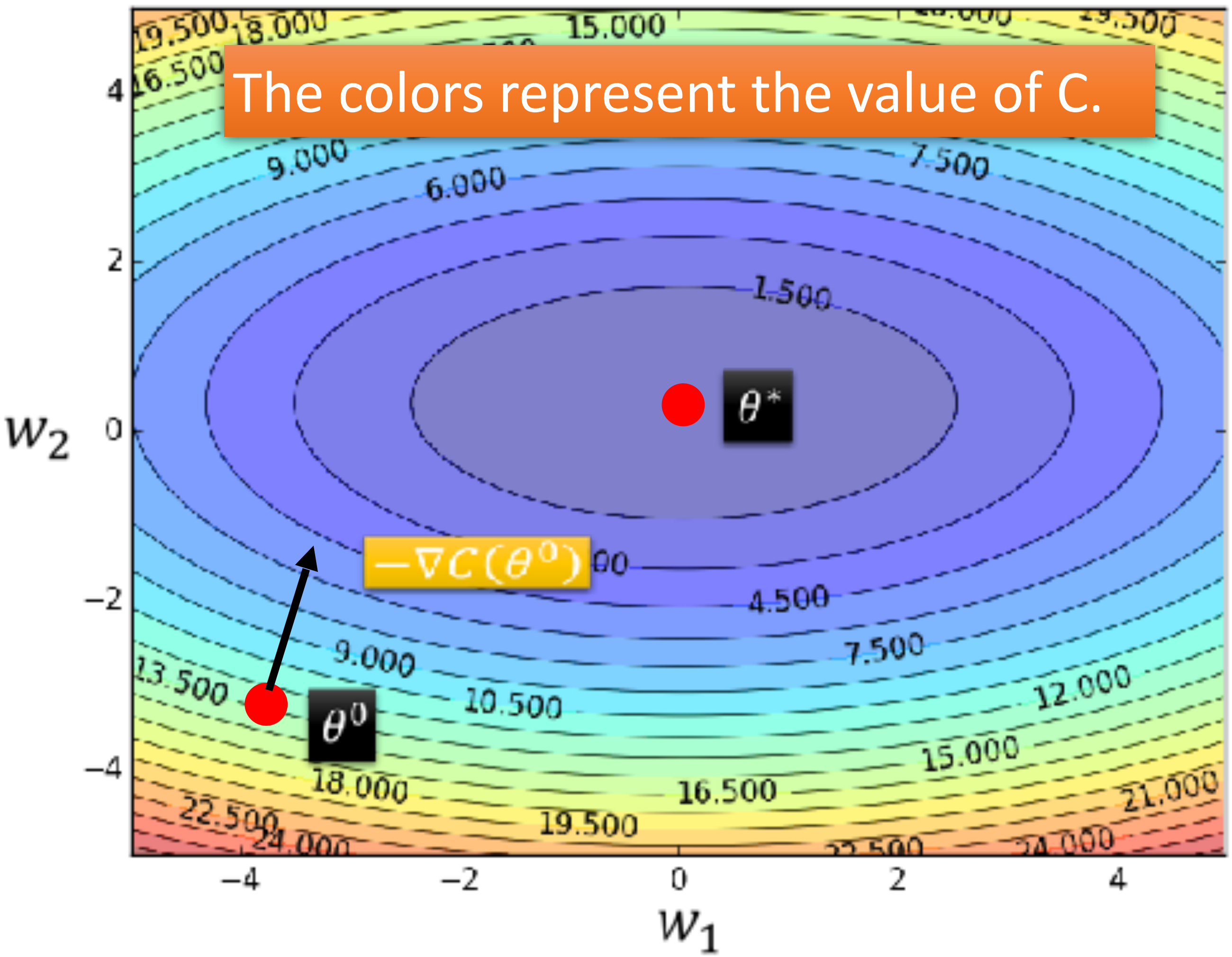
$$\rightarrow -\nabla C(\theta^0)$$

Gradient Descent

Assume there are only two parameters w_1 and w_2 in a network.

$$\theta = \{w_1, w_2\}$$

Error Surface



Randomly pick a starting point θ^0

Compute the negative gradient at θ^0

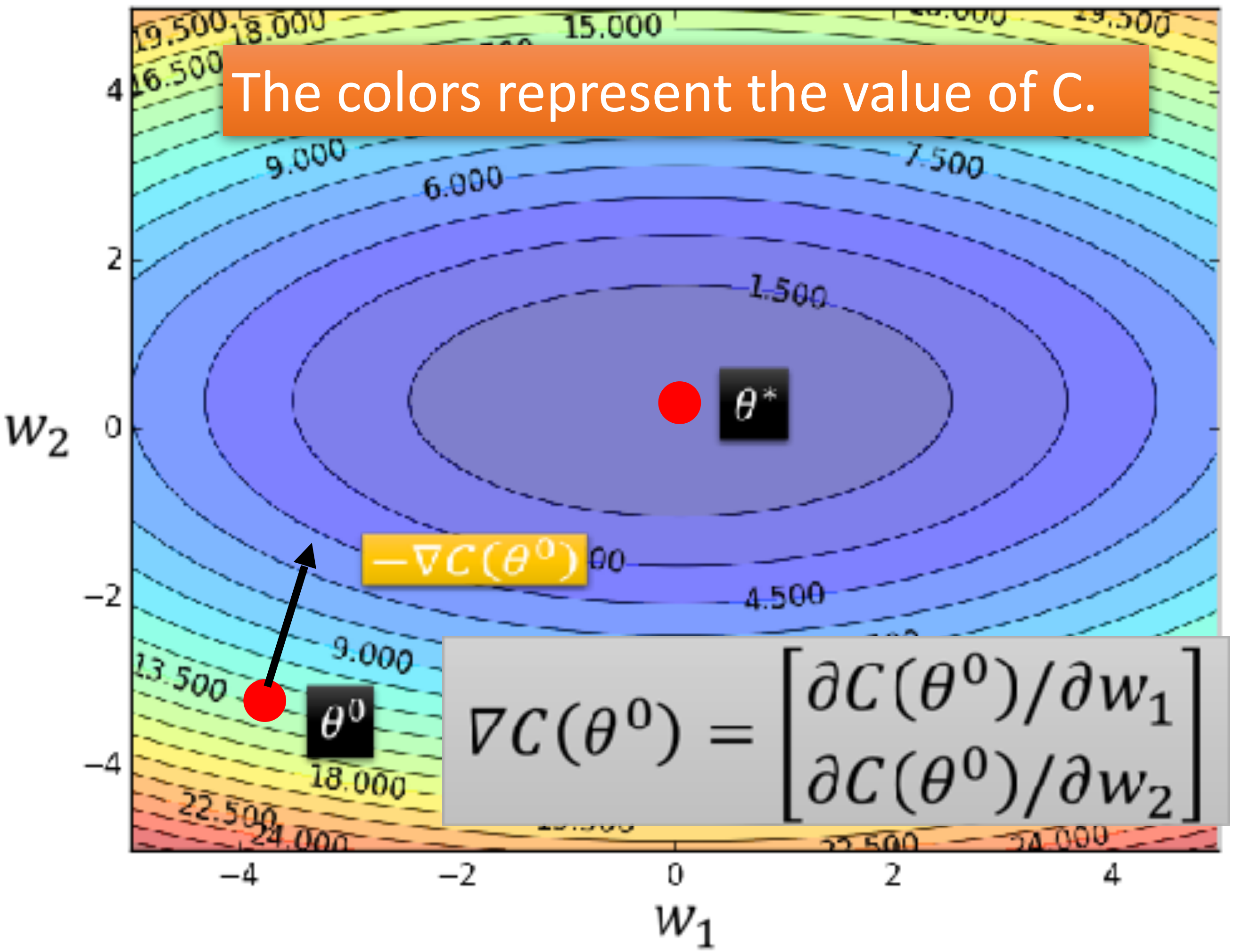
$\rightarrow -\nabla C(\theta^0)$

Gradient Descent

Assume there are only two parameters w_1 and w_2 in a network.

$$\theta = \{w_1, w_2\}$$

Error Surface



$$\nabla C(\theta^0) = \begin{bmatrix} \partial C(\theta^0)/\partial w_1 \\ \partial C(\theta^0)/\partial w_2 \end{bmatrix}$$

Randomly pick a starting point θ^0

Compute the negative gradient at θ^0

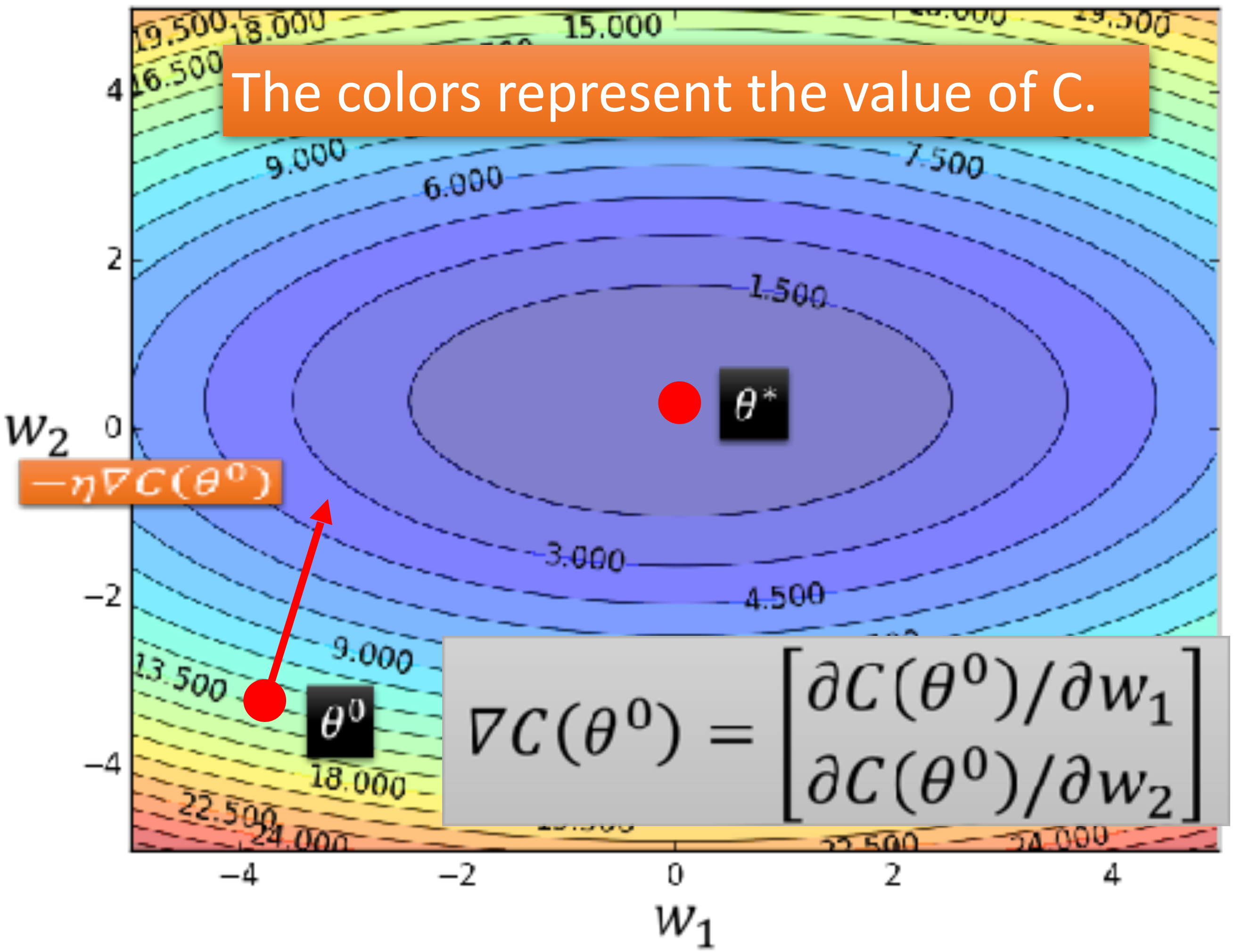
$\Rightarrow -\nabla C(\theta^0)$

Gradient Descent

Assume there are only two parameters w_1 and w_2 in a network.

$$\theta = \{w_1, w_2\}$$

Error Surface



$$\nabla C(\theta^0) = \begin{bmatrix} \partial C(\theta^0)/\partial w_1 \\ \partial C(\theta^0)/\partial w_2 \end{bmatrix}$$

Randomly pick a starting point θ^0

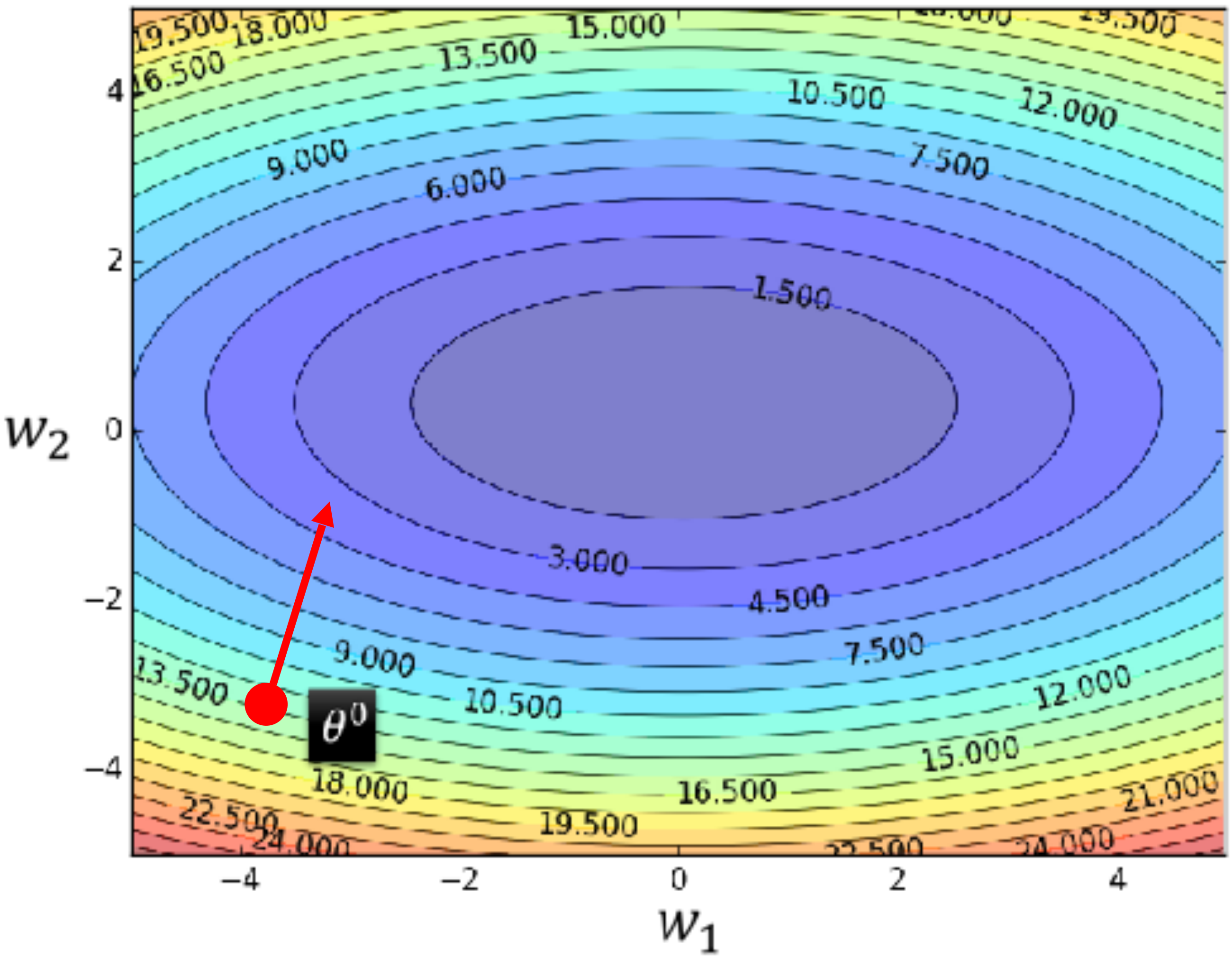
Compute the negative gradient at θ^0

$\Rightarrow -\nabla C(\theta^0)$

Times the learning rate η

$\Rightarrow -\eta \nabla C(\theta^0)$

Gradient Descent



Randomly pick a starting point θ^0

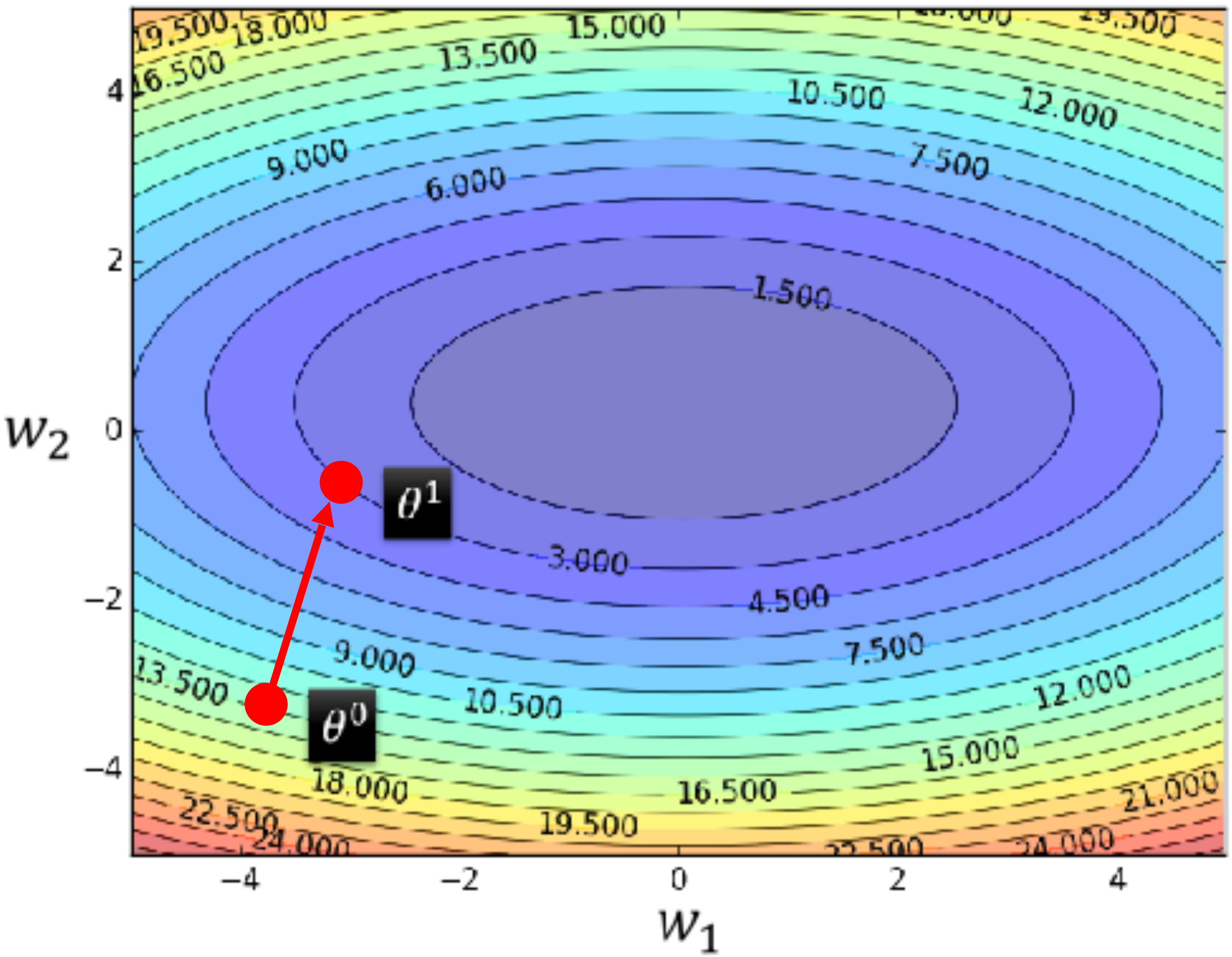
Compute the negative gradient at θ^0

$$\rightarrow -\nabla C(\theta^0)$$

Times the learning rate η

$$\rightarrow -\eta \nabla C(\theta^0)$$

Gradient Descent



Randomly pick a starting point θ^0

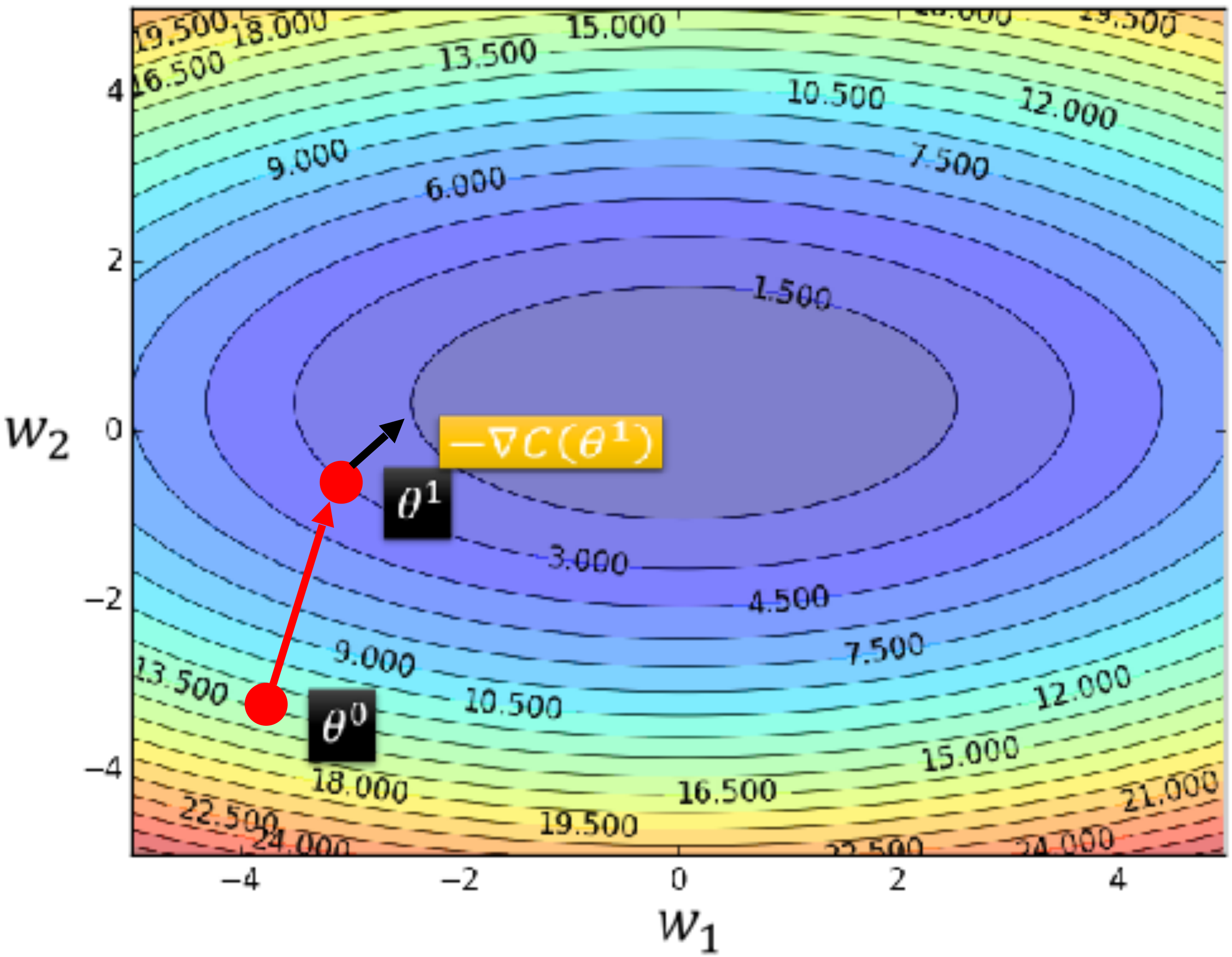
Compute the negative gradient at θ^0

$\rightarrow -\nabla C(\theta^0)$

Times the learning rate η

$\rightarrow -\eta \nabla C(\theta^0)$

Gradient Descent



Randomly pick a starting point θ^0

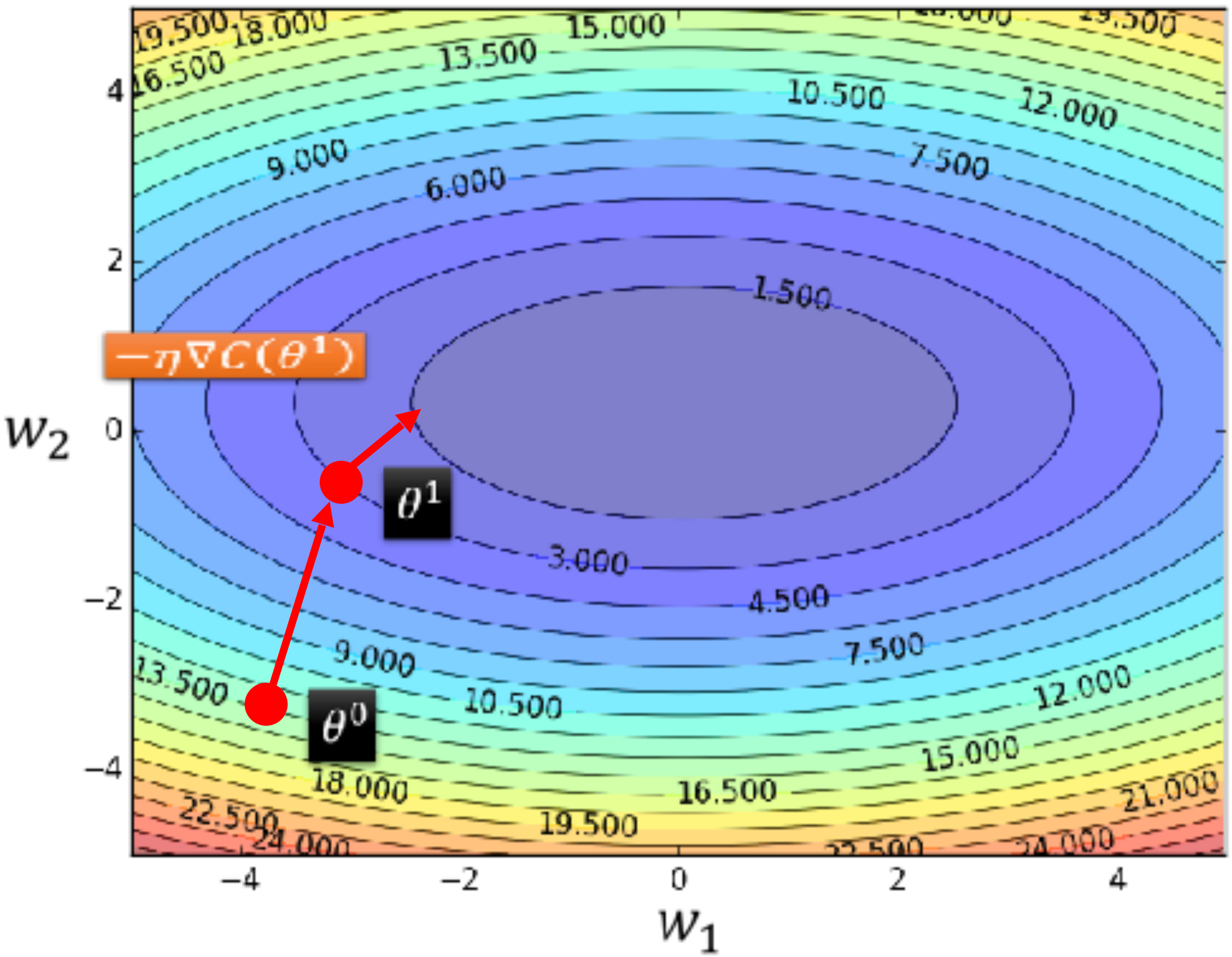
Compute the negative gradient at θ^0

$\rightarrow -\nabla C(\theta^0)$

Times the learning rate η

$\rightarrow -\eta \nabla C(\theta^0)$

Gradient Descent



Randomly pick a starting point θ^0

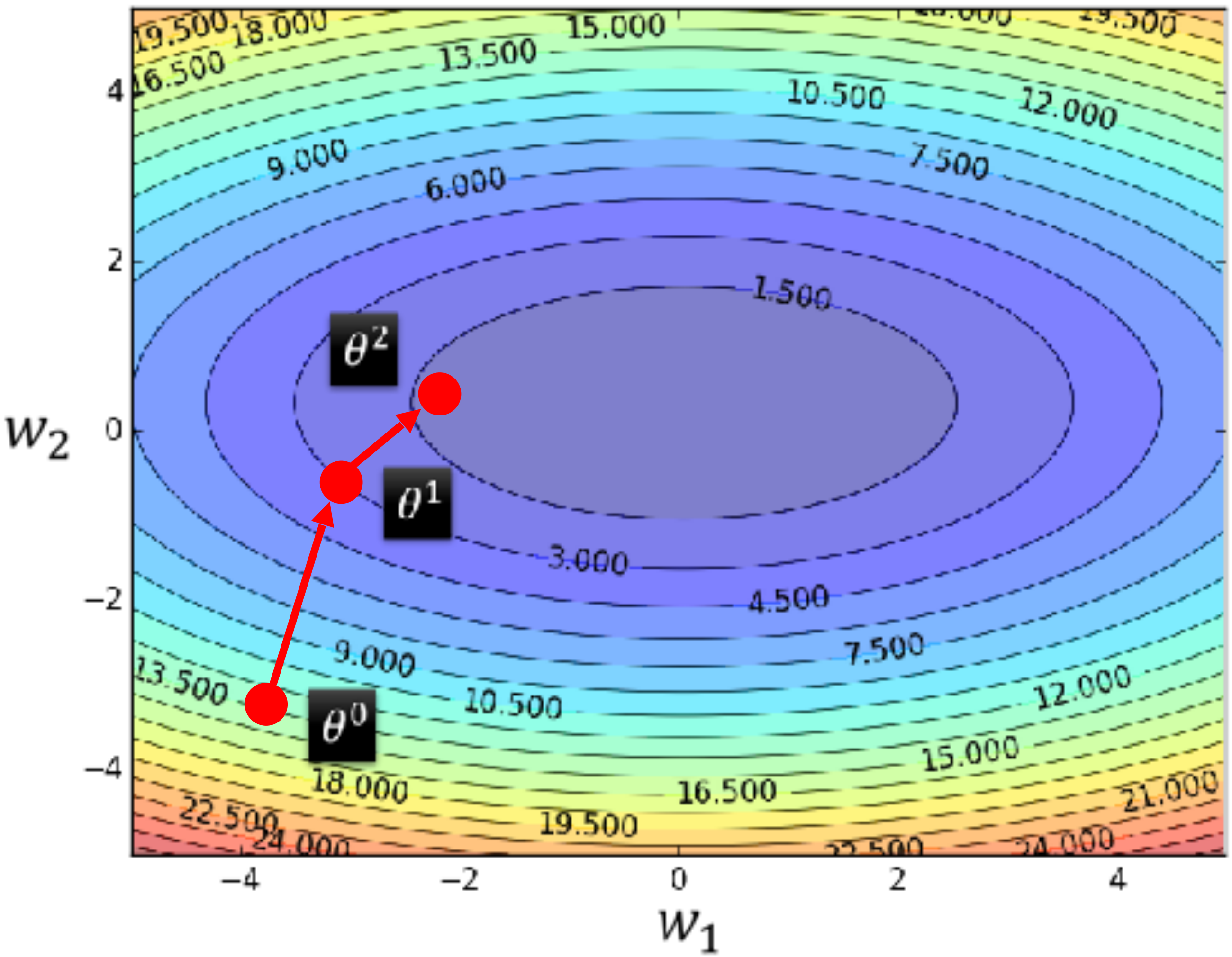
Compute the negative gradient at θ^0

$\Rightarrow -\nabla C(\theta^0)$

Times the learning rate η

$\Rightarrow -\eta\nabla C(\theta^0)$

Gradient Descent



Randomly pick a starting point θ^0

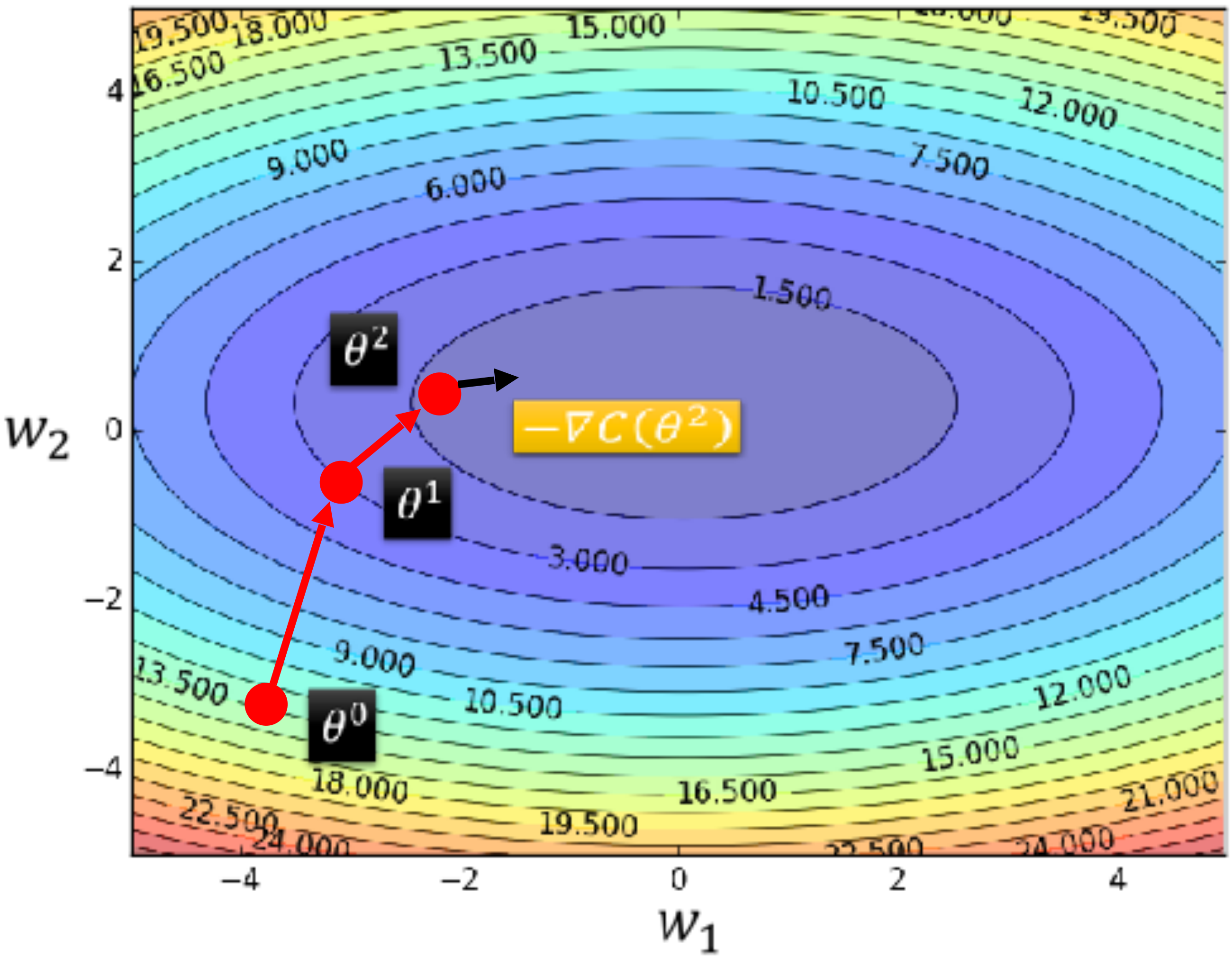
Compute the negative gradient at θ^0

$-\nabla C(\theta^0)$

Times the learning rate η

$-\eta \nabla C(\theta^0)$

Gradient Descent



Randomly pick a starting point θ^0

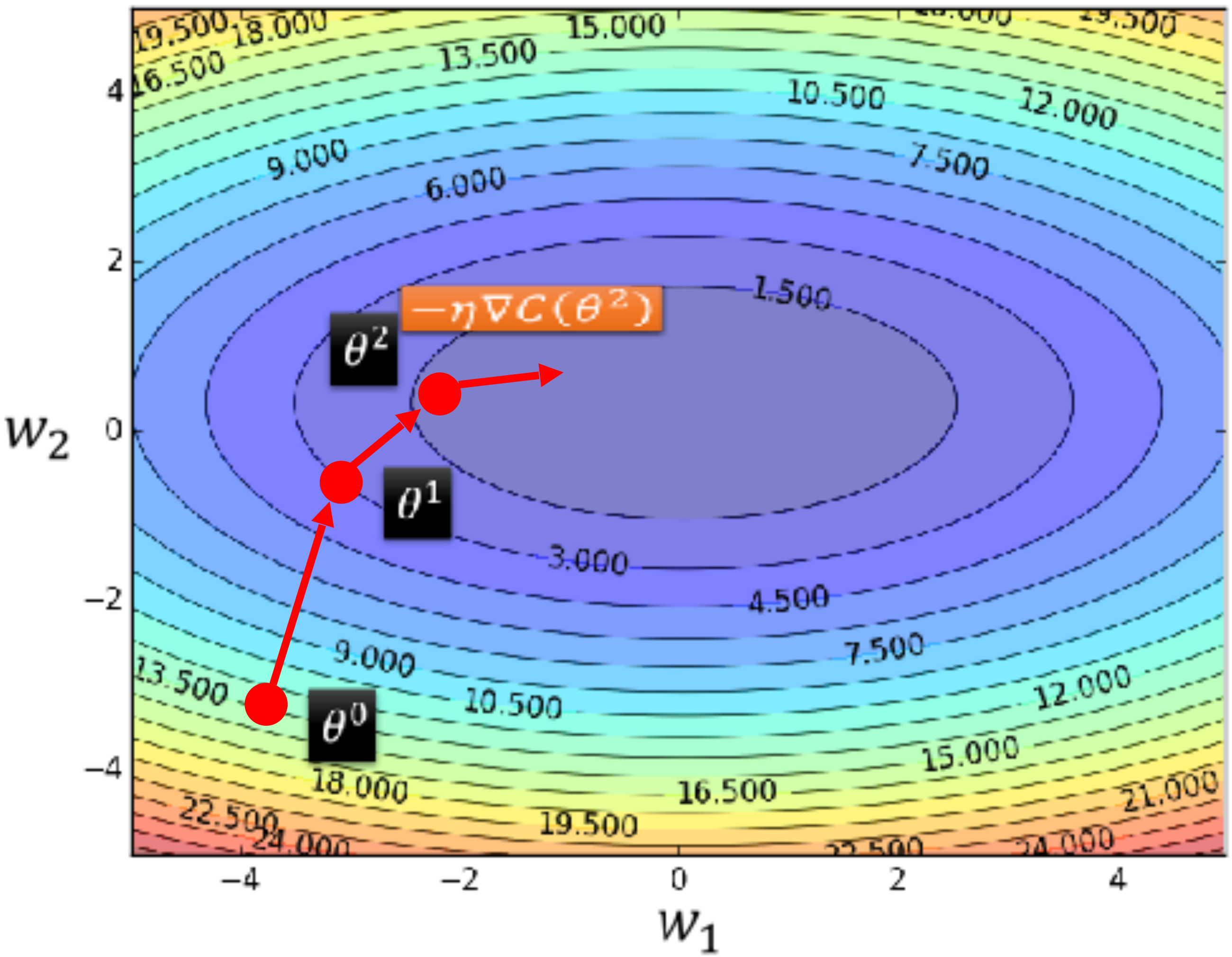
Compute the negative gradient at θ^0

$\Rightarrow -\nabla C(\theta^0)$

Times the learning rate η

$\Rightarrow -\eta\nabla C(\theta^0)$

Gradient Descent



Randomly pick a starting point θ^0

Compute the negative gradient at θ^0

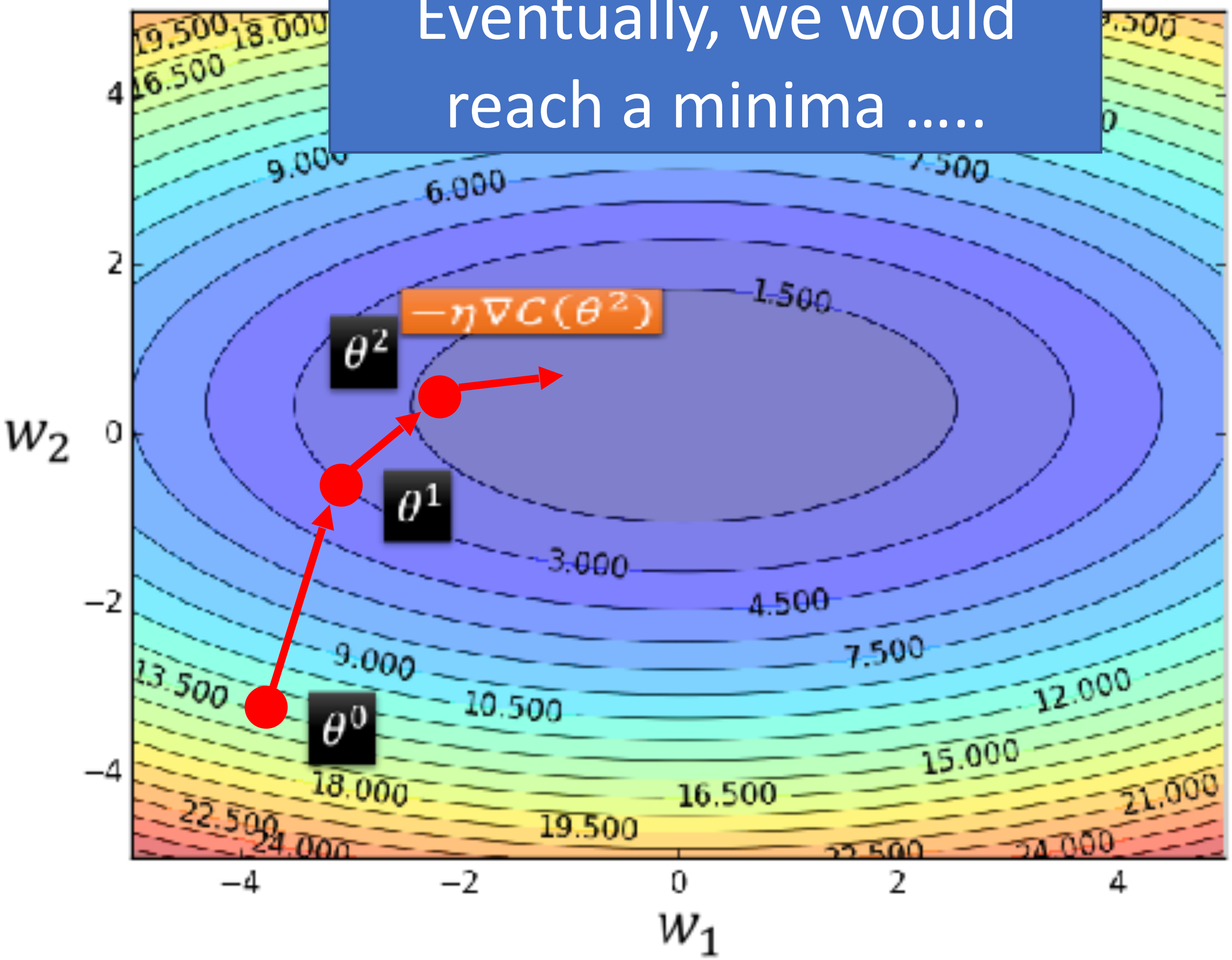
$\rightarrow -\nabla C(\theta^0)$

Times the learning rate η

$\rightarrow -\eta\nabla C(\theta^0)$

Gradient Descent

Eventually, we would reach a minima .....



Randomly pick a starting point θ^0

Compute the negative gradient at θ^0

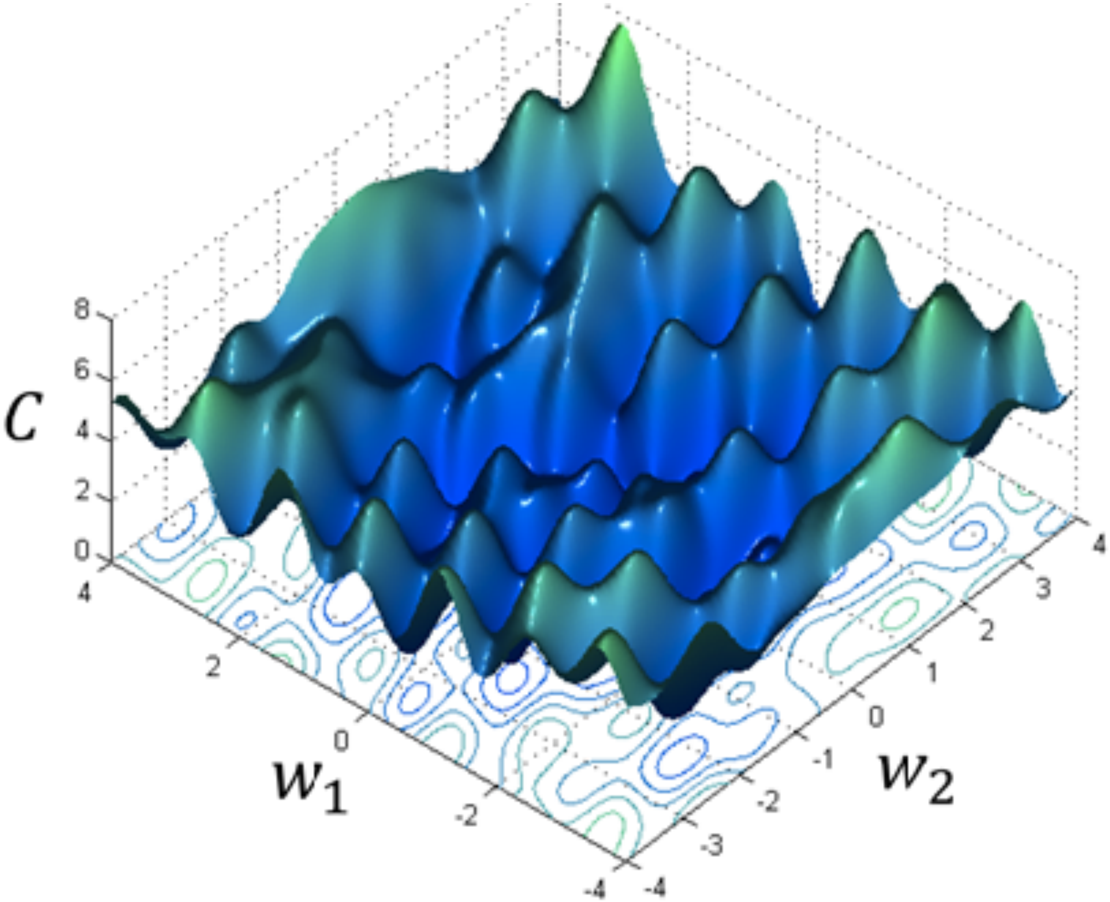
$\rightarrow -\nabla C(\theta^0)$

Times the learning rate η

$\rightarrow -\eta \nabla C(\theta^0)$

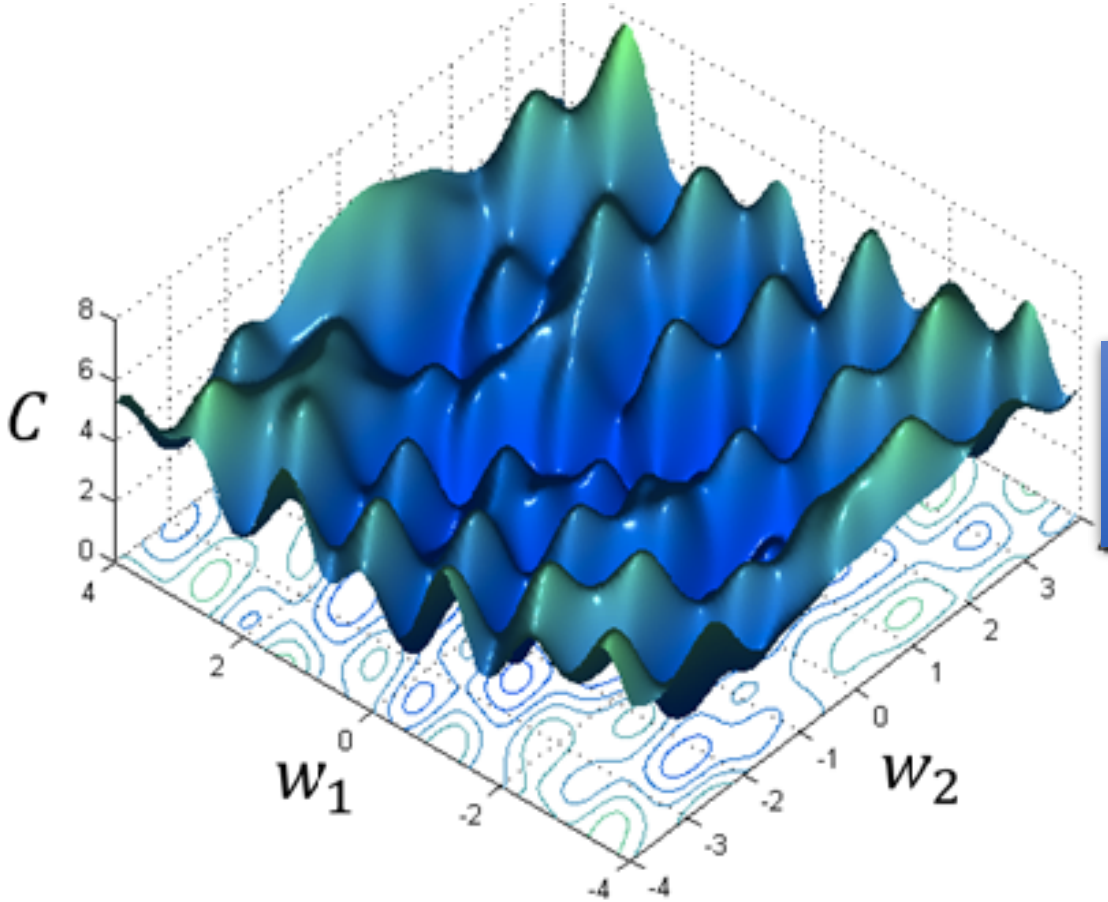
Local Minima

- Gradient descent never guarantee global minima



Local Minima

- Gradient descent never guarantee global minima

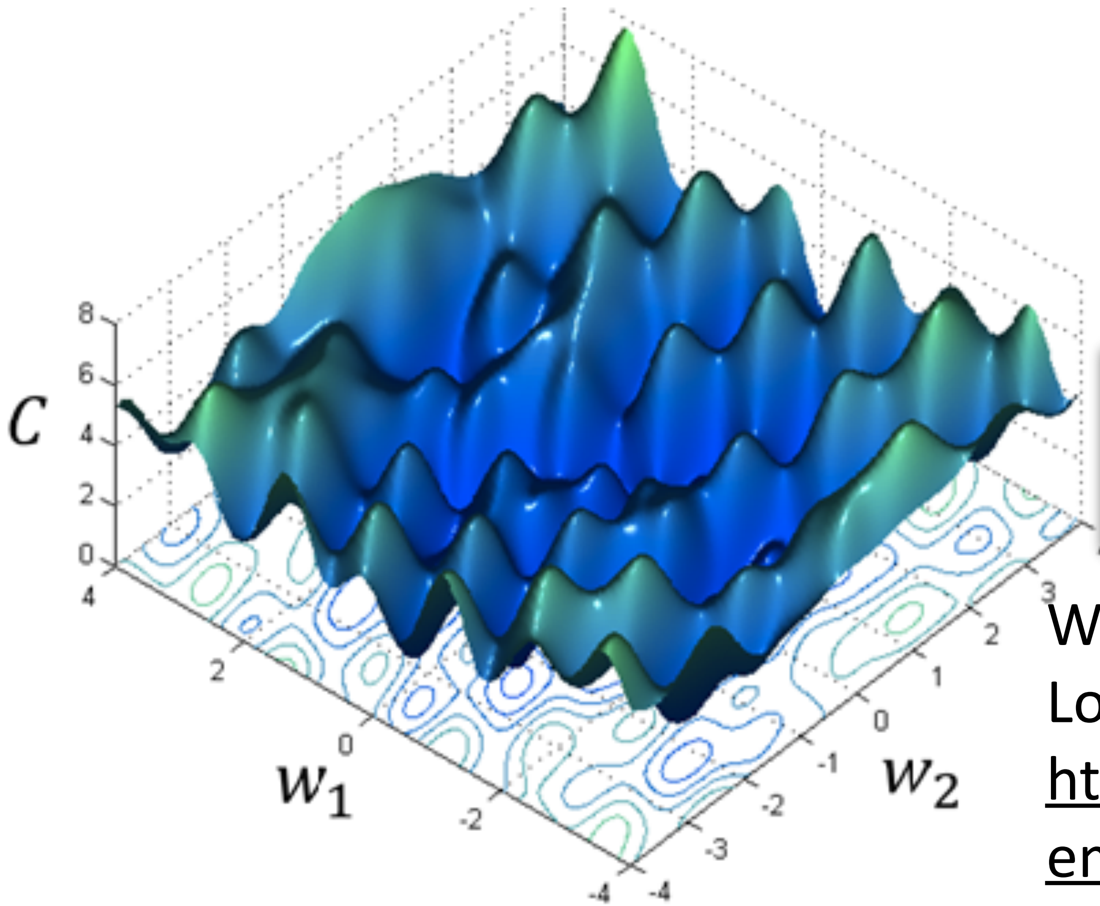


Different initial point θ^0

Reach different minima, so different results

Local Minima

- Gradient descent never guarantee global minima



Different initial point θ^0

Reach different minima, so different results

Who is Afraid of Non-Convex Loss Functions?
http://videolectures.net/eml07_lecun_wia/

Part II:
Why Deep?

Deeper is Better?

Layer X Size	Word Error Rate (%)
1 X 2k	24.2
2 X 2k	20.4
3 X 2k	18.4
4 X 2k	17.8
5 X 2k	17.2
7 X 2k	17.1

Seide, Frank, Gang Li, and Dong Yu. "Conversational Speech Transcription Using Context-Dependent Deep Neural Networks." *Interspeech*. 2011.

Deeper is Better?

Layer X Size	Word Error Rate (%)
1 X 2k	24.2
2 X 2k	20.4
3 X 2k	18.4
4 X 2k	17.8
5 X 2k	17.2
7 X 2k	17.1

Not surprised, more parameters, better performance

Seide, Frank, Gang Li, and Dong Yu. "Conversational Speech Transcription Using Context-Dependent Deep Neural Networks." *Interspeech*. 2011.

Universality Theorem

Universality Theorem

Any continuous function f

Universality Theorem

Any continuous function f

$$f : R^N \rightarrow R^{\mathrm{M}}$$

Universality Theorem

Any continuous function f

$$f : R^N \rightarrow R^{\mathrm{M}}$$

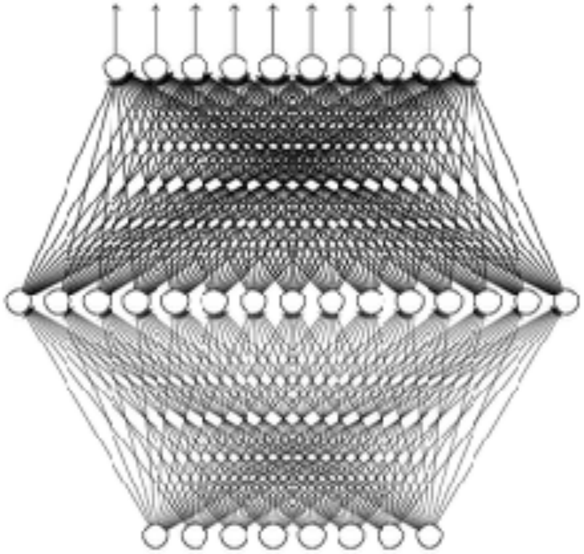
Can be realized by a network with one hidden layer

Universality Theorem

Any continuous function f

$$f : R^N \rightarrow R^{\mathrm{M}}$$

Can be realized by a network with one hidden layer



Reference for the reason: http://neuralnetworksanddeeplearning.com/chap4.html

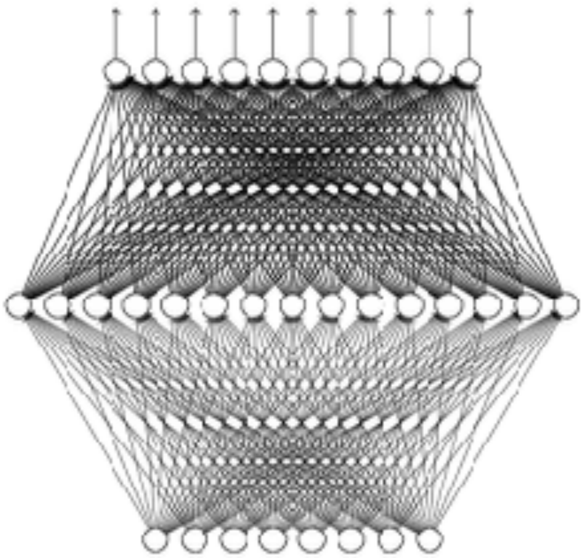
Universality Theorem

Any continuous function f

$$f : R^N \rightarrow R^M$$

Can be realized by a network with one hidden layer

(given **enough** hidden neurons)



Reference for the reason: http://neuralnetworksanddeeplearning.com/chap4.html

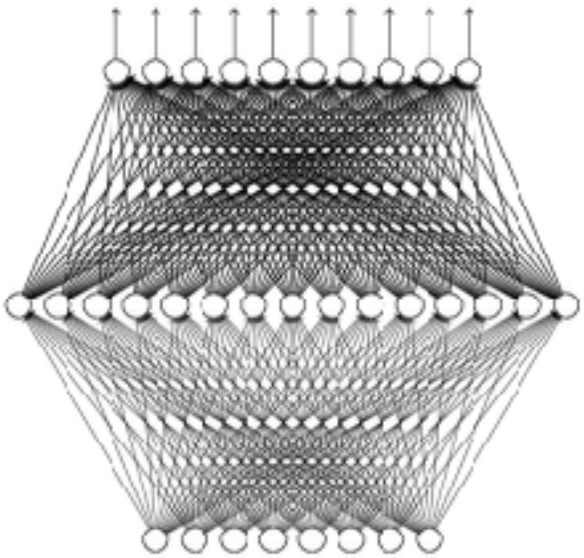
Universality Theorem

Any continuous function f

$$f : R^N \rightarrow R^{\mathrm{M}}$$

Can be realized by a network with one hidden layer

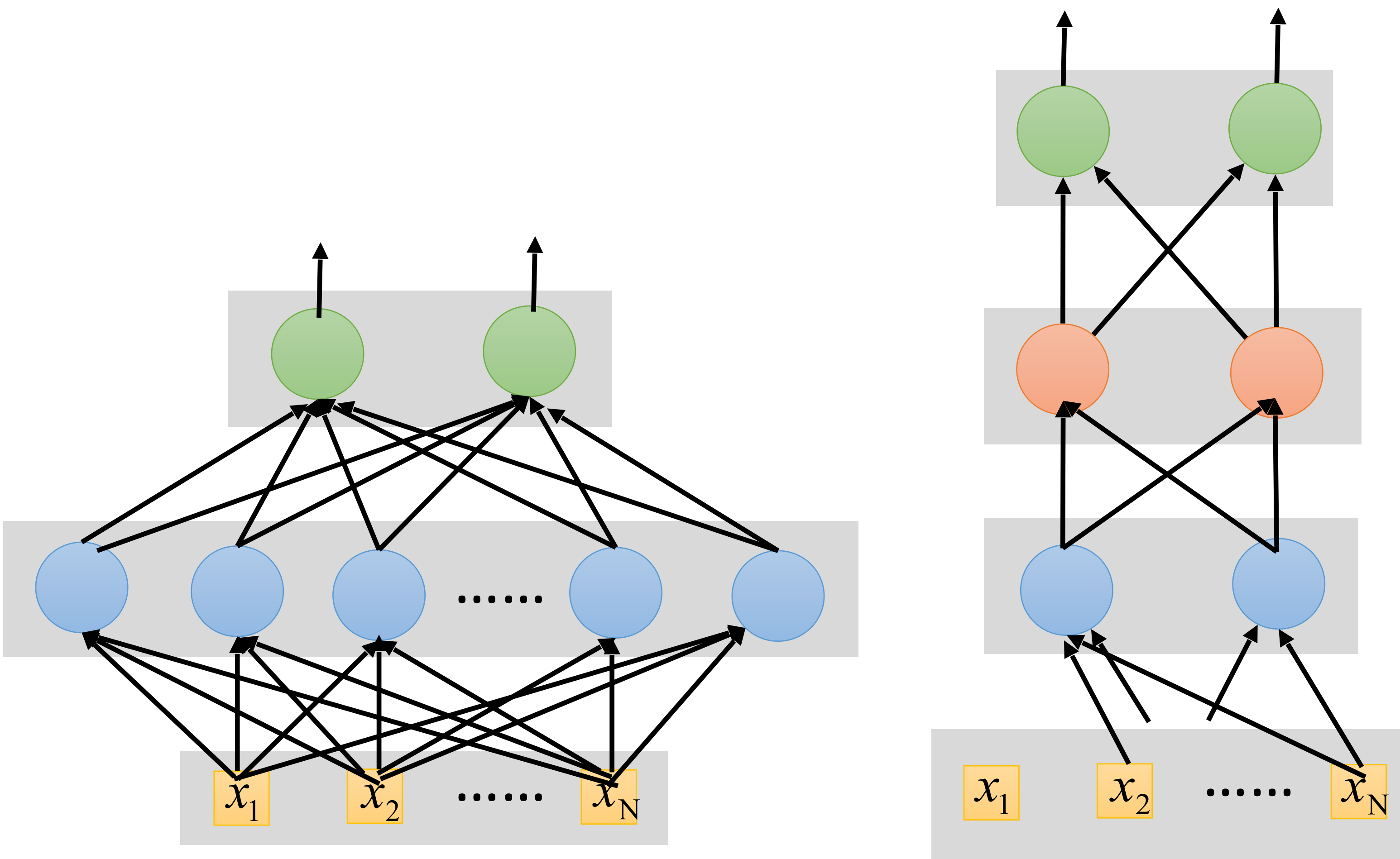
(given **enough** hidden neurons)



Reference for the reason: http://neuralnetworksanddeeplearning.com/chap4.html

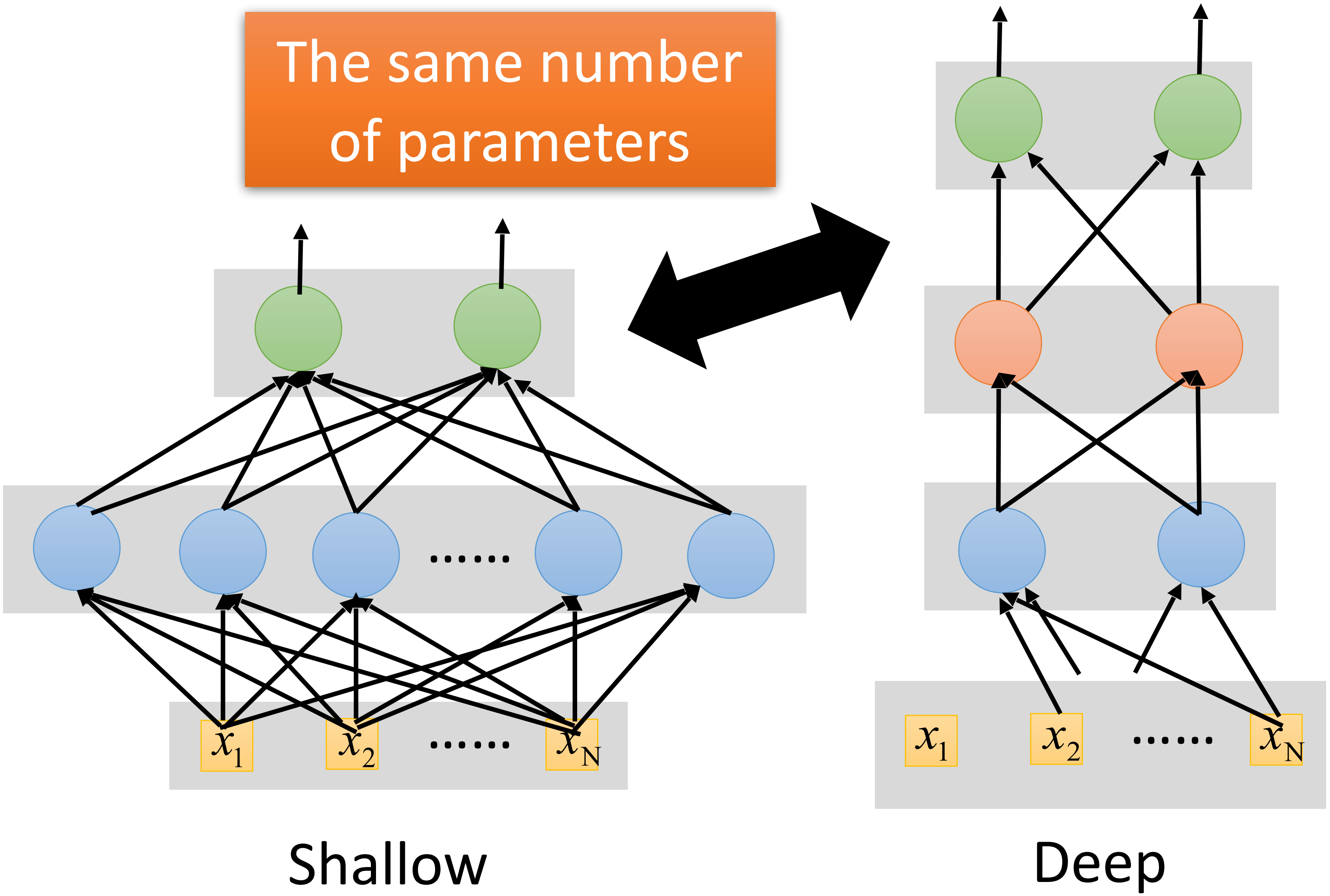
Why "Deep" neural network not "Fat" neural network?

Fat + Short v.s. Thin + Tall

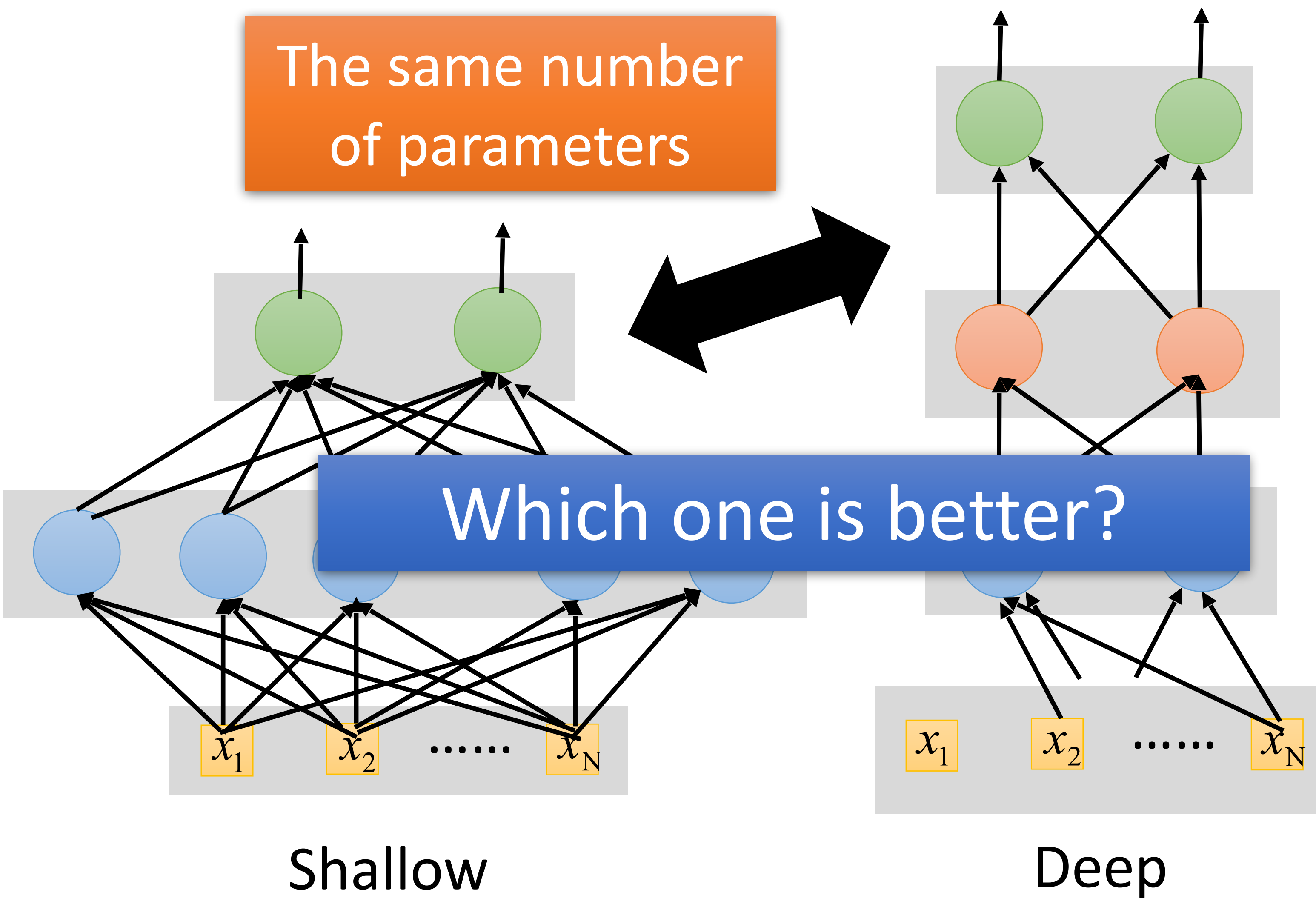


Shallow   Deep

Fat + Short v.s. Thin + Tall



Fat + Short v.s. Thin + Tall



Fat + Short v.s. Thin + Tall

Layer X Size	Word Error Rate (%)	Layer X Size	Word Error Rate (%)
1 X 2k	24.2		
2 X 2k	20.4		
3 X 2k	18.4		
4 X 2k	17.8		
5 X 2k	17.2	1 X 3772	22.5
7 X 2k	17.1	1 X 4634	22.6
		1 X 16k	22.1

Seide, Frank, Gang Li, and Dong Yu. "Conversational Speech Transcription Using Context-Dependent Deep Neural Networks." *Interspeech*. 2011.

Fat + Short v.s. Thin + Tall

Layer X Size	Word Error Rate (%)	Layer X Size	Word Error Rate (%)
1 X 2k	24.2		
2 X 2k	20.4		
3 X 2k	18.4		
4 X 2k	17.8		
5 X 2k	17.2	1 X 3772	22.5
7 X 2k	17.1	1 X 4634	22.6
		1 X 16k	22.1

Seide, Frank, Gang Li, and Dong Yu. "Conversational Speech Transcription Using Context-Dependent Deep Neural Networks." *Interspeech*. 2011.

Fat + Short v.s. Thin + Tall

Layer X Size	Word Error Rate (%)	Layer X Size	Word Error Rate (%)
1 X 2k	24.2		
2 X 2k	20.4		
3 X 2k	18.4		
4 X 2k	17.8		
5 X 2k	17.2	1 X 3772	22.5
7 X 2k	17.1	1 X 4634	22.6
		1 X 16k	22.1

Seide, Frank, Gang Li, and Dong Yu. "Conversational Speech Transcription Using Context-Dependent Deep Neural Networks." *Interspeech*. 2011.

Fat + Short v.s. Thin + Tall

Layer X Size	Word Error Rate (%)	Layer X Size	Word Error Rate (%)
1 X 2k	24.2		
2 X 2k	20.4		
3 X 2k	18.4		
4 X 2k	17.8		
5 X 2k	17.2	1 X 3772	22.5
7 X 2k	17.1	1 X 4634	22.6
		1 X 16k	22.1

Seide, Frank, Gang Li, and Dong Yu. "Conversational Speech Transcription Using Context-Dependent Deep Neural Networks." *Interspeech*. 2011.

Fat + Short v.s. Thin + Tall

Layer X Size	Word Error Rate (%)	Layer X Size	Word Error Rate (%)
1 X 2k	24.2		
2 X 2k	20.4		
3 X 2k	18.4		
4 X 2k	17.8		
5 X 2k	17.2	1 X 3772	22.5
7 X 2k	17.1	1 X 4634	22.6
		1 X 16k	22.1

Seide, Frank, Gang Li, and Dong Yu. "Conversational Speech Transcription Using Context-Dependent Deep Neural Networks." *Interspeech*. 2011.

Fat + Short v.s. Thin + Tall

Layer X Size	Word Error Rate (%)	Layer X Size	Word Error Rate (%)
1 X 2k	24.2		
2 X 2k	20.4		
3 X 2k	18.4		
4 X 2k	17.8		
5 X 2k	17.2	1 X 3772	22.5
7 X 2k	17.1	1 X 4634	22.6
		1 X 16k	22.1

Seide, Frank, Gang Li, and Dong Yu. "Conversational Speech Transcription Using Context-Dependent Deep Neural Networks." *Interspeech*. 2011.

Why Deep?

- Deep → Modularization

Image

Girls with long hair

Boys with long hair

Girls with short hair

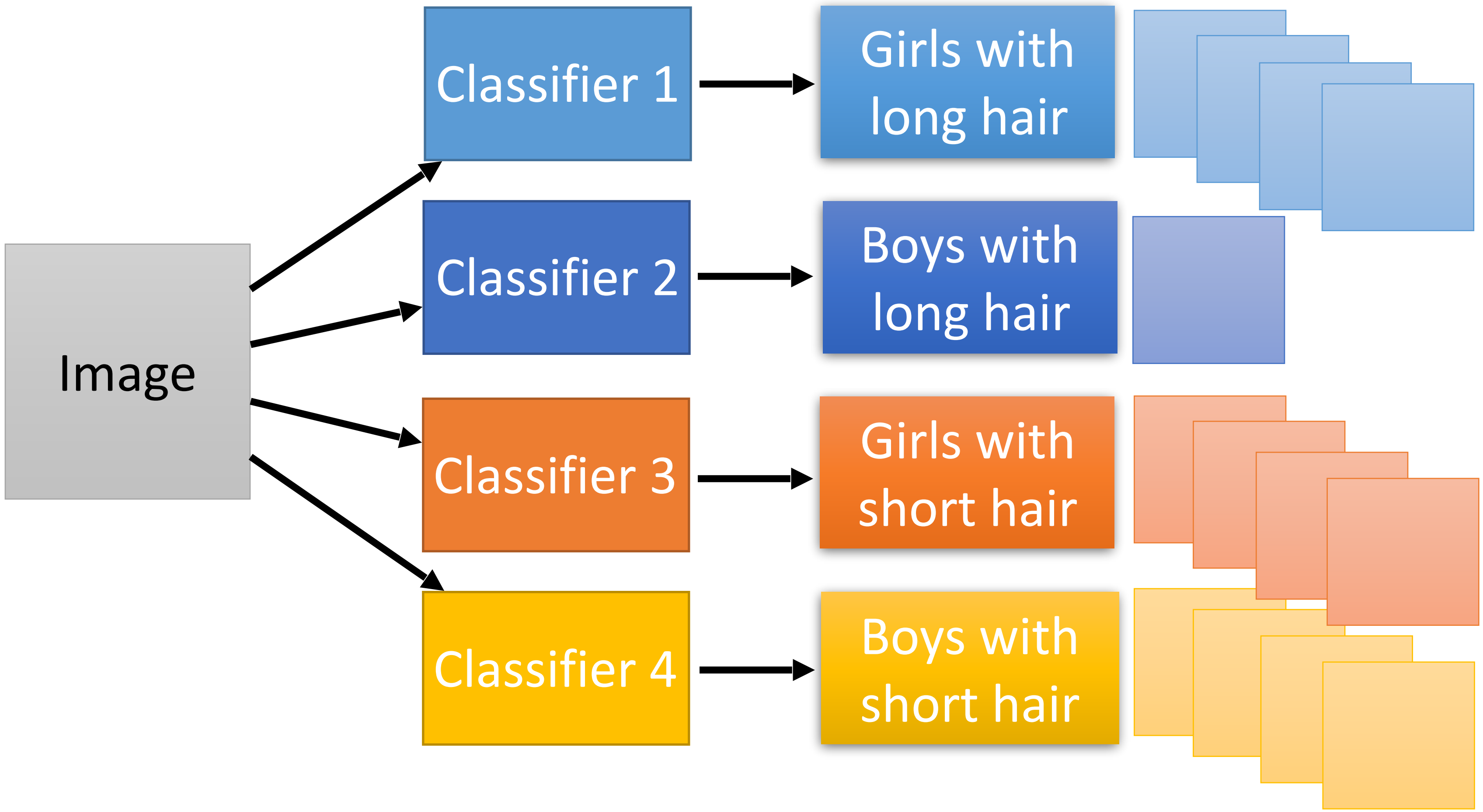
Boys with short hair

Why Deep?

- Deep → Modularization

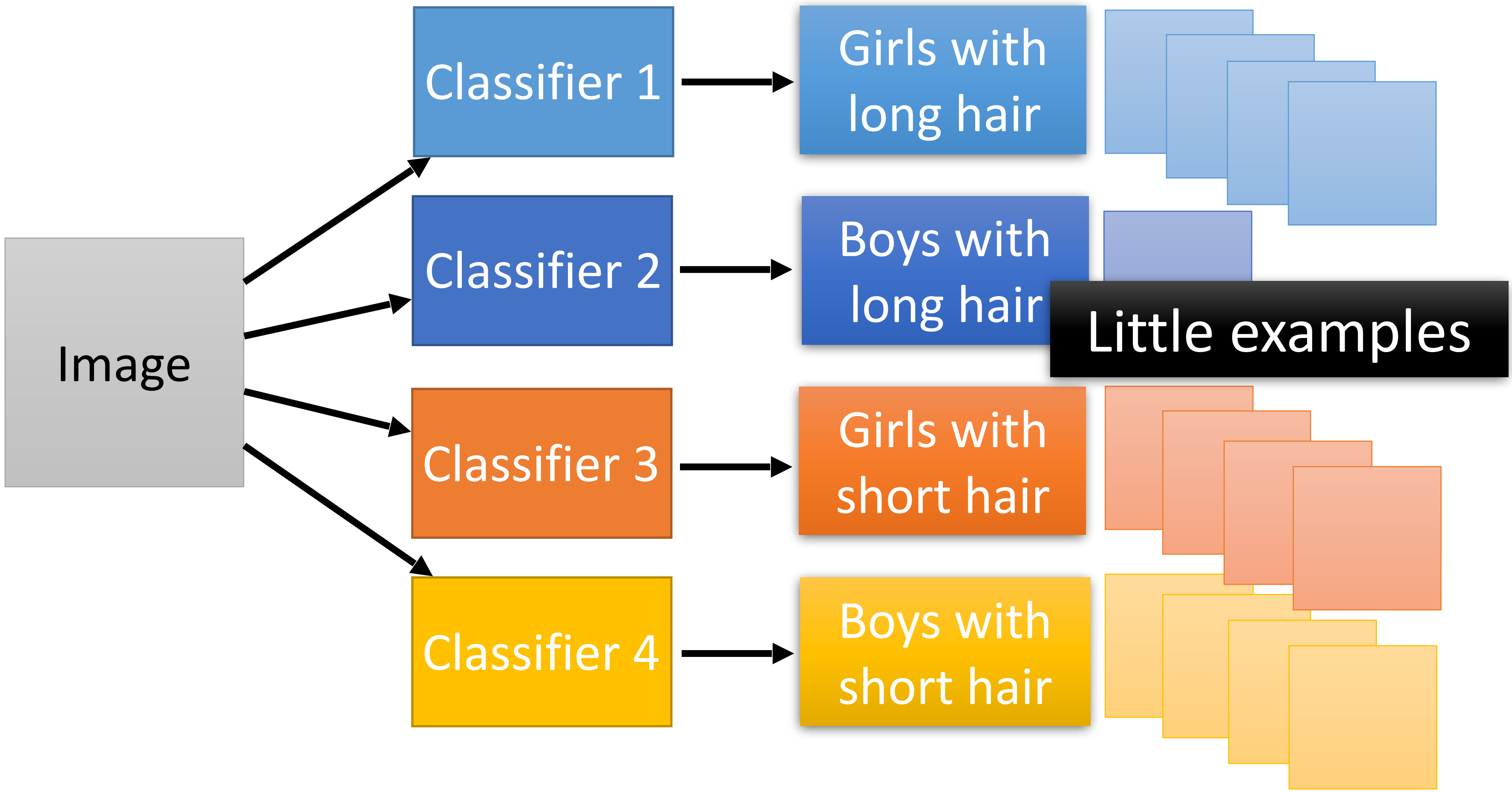
Why Deep?

- Deep → Modularization



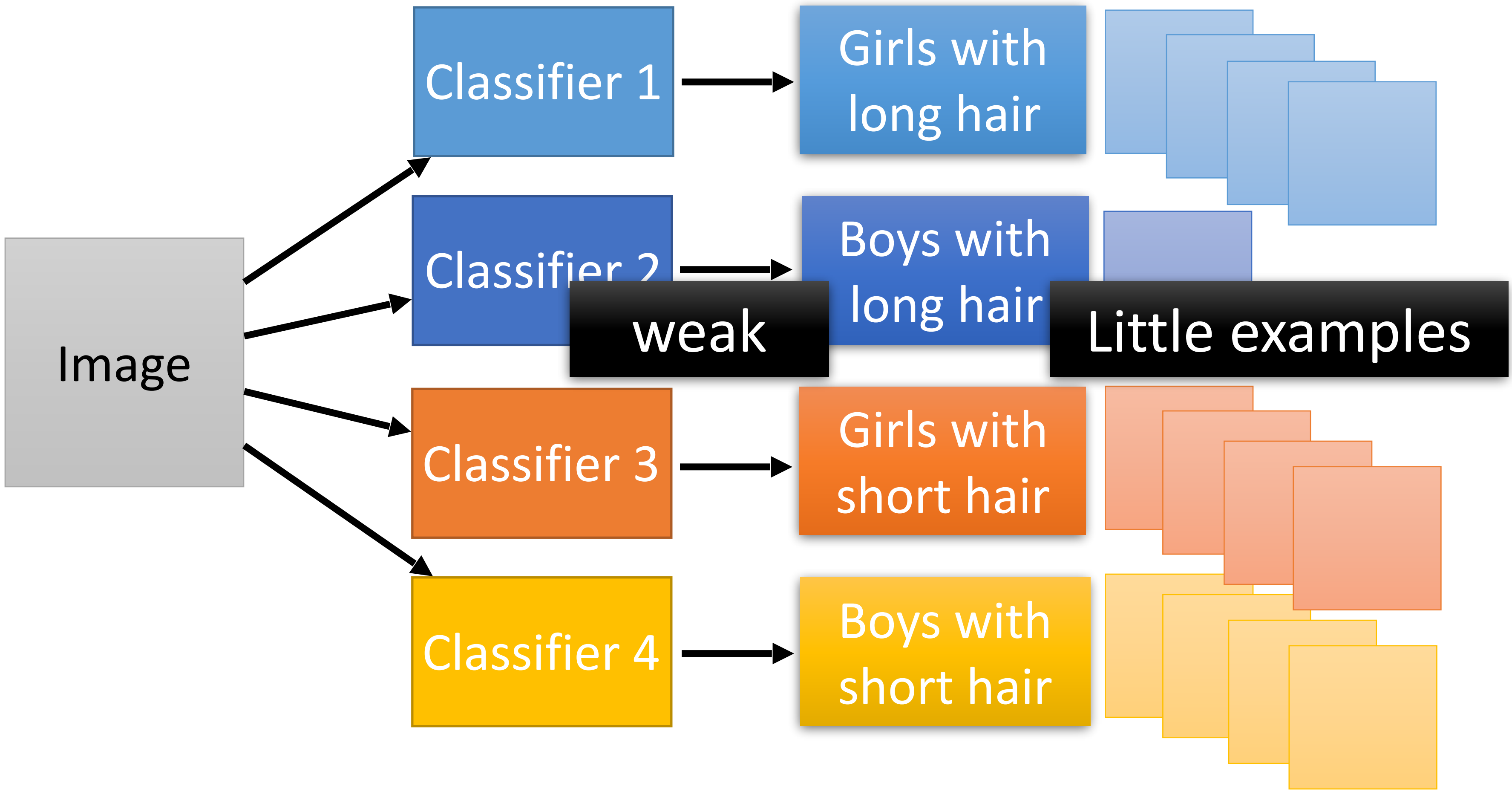
Why Deep?

- Deep → Modularization



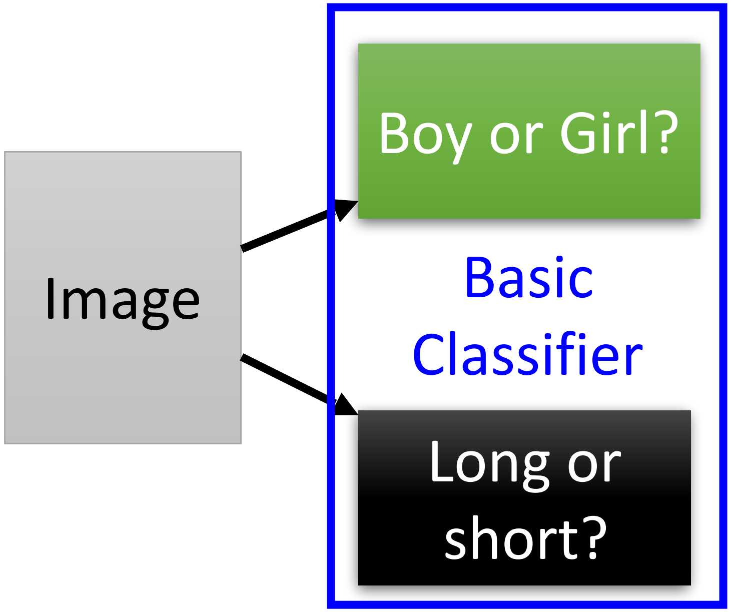
Why Deep?

- Deep → Modularization



Why Deep?

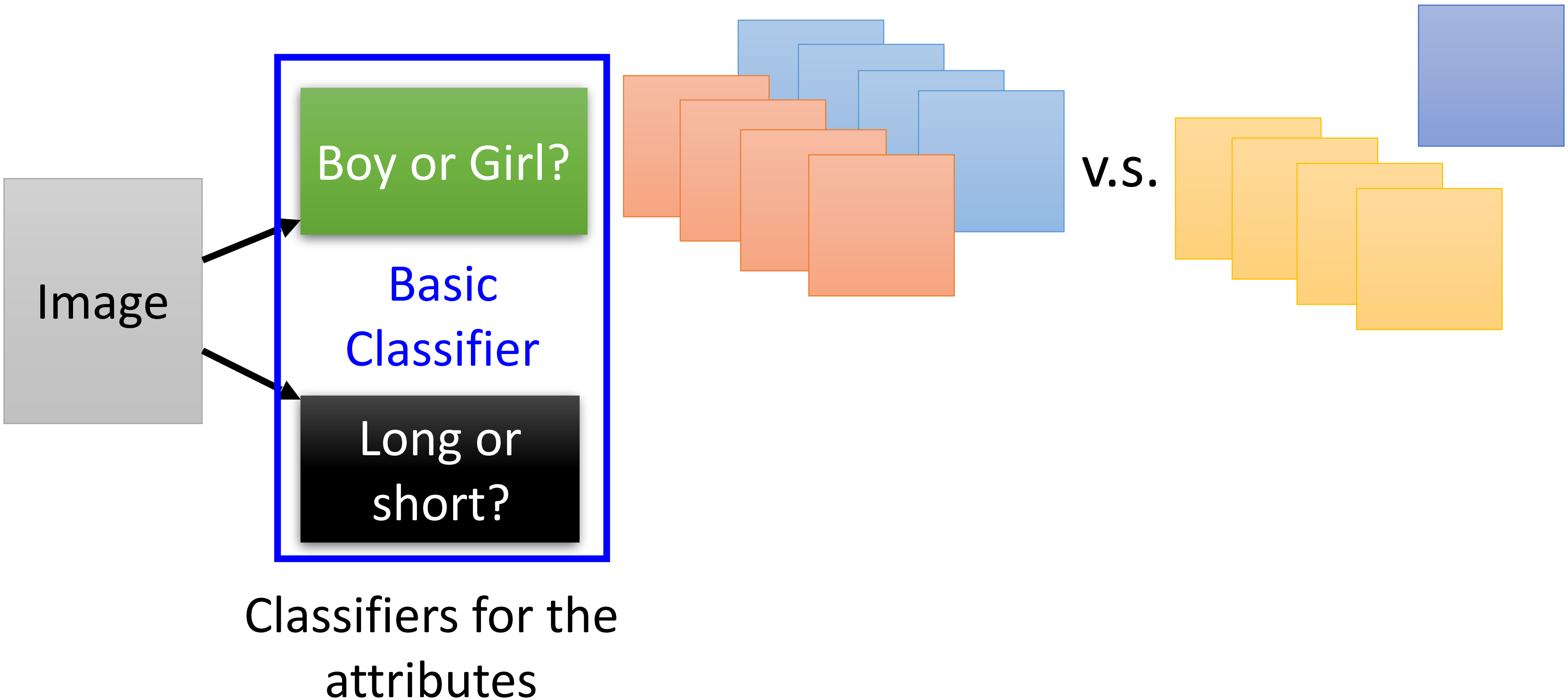
- Deep → Modularization



Classifiers for the attributes

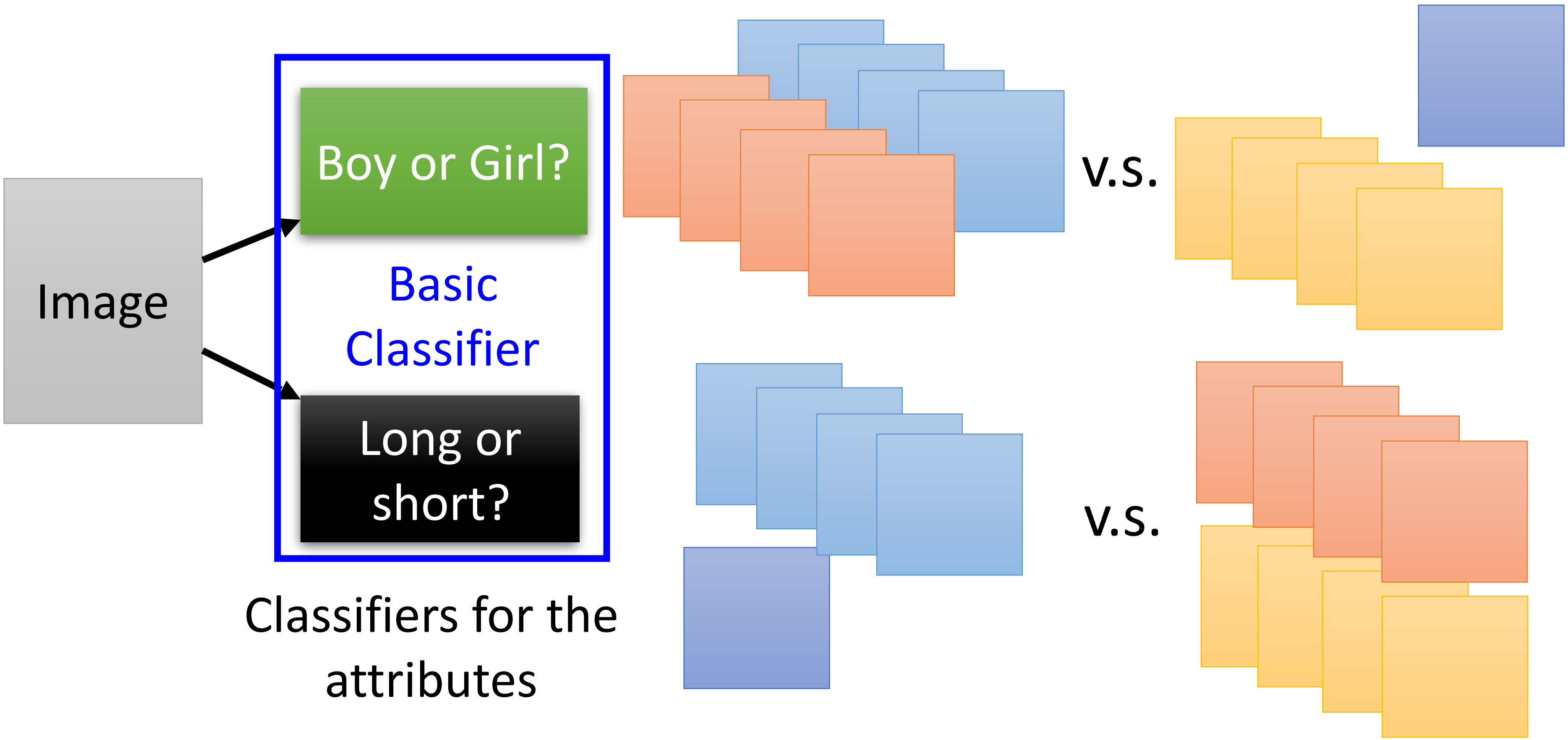
Why Deep?

- Deep → Modularization



Why Deep?

- Deep → Modularization

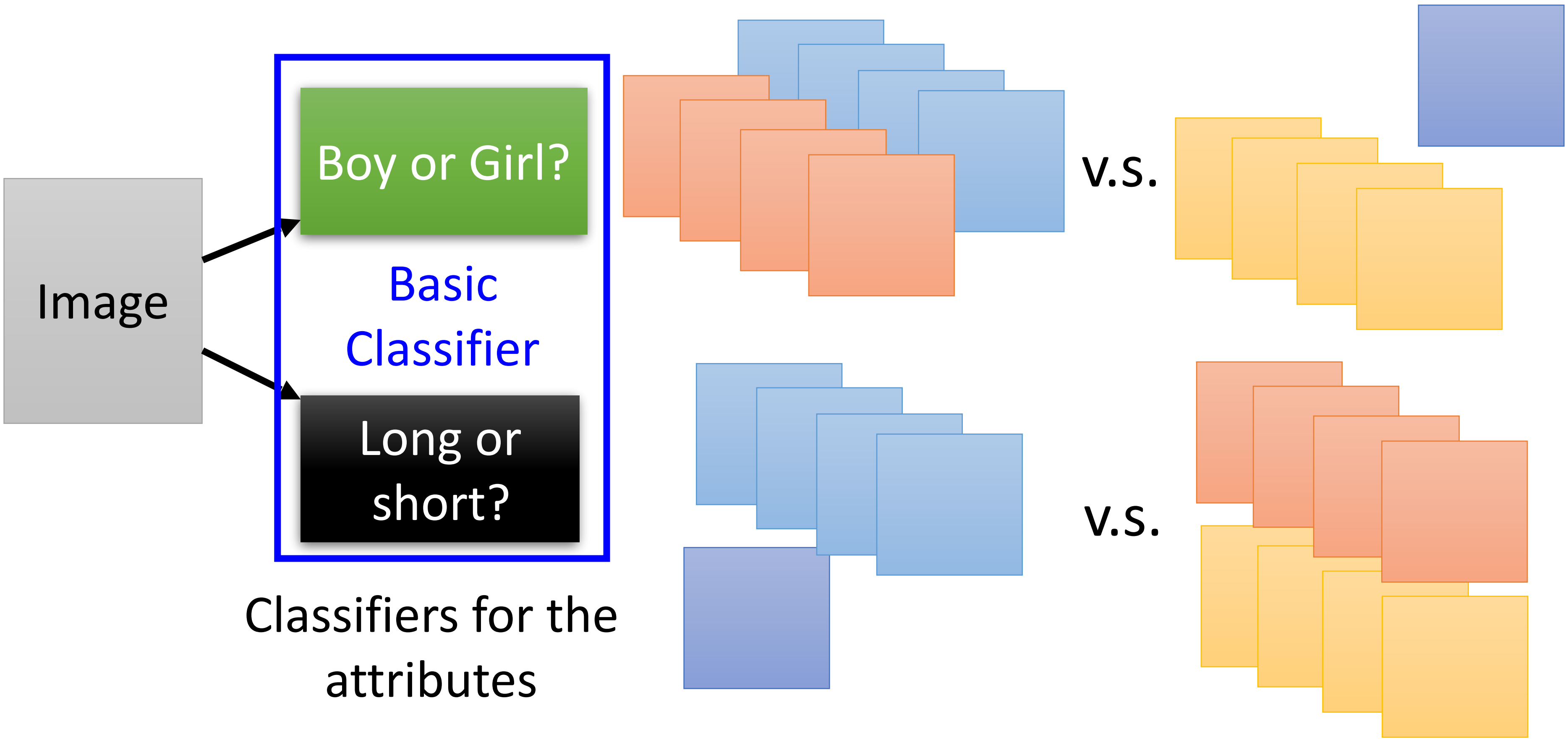


Classifiers for the attributes

Why Deep?

Each basic classifier can have sufficient training examples.

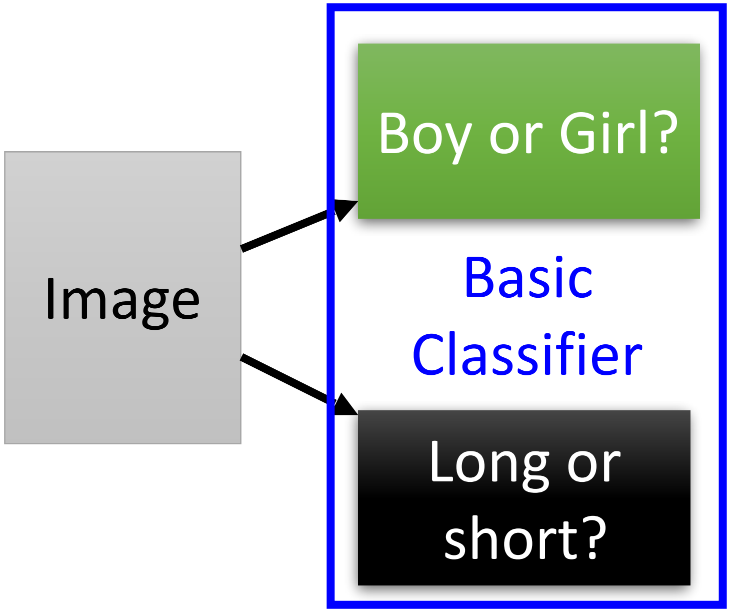
- Deep → Modularization



Classifiers for the attributes

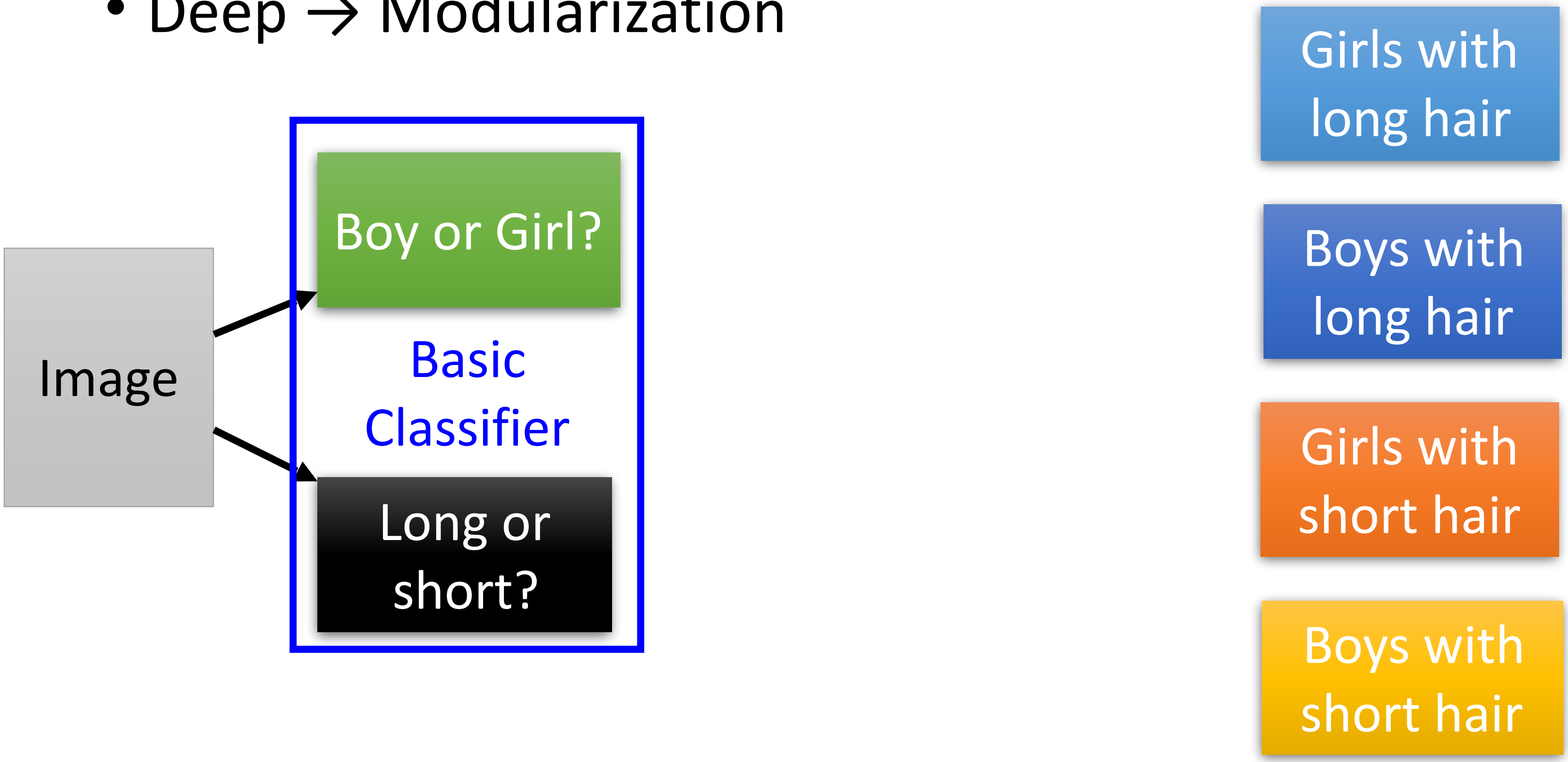
Why Deep?

- Deep → Modularization



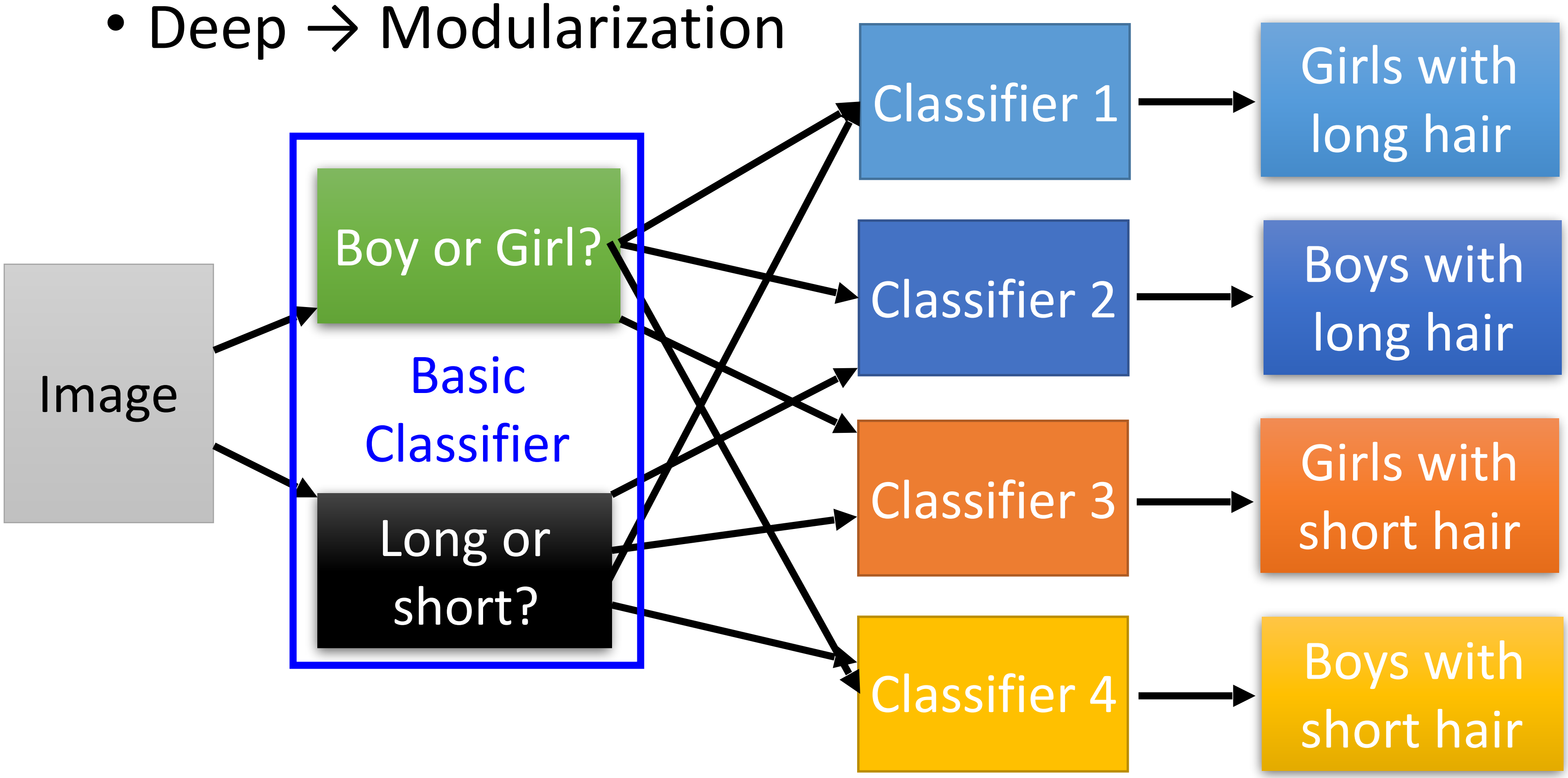
Why Deep?

- Deep → Modularization



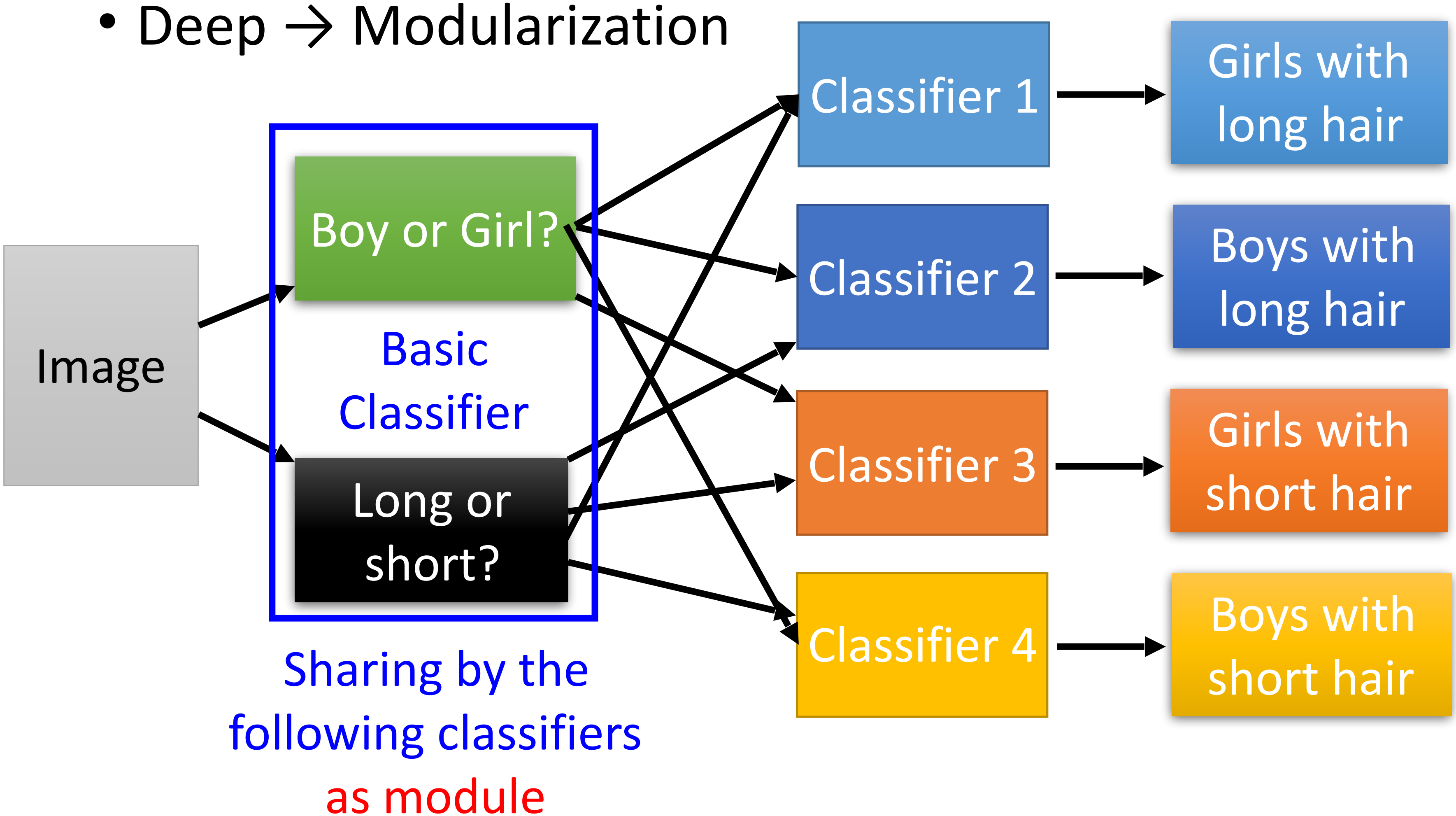
Why Deep?

- Deep → Modularization



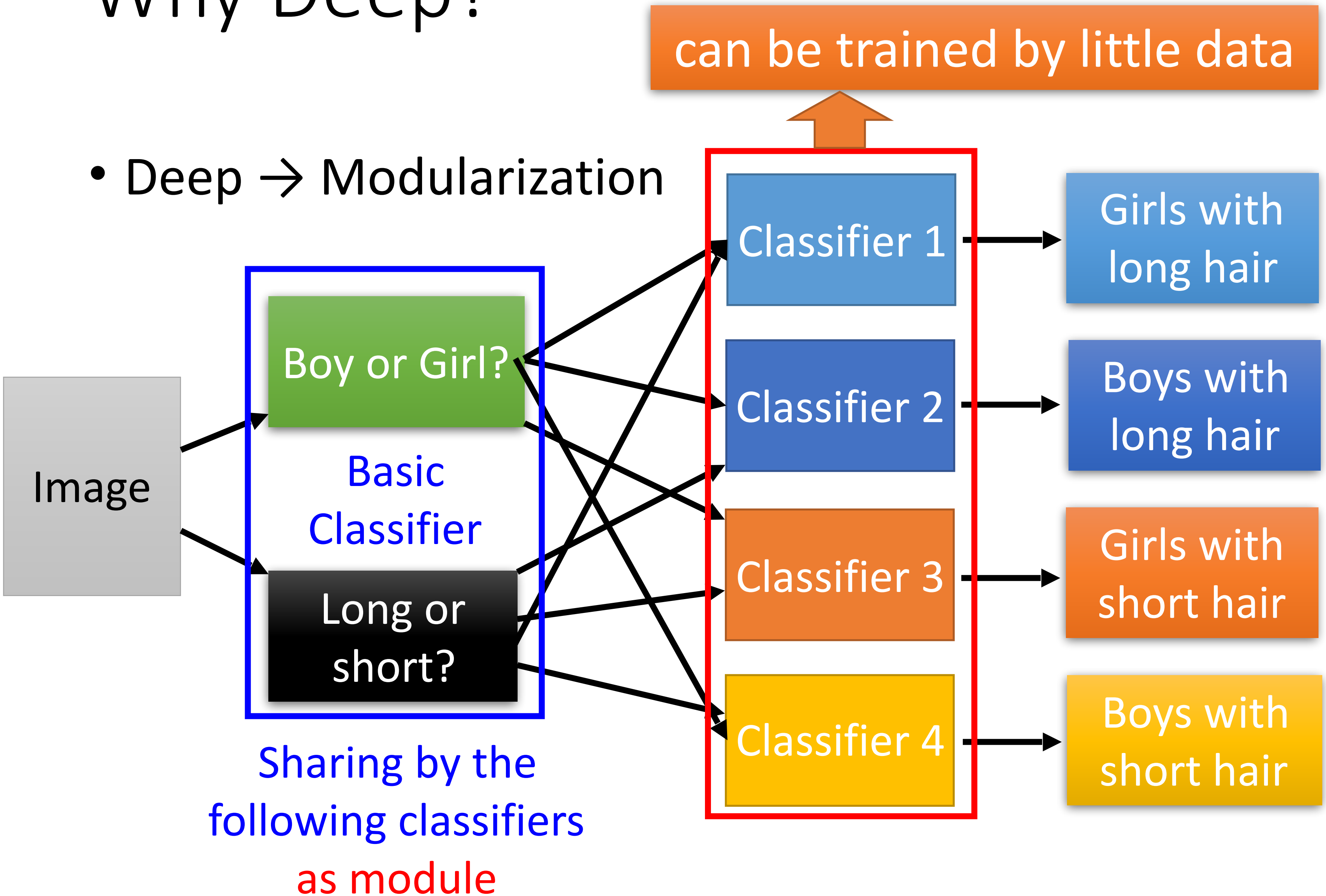
Why Deep?

- Deep → Modularization



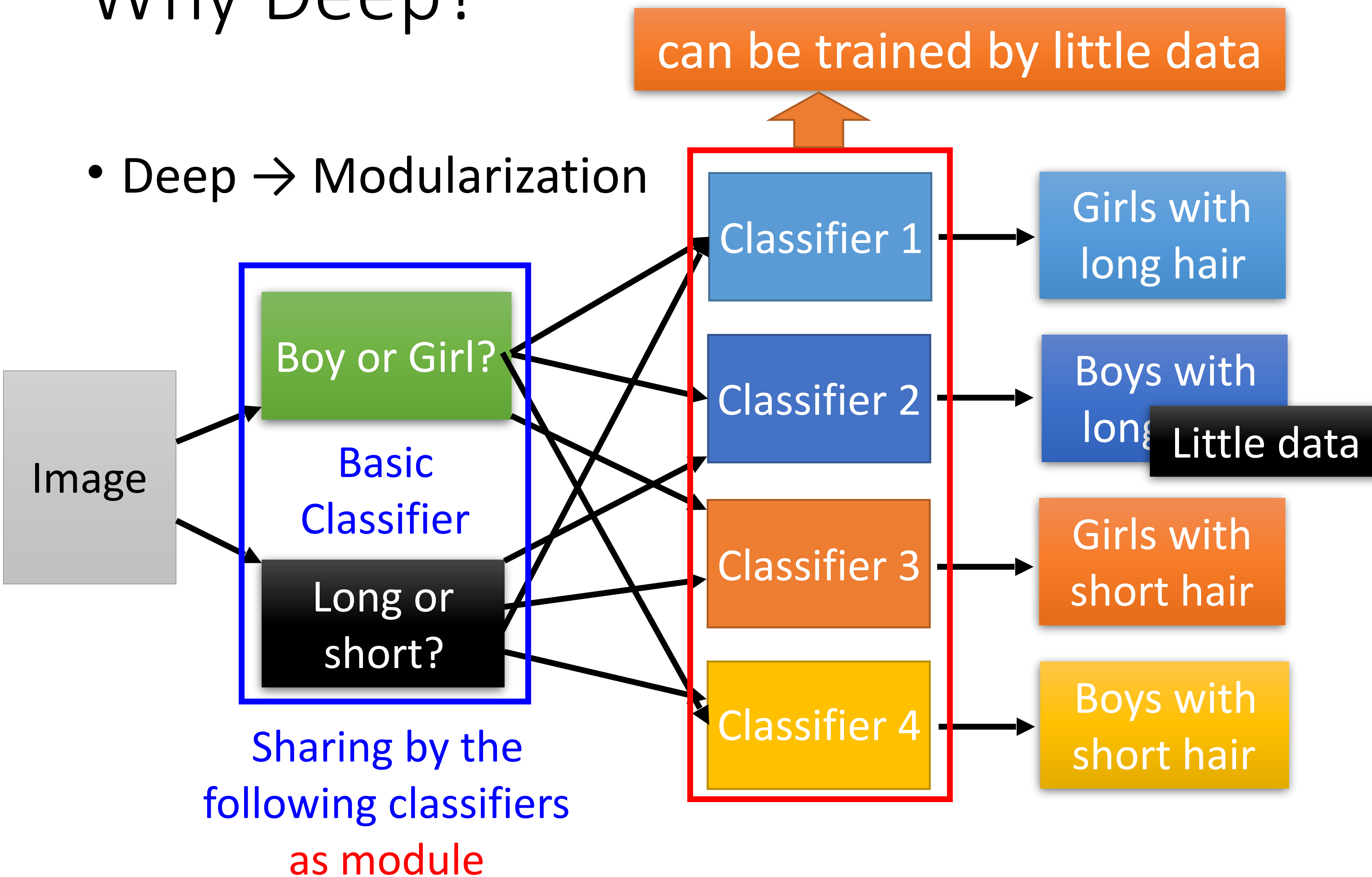
Why Deep?

- Deep → Modularization



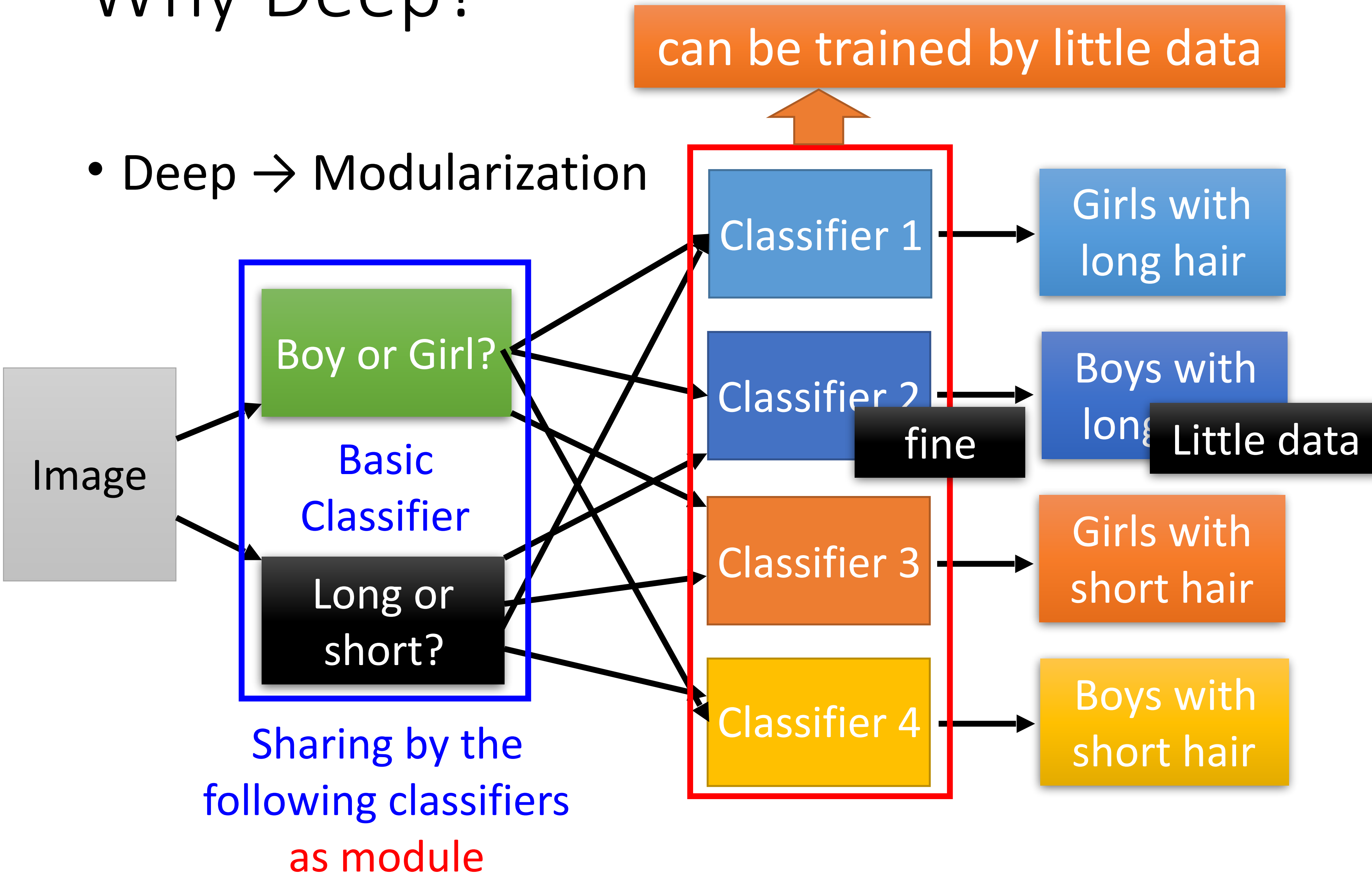
Why Deep?

- Deep → Modularization



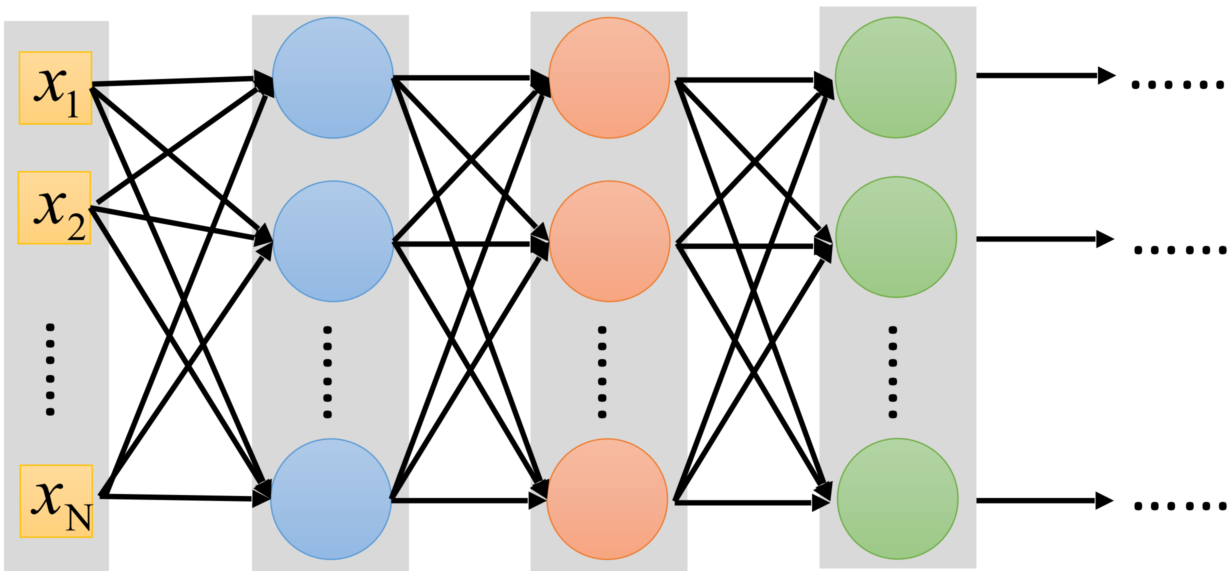
Why Deep?

- Deep → Modularization



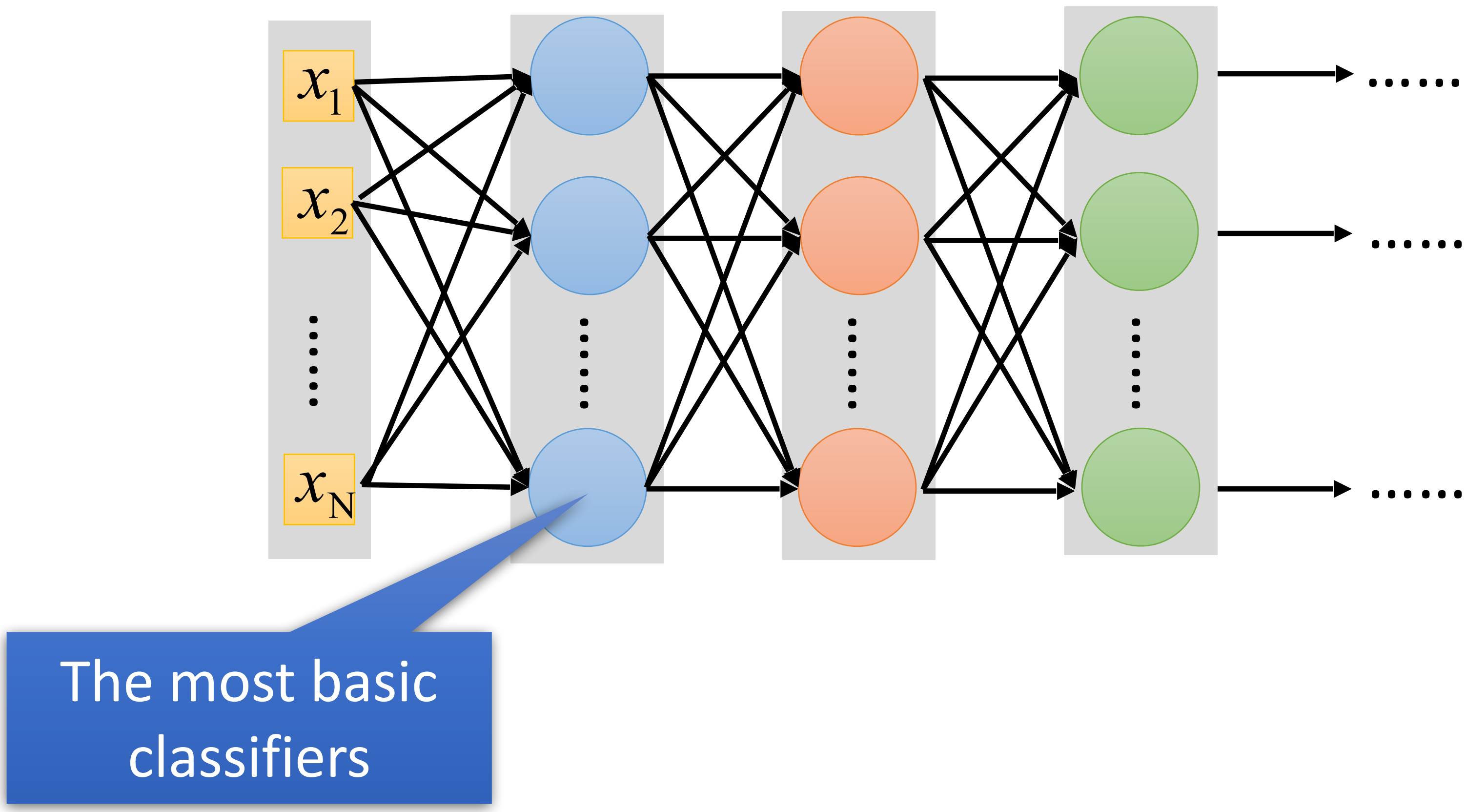
Why Deep?

- Deep → Modularization



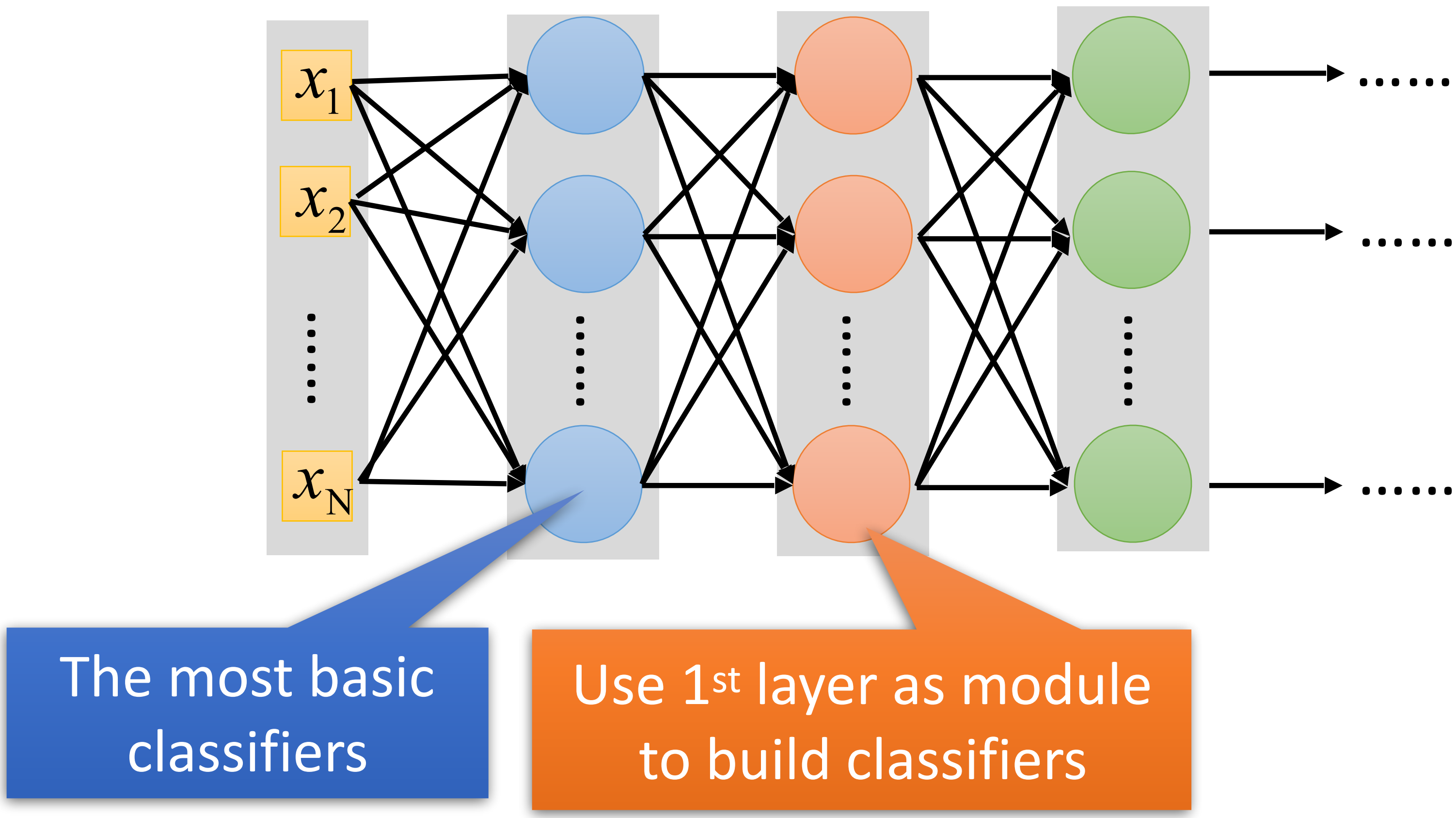
Why Deep?

- Deep → Modularization



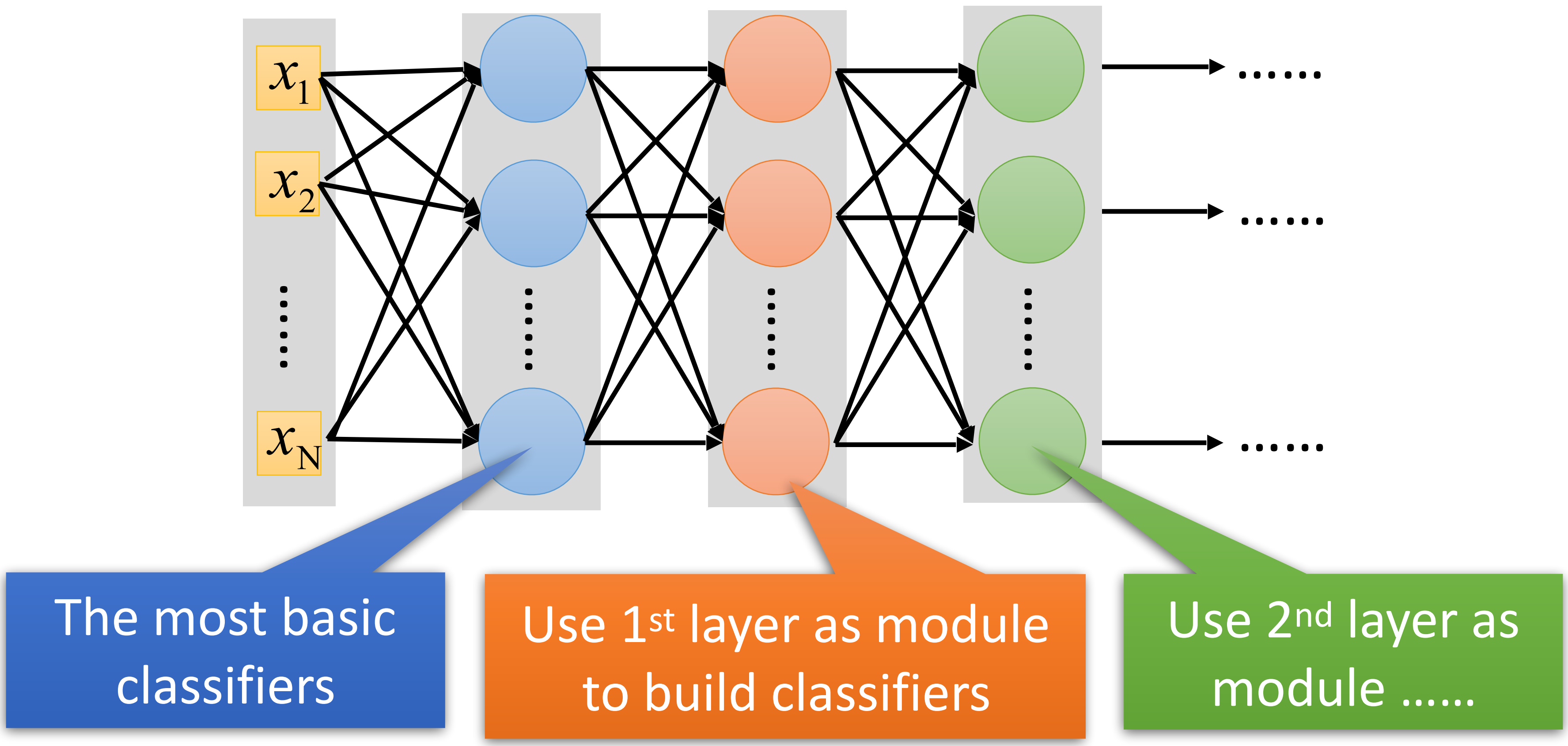
Why Deep?

- Deep → Modularization



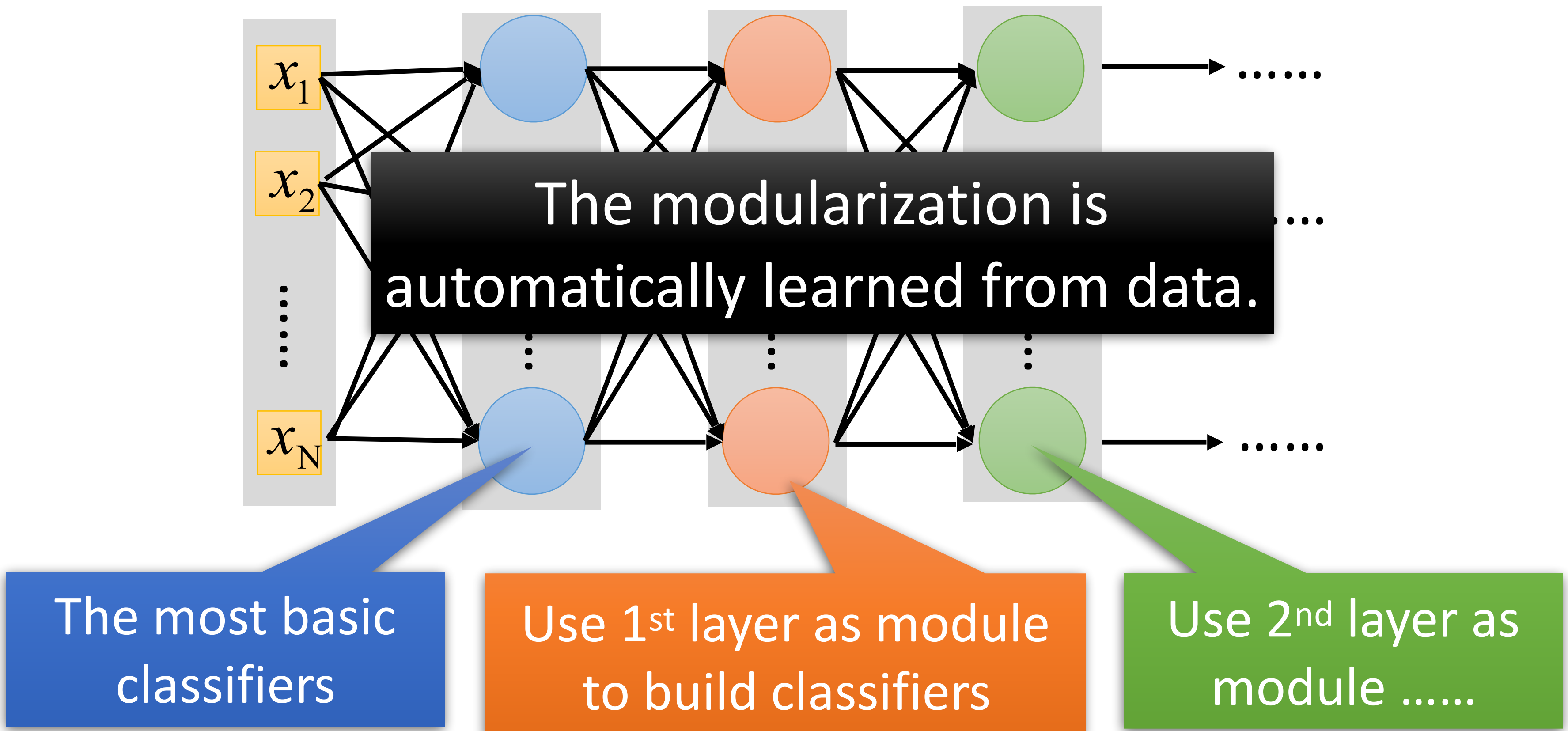
Why Deep?

- Deep → Modularization



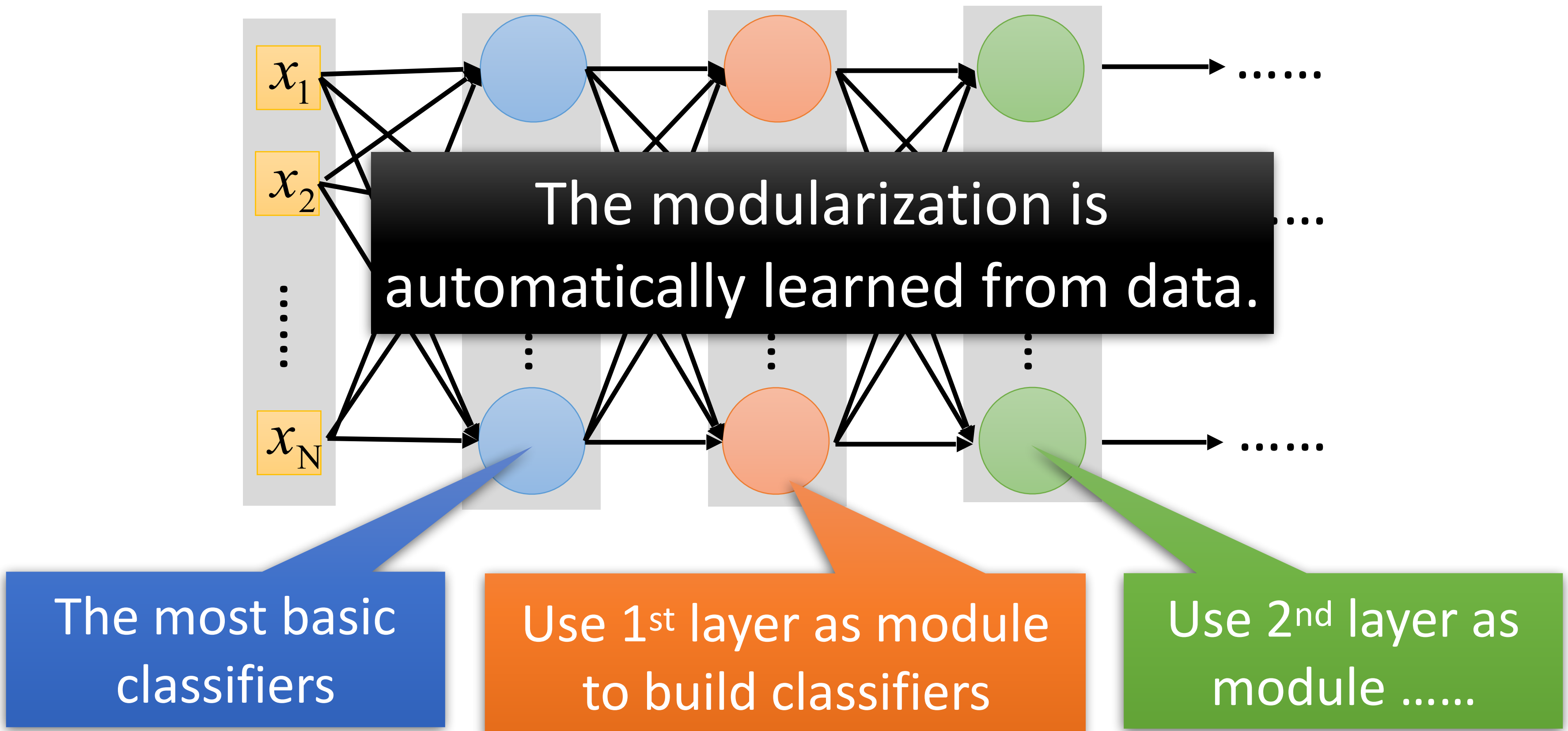
Why Deep?

- Deep → Modularization



Why Deep?

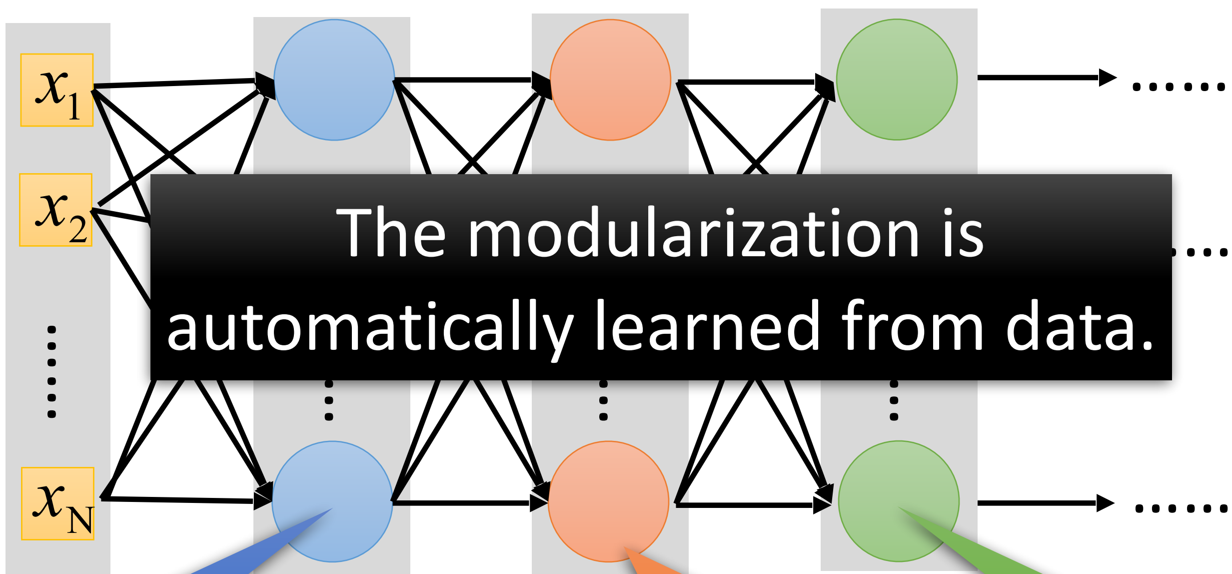
- Deep → Modularization → Less training data?



Why Deep?

Deep Learning also works on small data set like TIMIT.

- Deep → Modularization → Less training data?



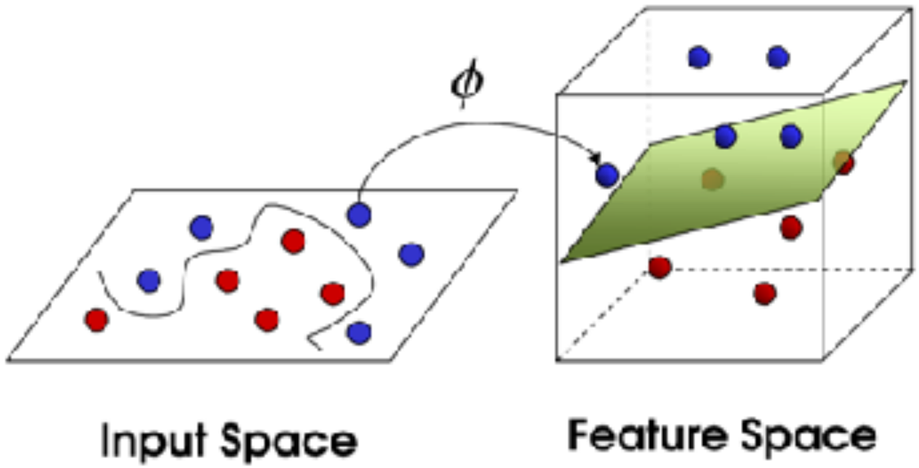
The modularization is automatically learned from data.

The most basic classifiers

Use 1st layer as module to build classifiers

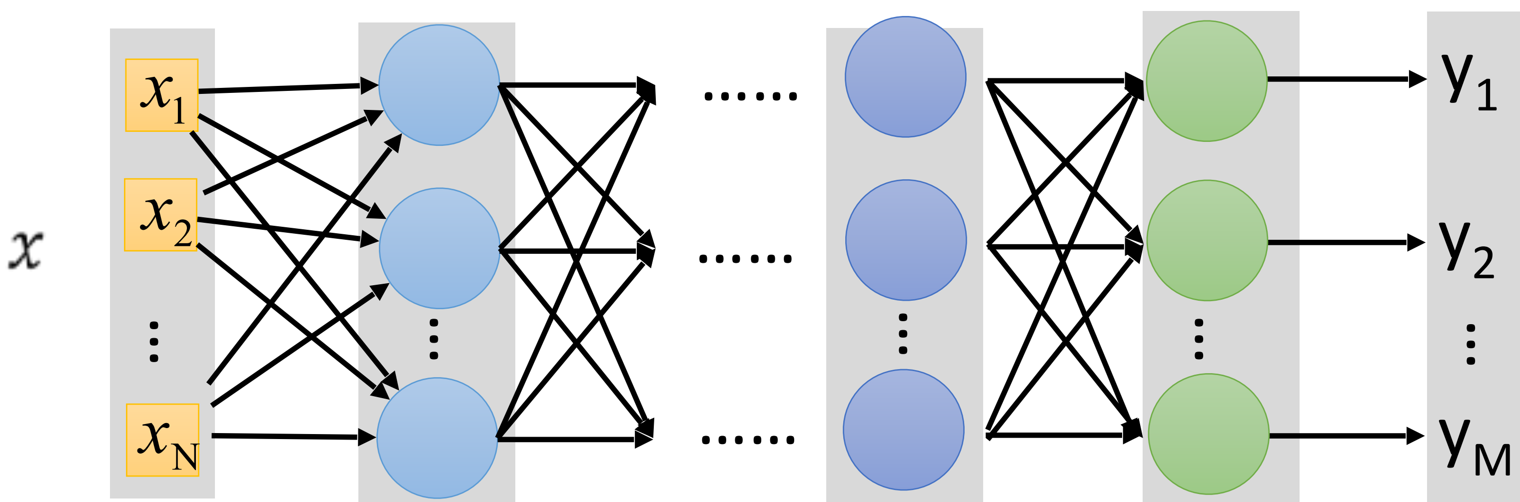
Use 2nd layer as module ……

[SVM](#)



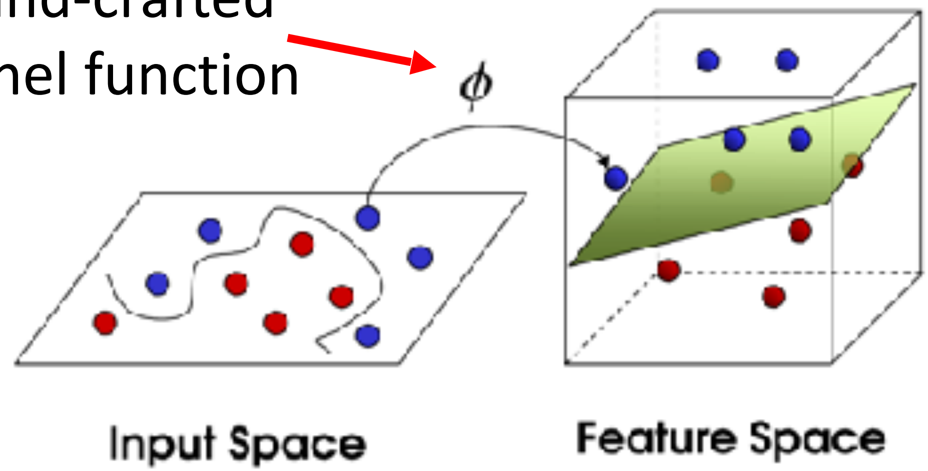
Source of image: http://www.gipsa-lab.grenoble-inp.fr/transfert/seminaire/455_Kadri2013Gipsa-lab.pdf

[Deep Learning](#)



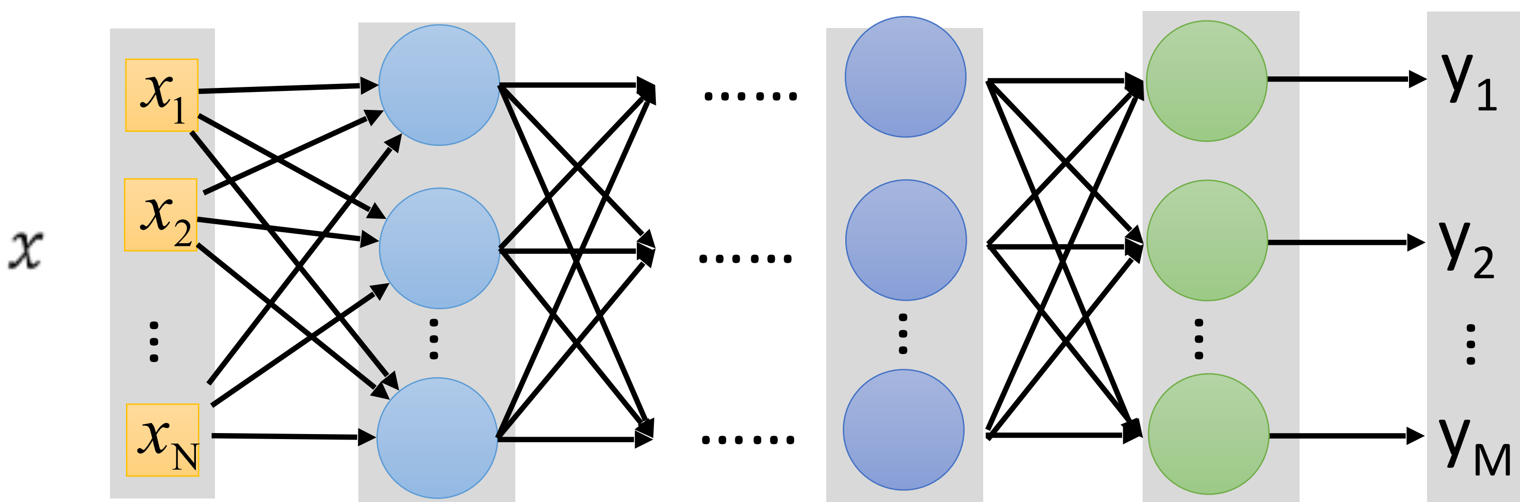
SVM

Hand-crafted kernel function

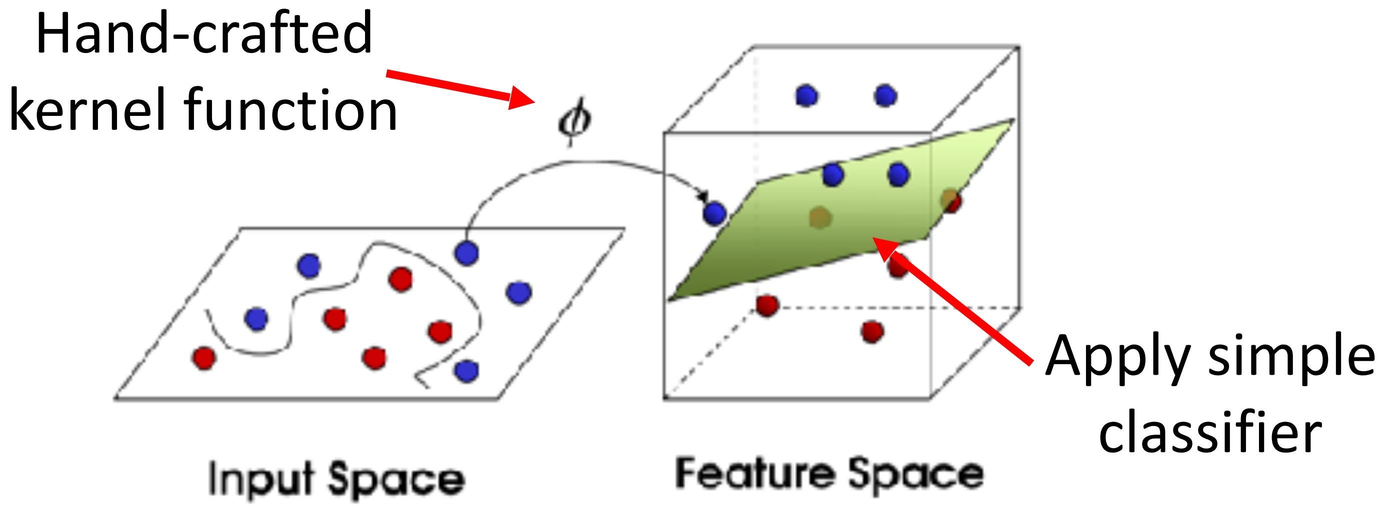


Source of image: http://www.gipsa-lab.grenoble-inp.fr/transfert/seminaire/455_Kadri2013Gipsa-lab.pdf

Deep Learning

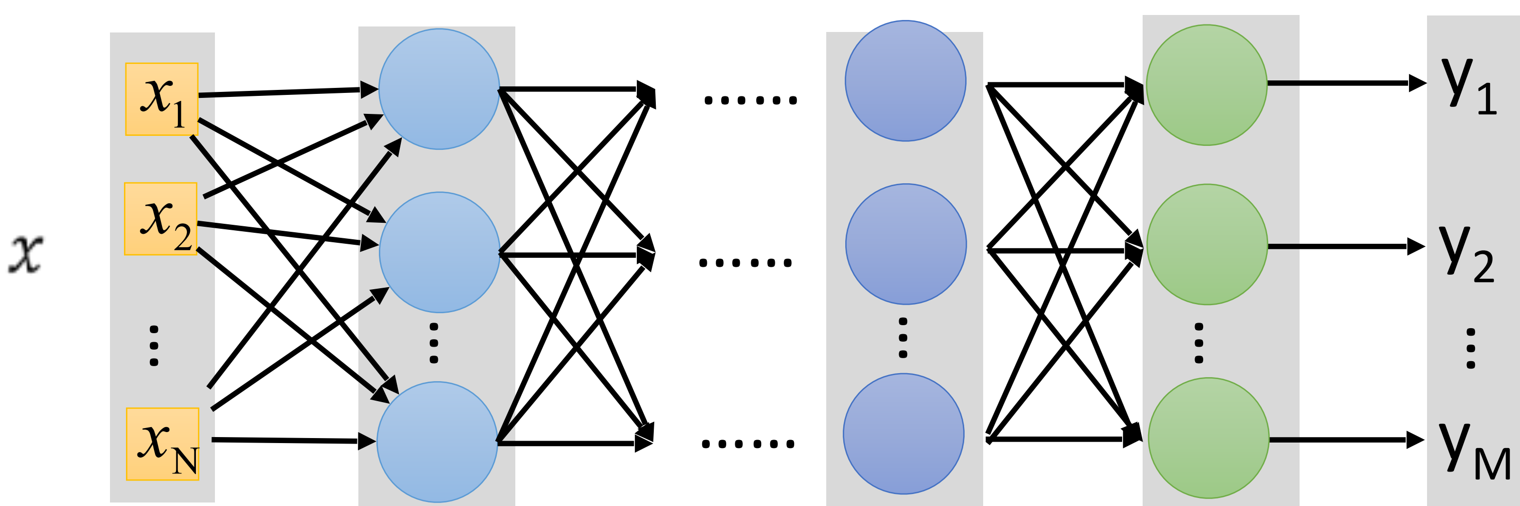


SVM

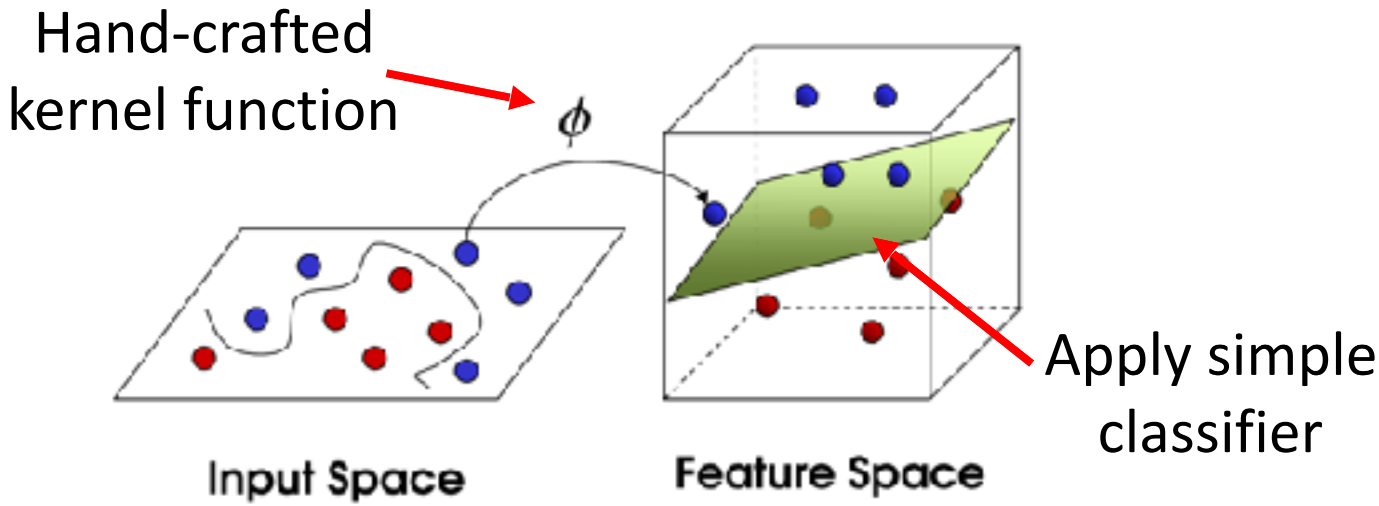


Source of image: http://www.gipsa-lab.grenoble-inp.fr/
transfert/seminaire/455_Kadri2013Gipsa-lab.pdf

Deep Learning

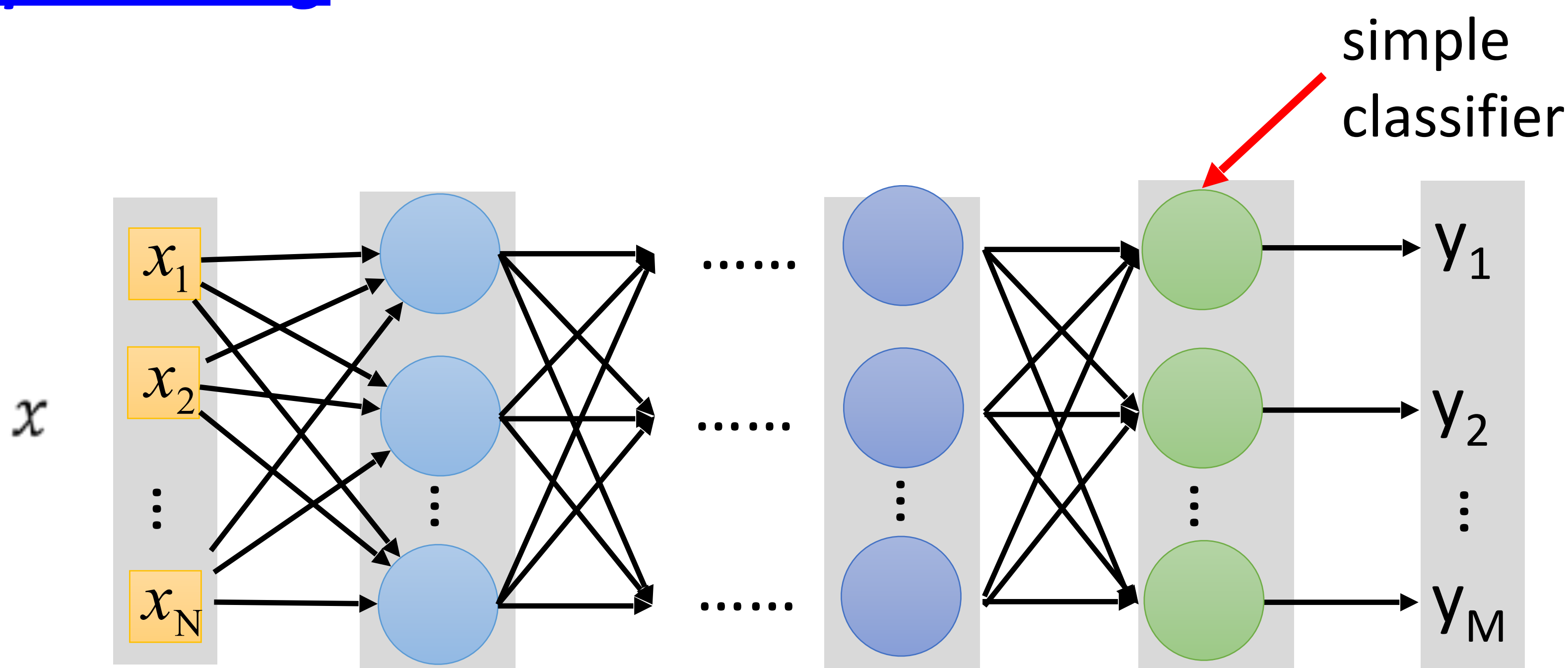


SVM

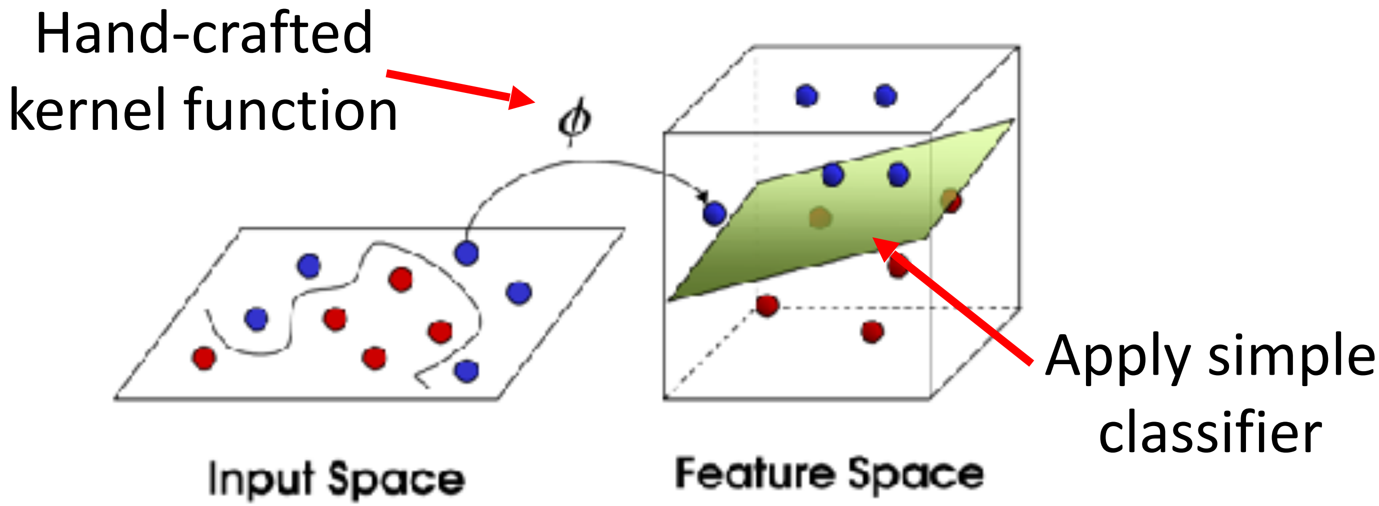


Source of image: http://www.gipsa-lab.grenoble-inp.fr/transfert/seminaire/455_Kadri2013Gipsa-lab.pdf

Deep Learning

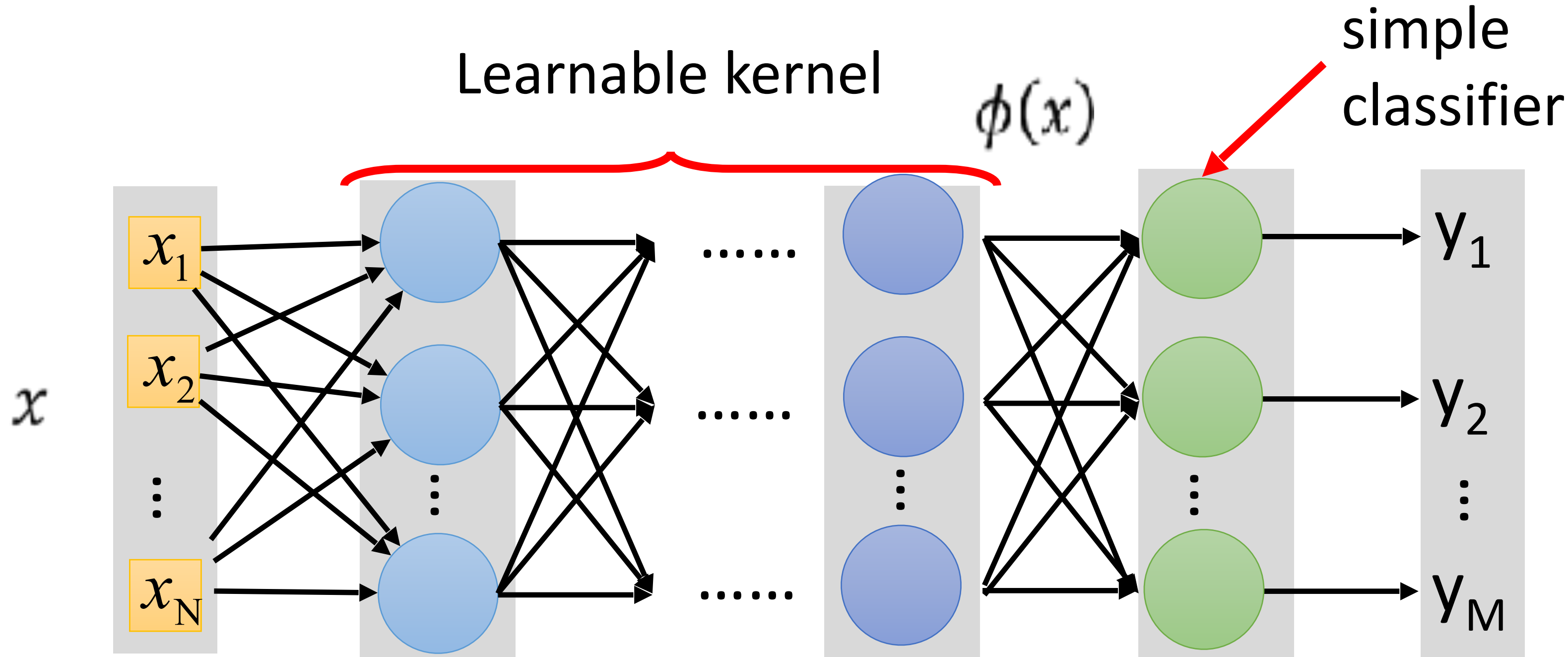


SVM

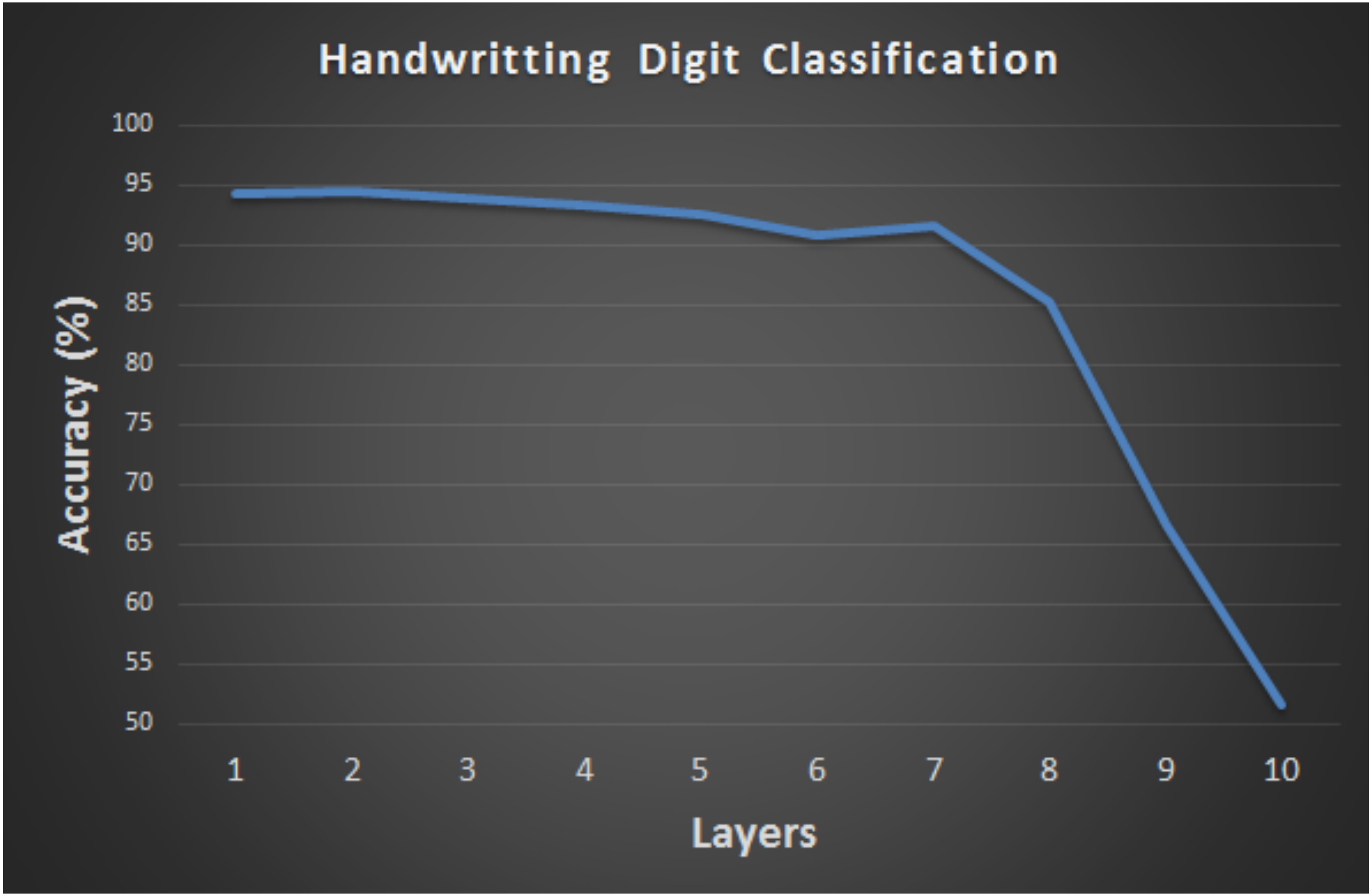


Source of image: http://www.gipsa-lab.grenoble-inp.fr/transfert/seminaire/455_Kadri2013Gipsa-lab.pdf

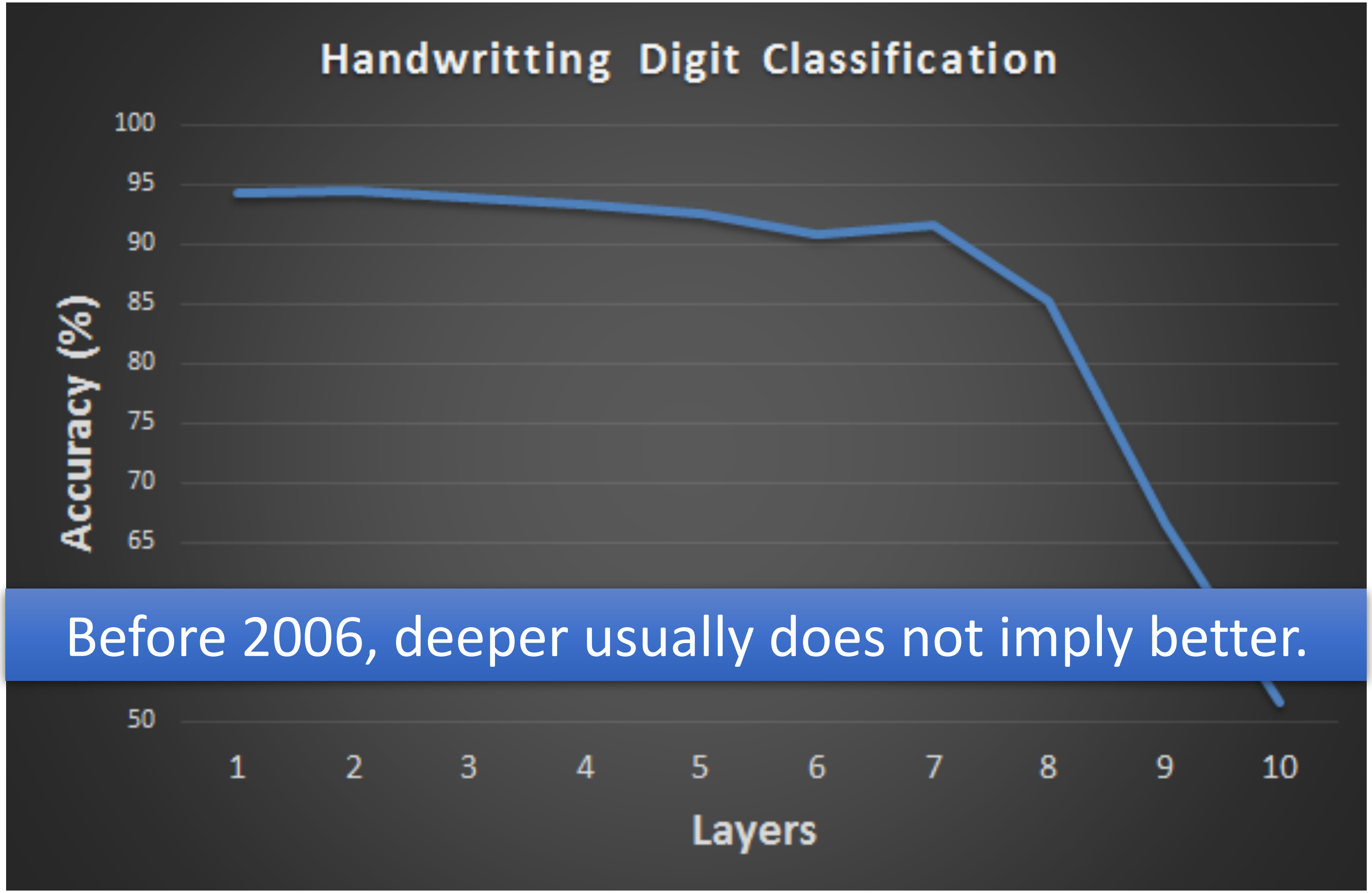
Deep Learning



Hard to get the power of Deep …



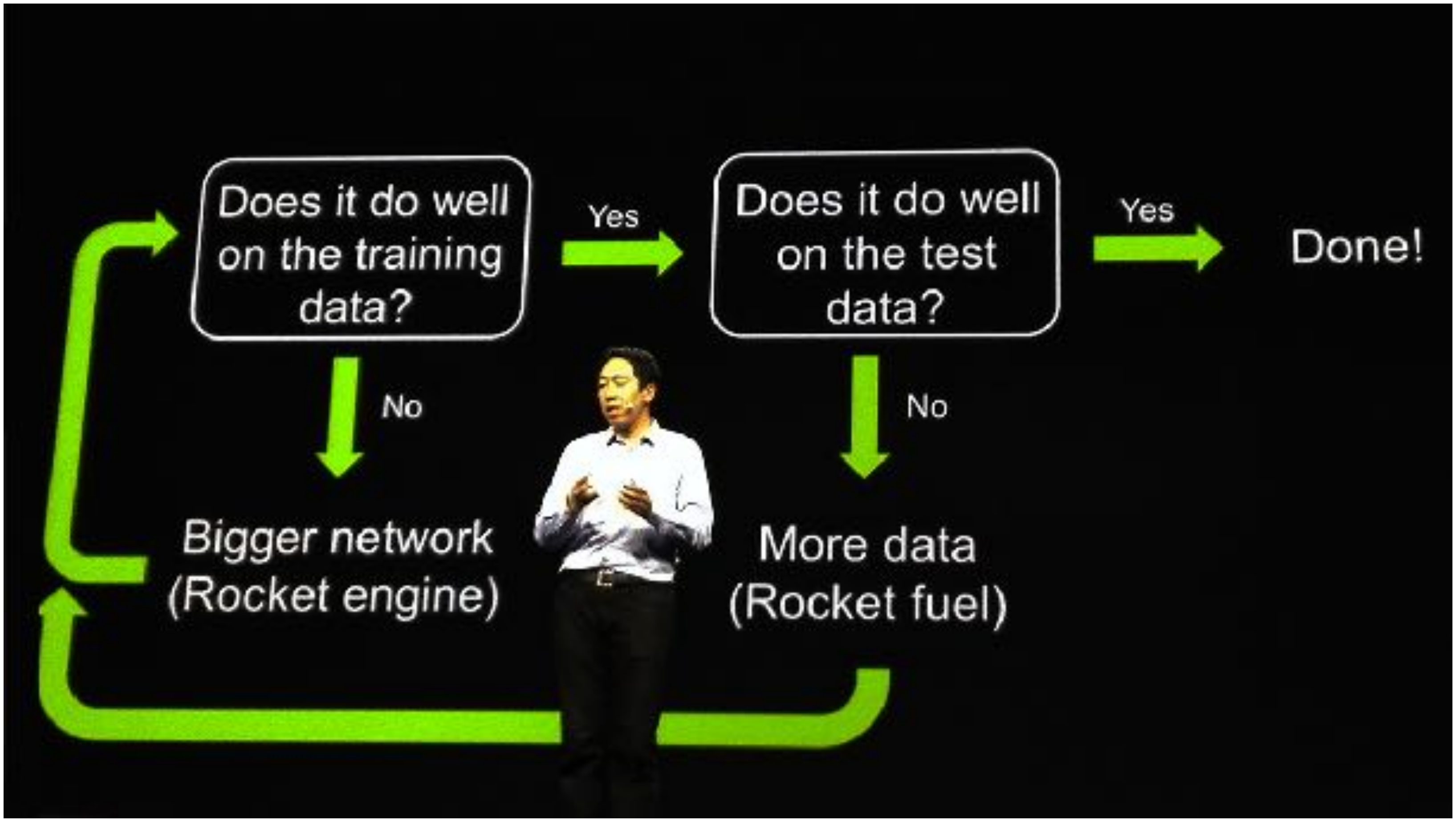
Hard to get the power of Deep …



Before 2006, deeper usually does not imply better.

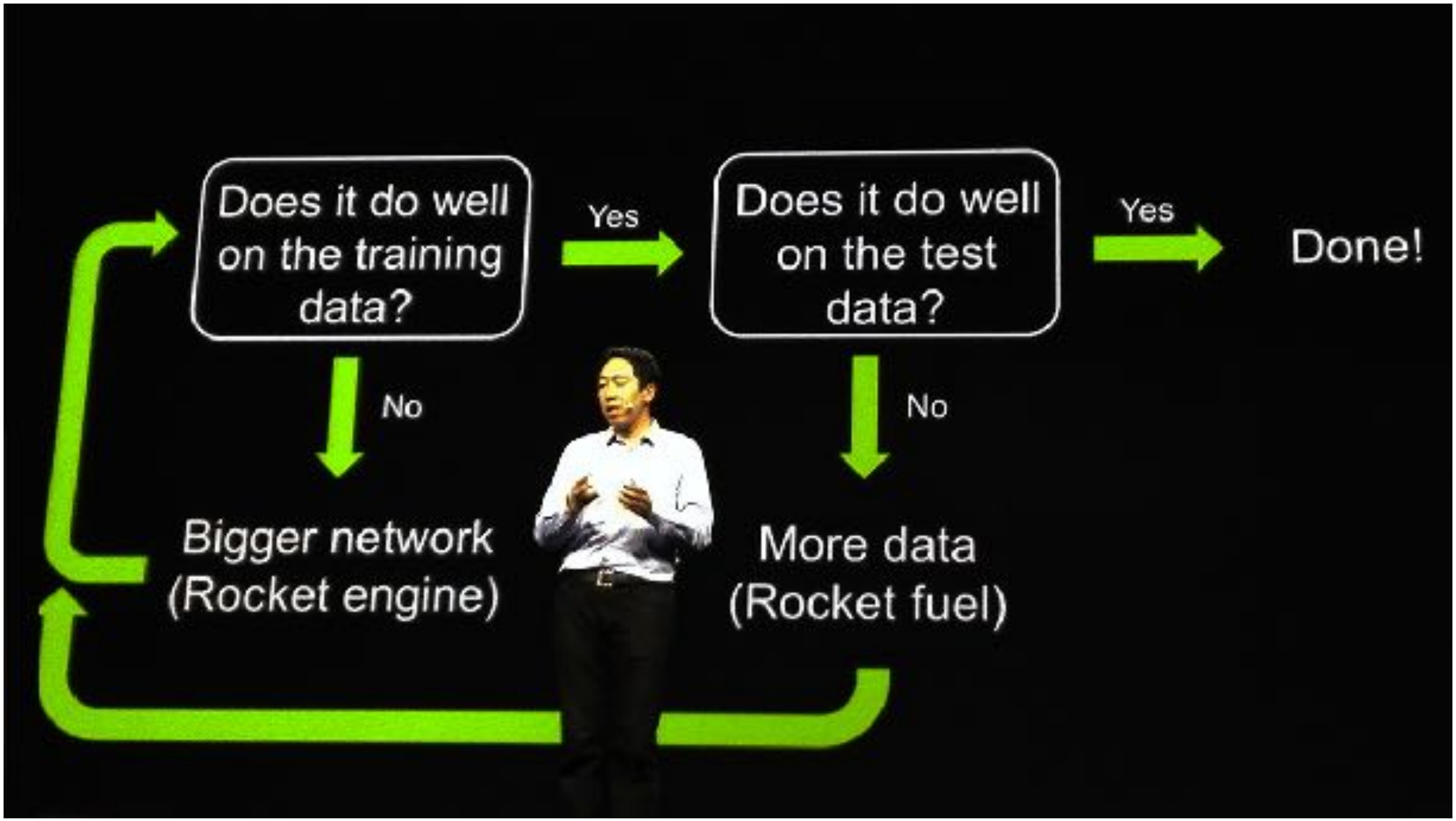
Part III:
Tips for Training DNN

Recipe for Learning



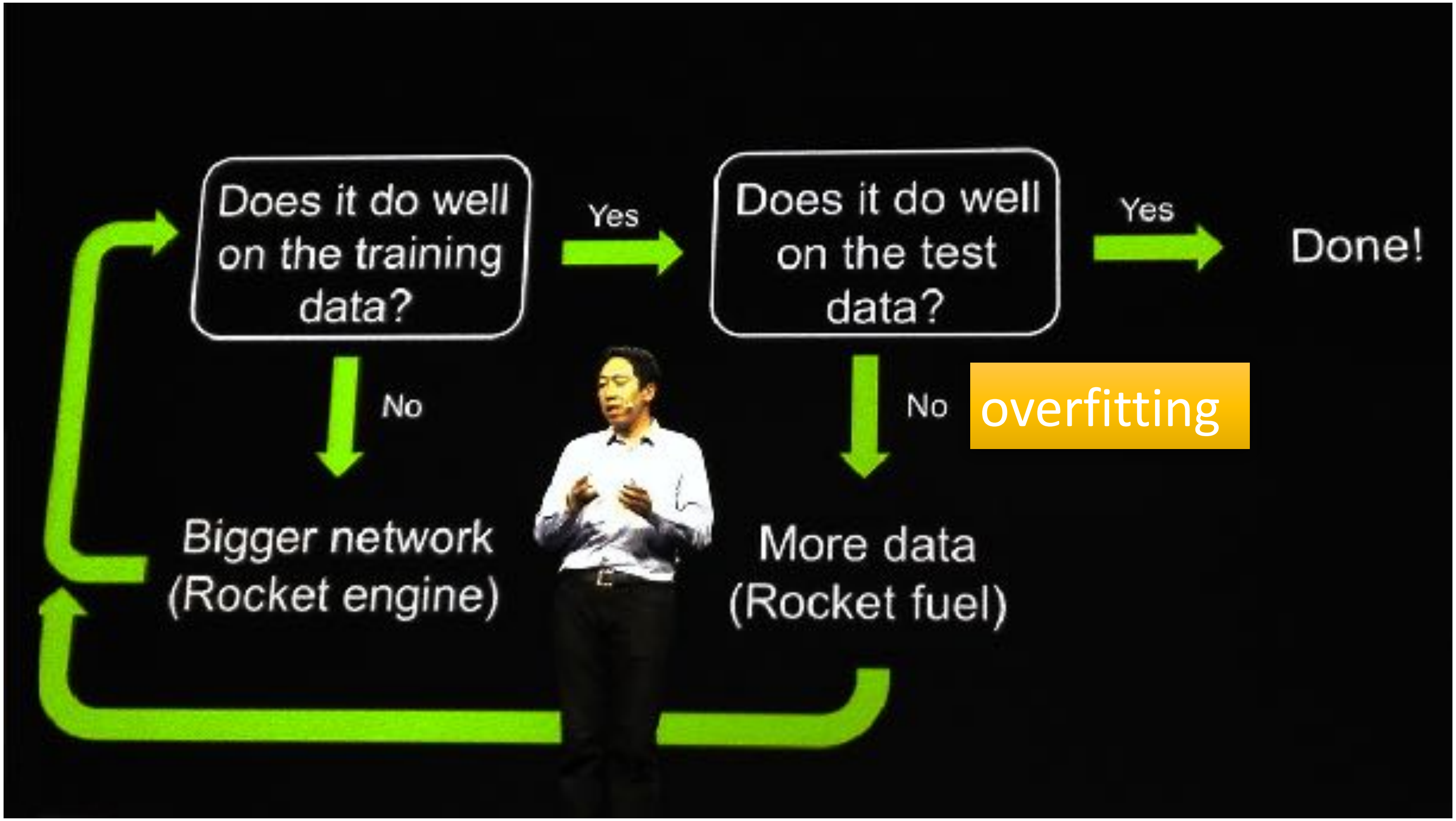
http://www.gizmodo.com.au/2015/04/the-basic-recipe-for-machine-learning-explained-in-a-single-powerpoint-slide/

Recipe for Learning



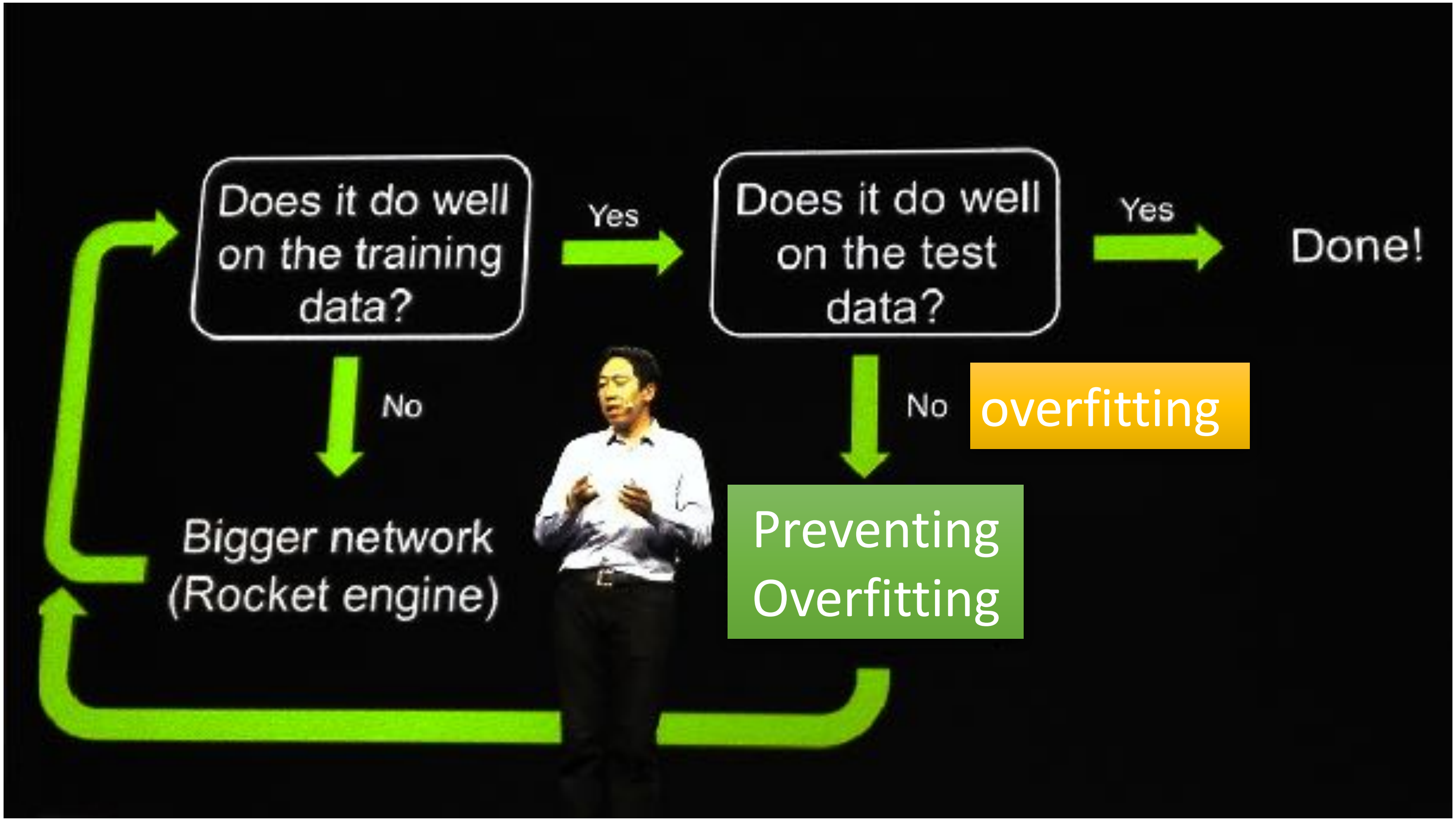
http://www.gizmodo.com.au/2015/04/the-basic-recipe-for-machine-learning-explained-in-a-single-powerpoint-slide/

Recipe for Learning



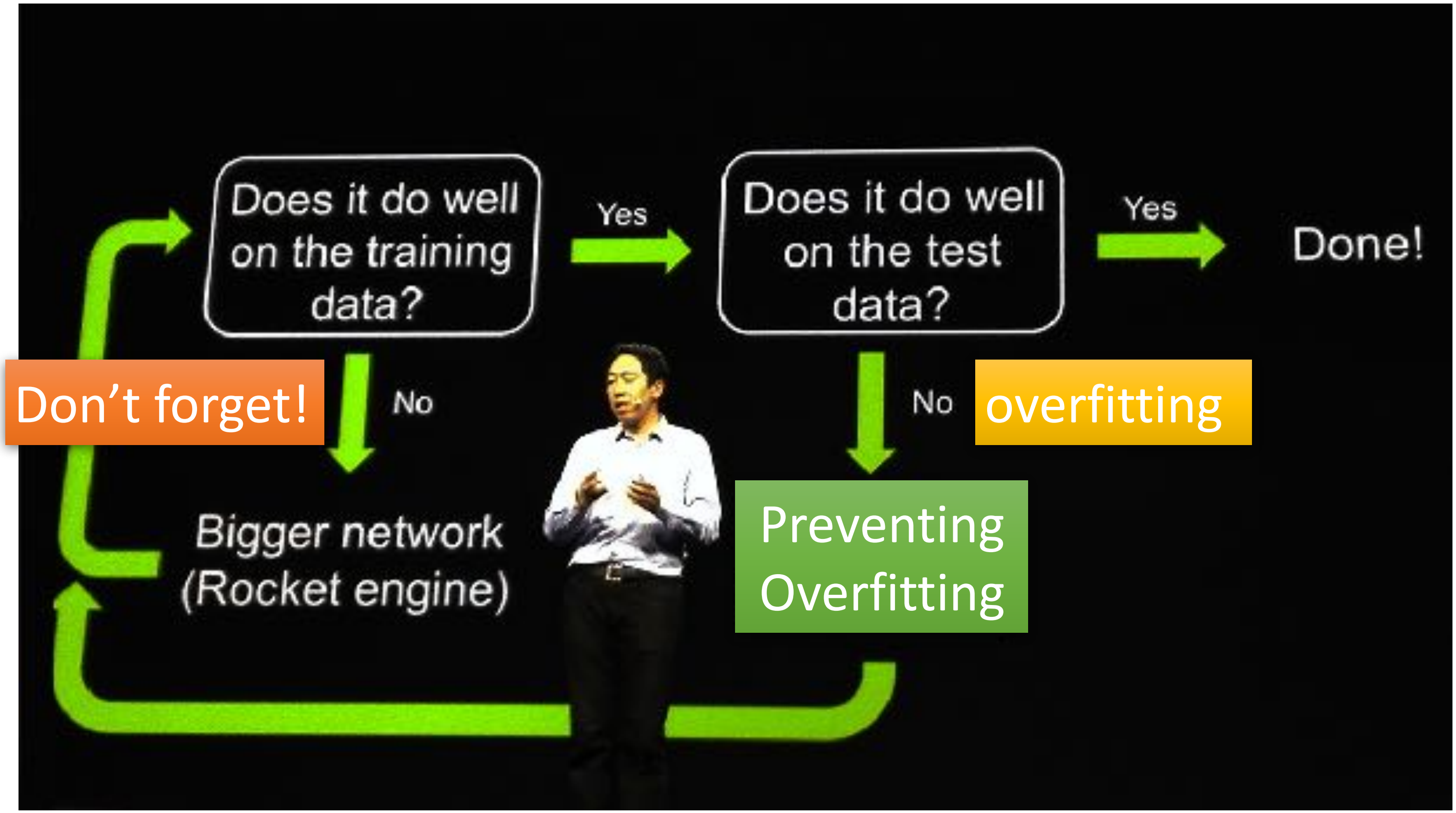
http://www.gizmodo.com.au/2015/04/the-basic-recipe-for-machine-learning-explained-in-a-single-powerpoint-slide/

Recipe for Learning



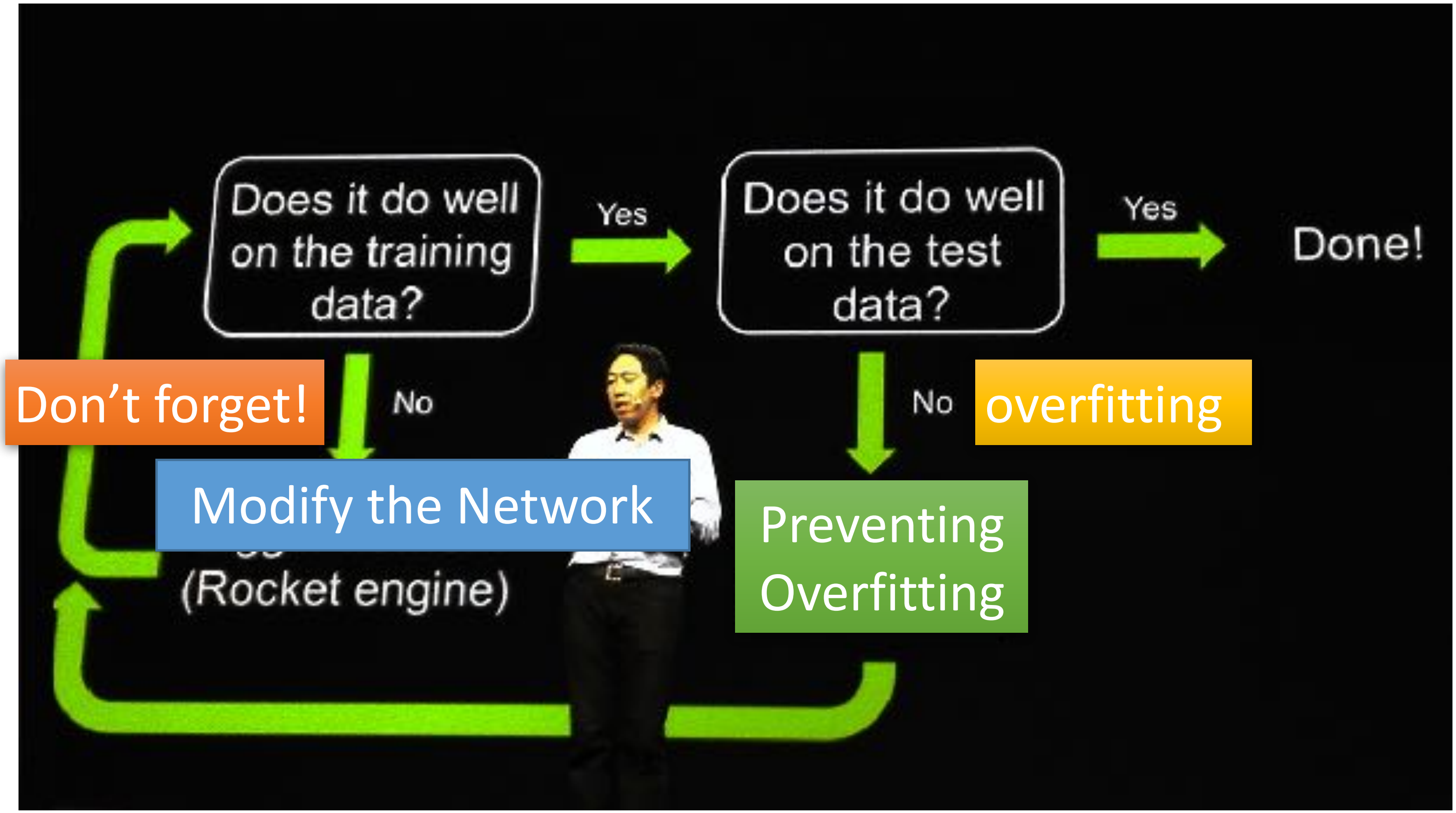
http://www.gizmodo.com.au/2015/04/the-basic-recipe-for-machine-learning-explained-in-a-single-powerpoint-slide/

Recipe for Learning



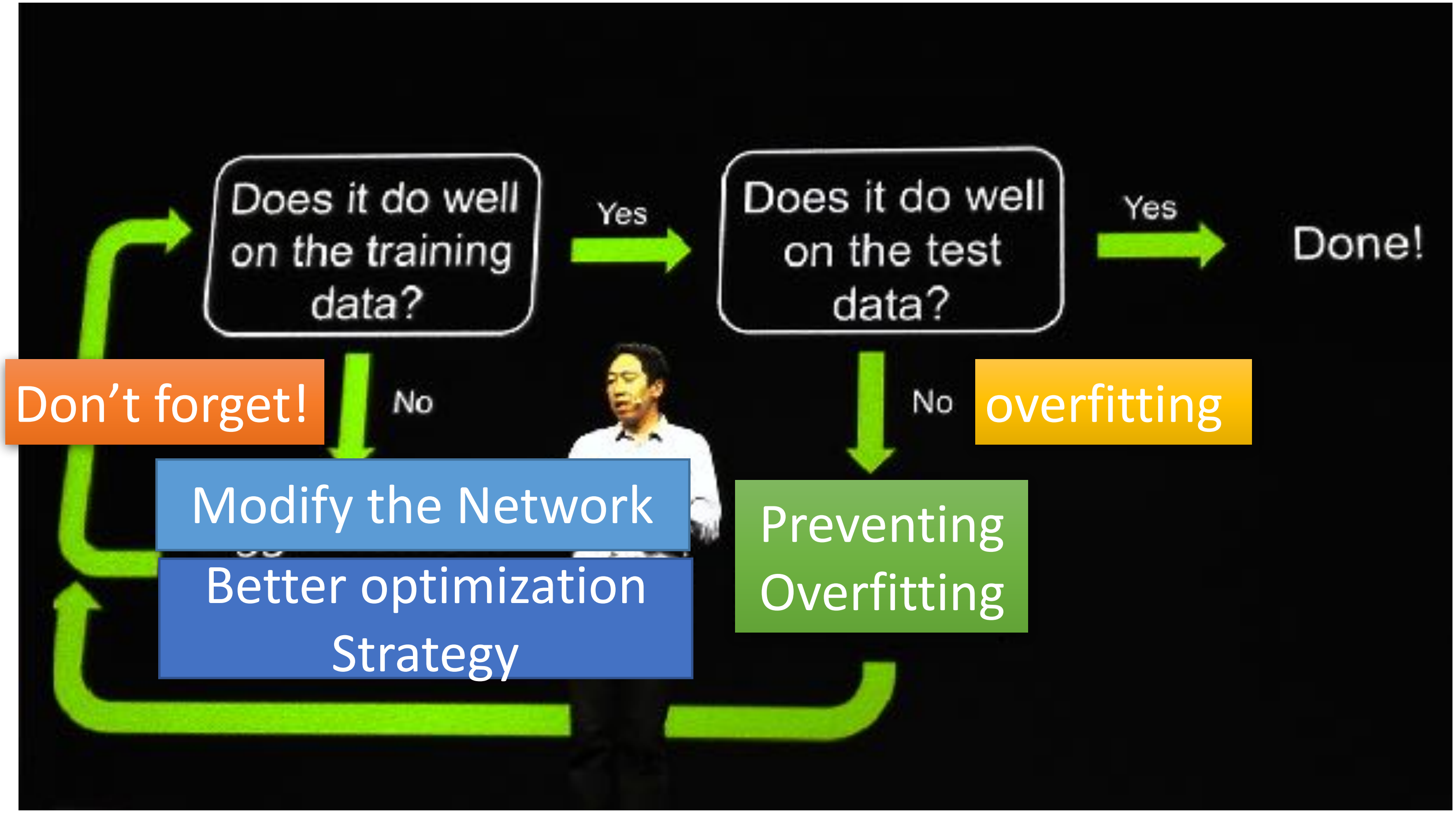
http://www.gizmodo.com.au/2015/04/the-basic-recipe-for-machine-learning-explained-in-a-single-powerpoint-slide/

Recipe for Learning



http://www.gizmodo.com.au/2015/04/the-basic-recipe-for-machine-learning-explained-in-a-single-powerpoint-slide/

Recipe for Learning



http://www.gizmodo.com.au/2015/04/the-basic-recipe-for-machine-learning-explained-in-a-single-powerpoint-slide/

Recipe for Learning

- Modify the Network
  - New activation functions, for example, ReLU or Maxout
- Better optimization Strategy
  - Adaptive learning rates
- Prevent Overfitting
  - Dropout

Recipe for Learning

- Modify the Network
  - New activation functions, for example, ReLU or Maxout
- Better optimization Strategy
  - Adaptive learning rates
- Prevent Overfitting
  - Dropout

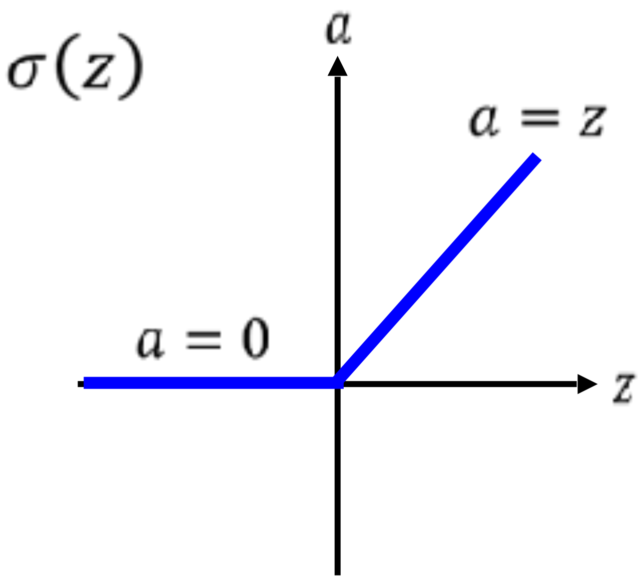
Only use this approach when you already obtained good results on the training data.

Part III:
Tips for Training DNN
New Activation Function

ReLU

- Rectified Linear Unit (ReLU)

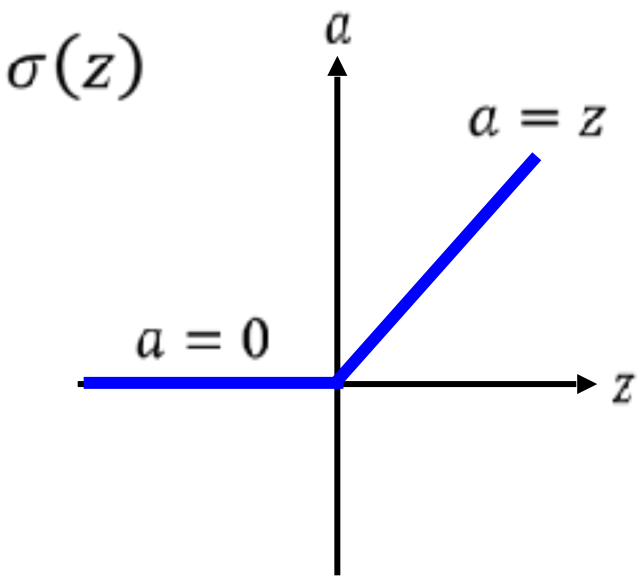
Reason:



[Xavier Glorot, AISTATS'11]
[Andrew L. Maas, ICML'13]
[Kaiming He, arXiv'15]

ReLU

- Rectified Linear Unit (ReLU)



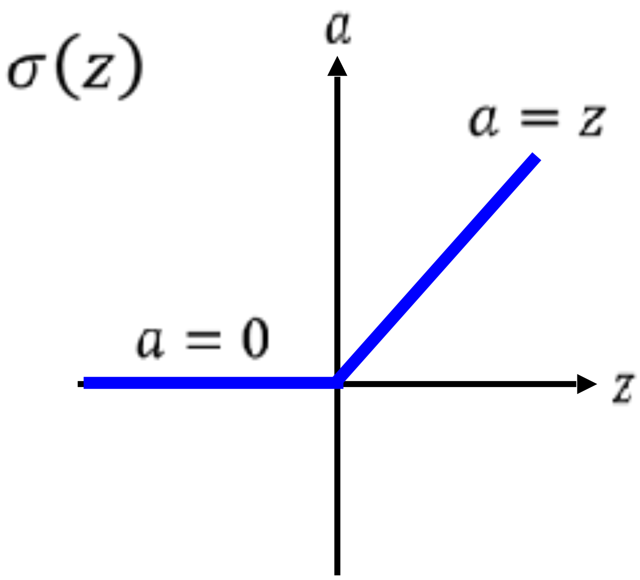
[Xavier Glorot, AISTATS'11]
[Andrew L. Maas, ICML'13]
[Kaiming He, arXiv'15]

Reason:

1. Fast to compute

ReLU

- Rectified Linear Unit (ReLU)



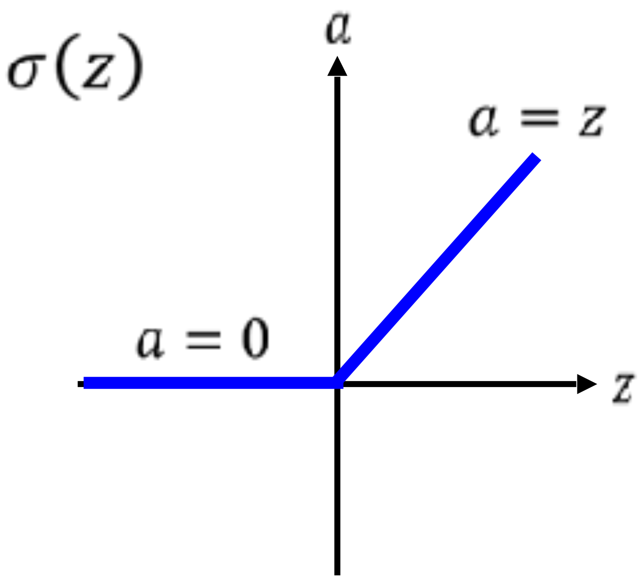
[Xavier Glorot, AISTATS'11]
[Andrew L. Maas, ICML'13]
[Kaiming He, arXiv'15]

Reason:

1. Fast to compute
2. Biological reason

ReLU

- Rectified Linear Unit (ReLU)



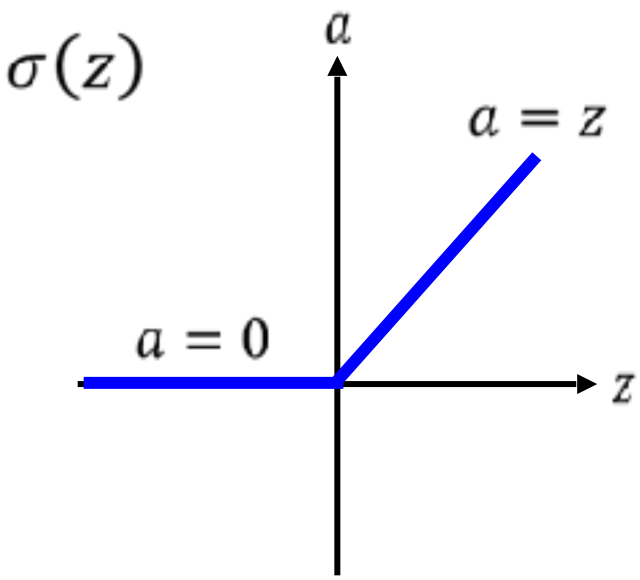
[Xavier Glorot, AISTATS'11]
[Andrew L. Maas, ICML'13]
[Kaiming He, arXiv'15]

Reason:

1. Fast to compute
2. Biological reason
3. Infinite sigmoid with different biases

ReLU

- Rectified Linear Unit (ReLU)



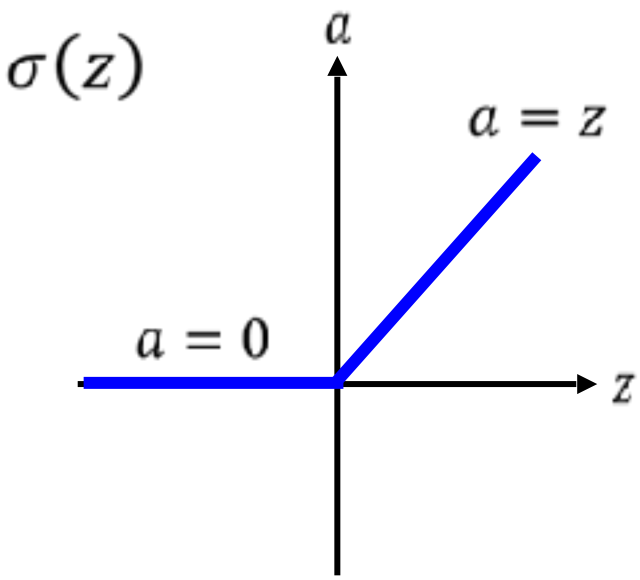
[Xavier Glorot, AISTATS’11]
[Andrew L. Maas, ICML’13]
[Kaiming He, arXiv’15]

Reason:

1. Fast to compute
2. Biological reason
3. Infinite sigmoid with different biases
4. Vanishing gradient problem

ReLU

- Rectified Linear Unit (ReLU)

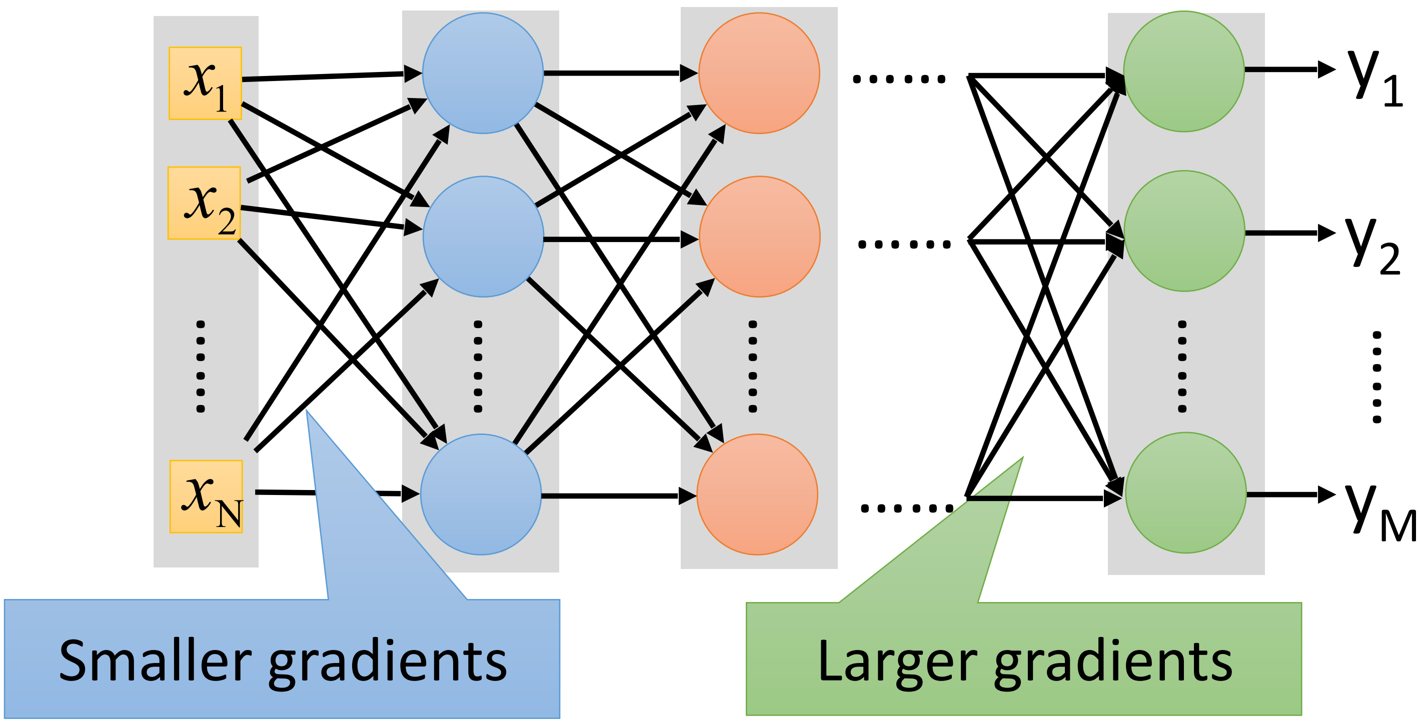


[Xavier Glorot, AISTATS'11]
[Andrew L. Maas, ICML'13]
[Kaiming He, arXiv'15]

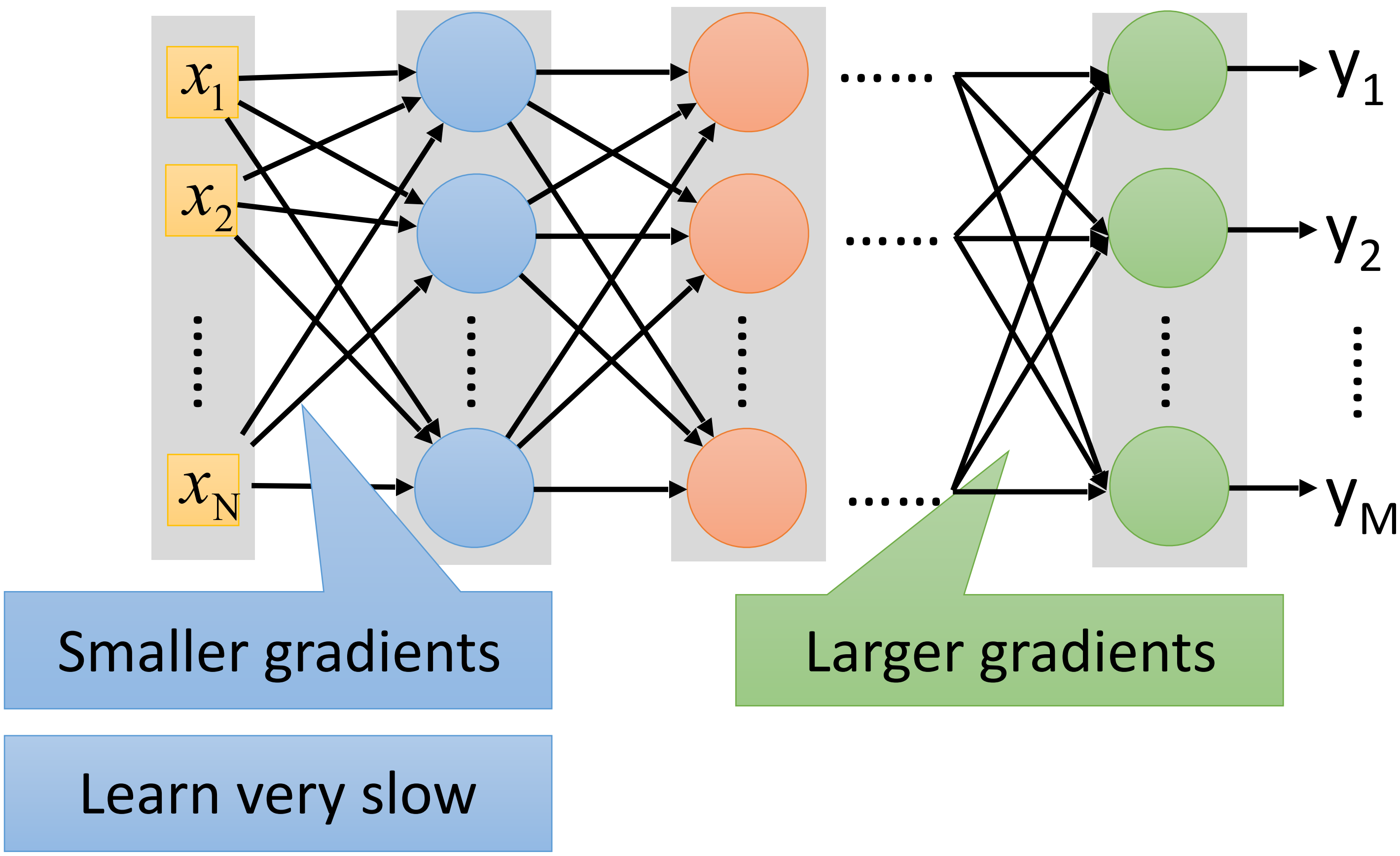
Reason:

1. Fast to compute
2. Biological reason
3. Infinite sigmoid with different biases
4. Vanishing gradient problem

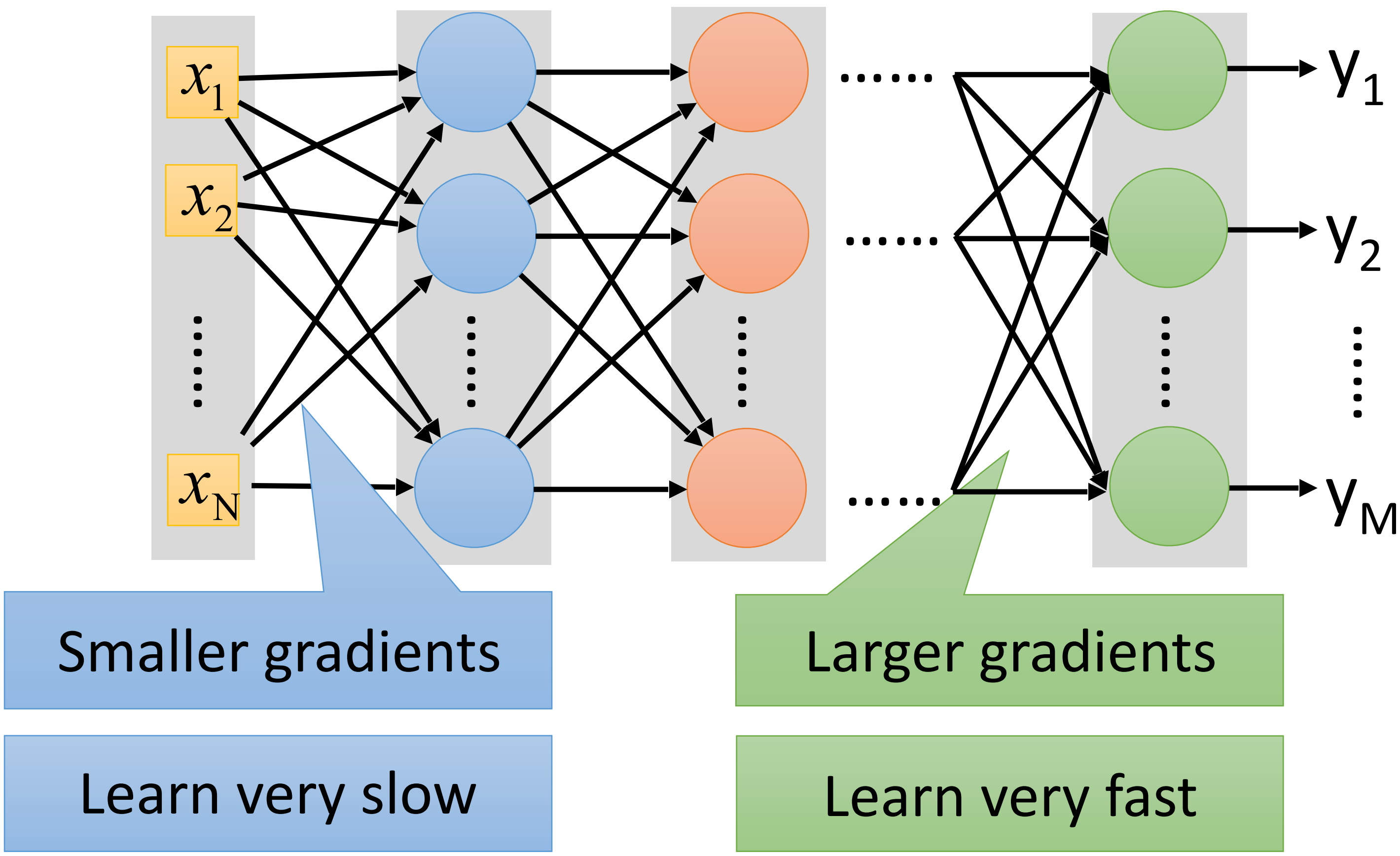
Vanishing Gradient Problem



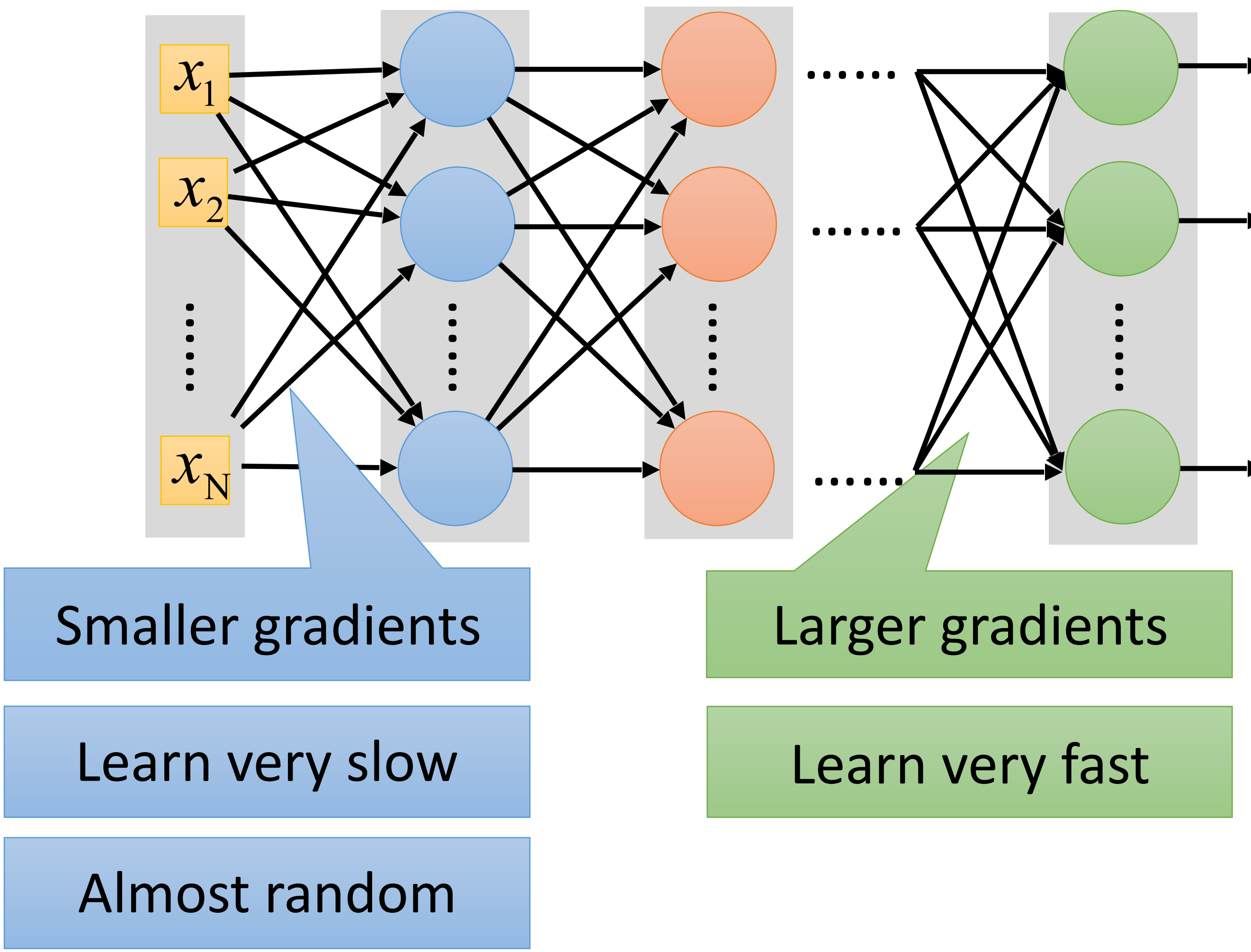
Vanishing Gradient Problem



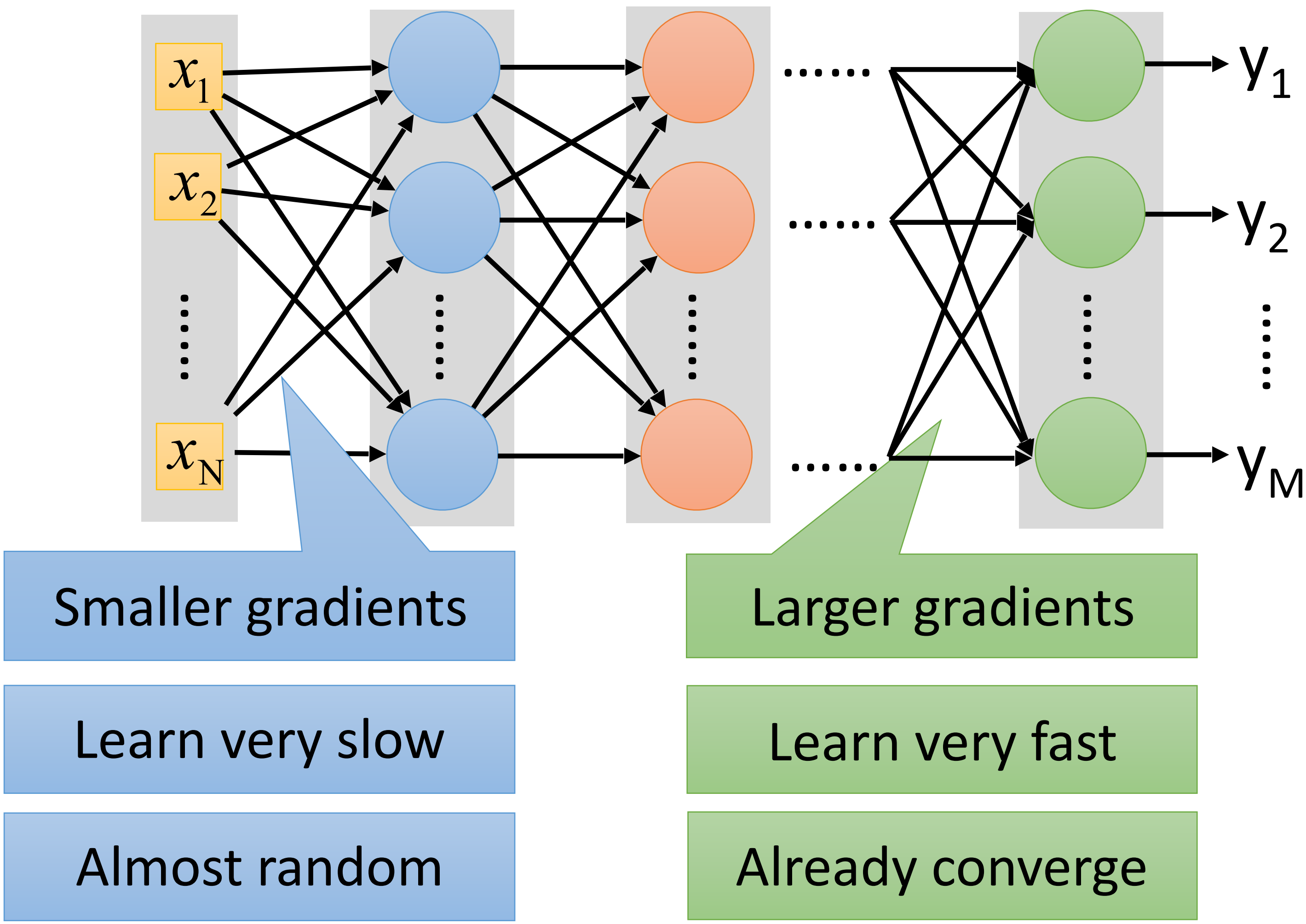
Vanishing Gradient Problem



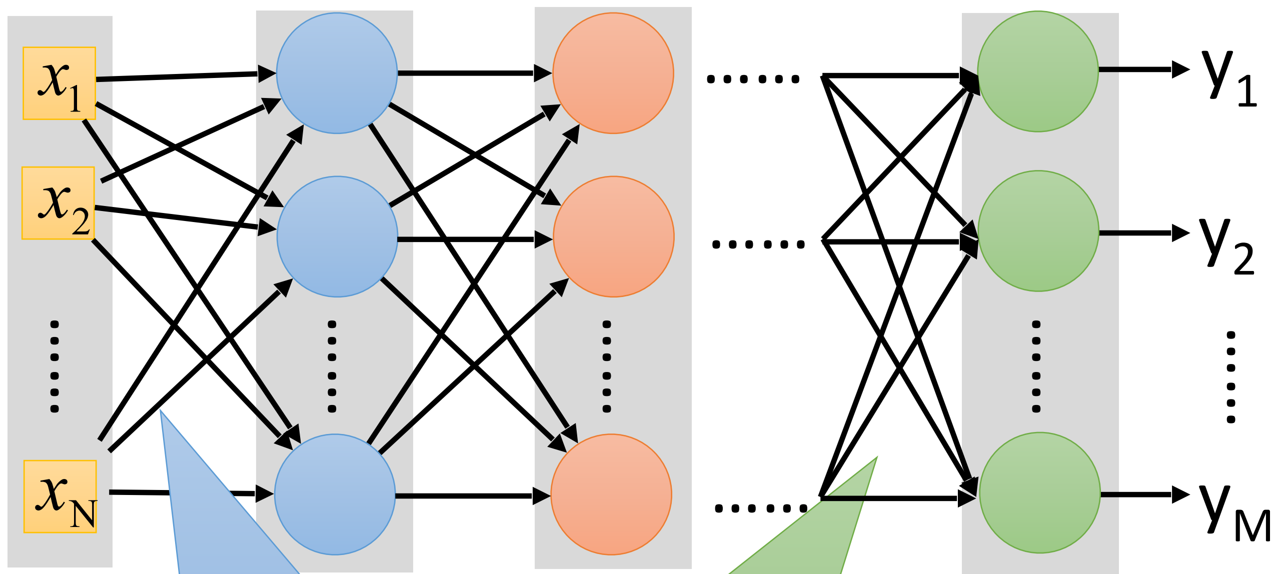
Vanishing Gradient Problem



Vanishing Gradient Problem



Vanishing Gradient Problem



Smaller gradients

Learn very slow

Almost random

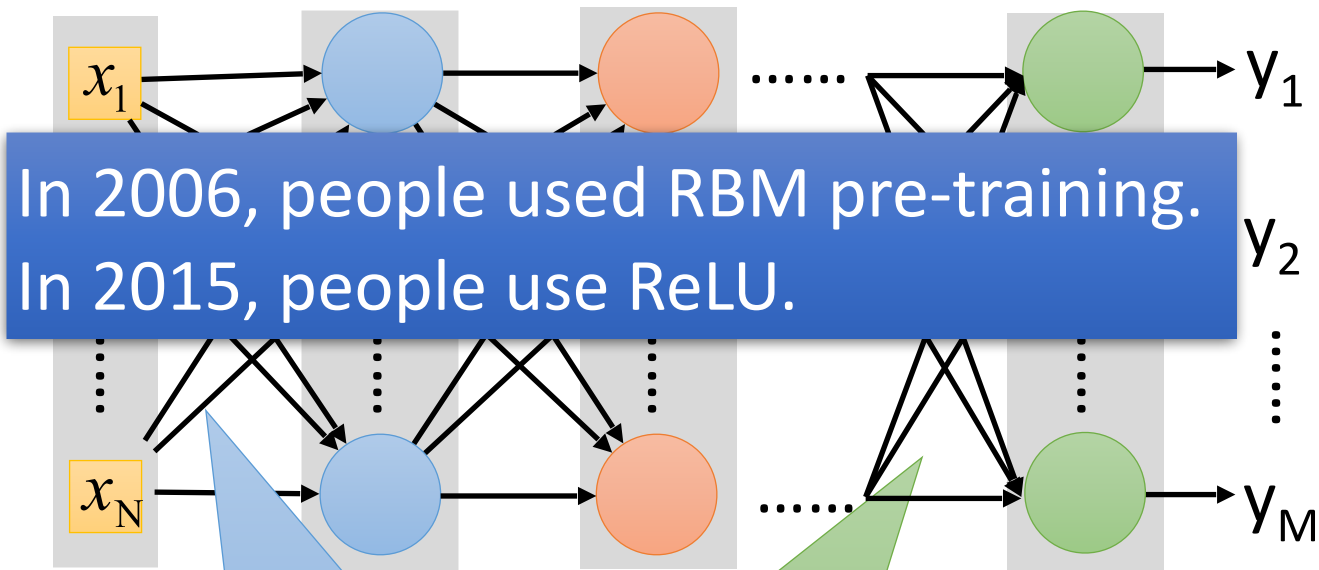
Larger gradients

Learn very fast

Already converge

based on random!?

Vanishing Gradient Problem



Smaller gradients

Learn very slow

Almost random

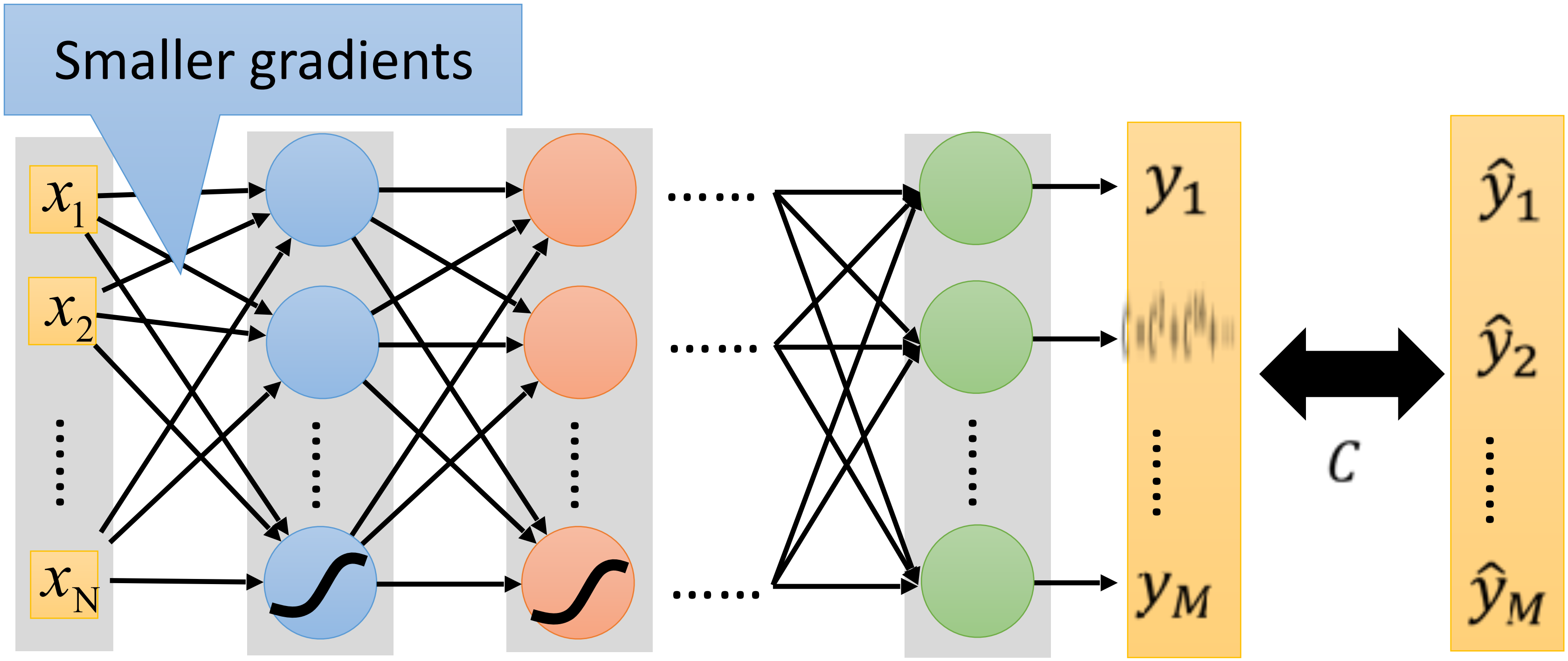
Larger gradients

Learn very fast

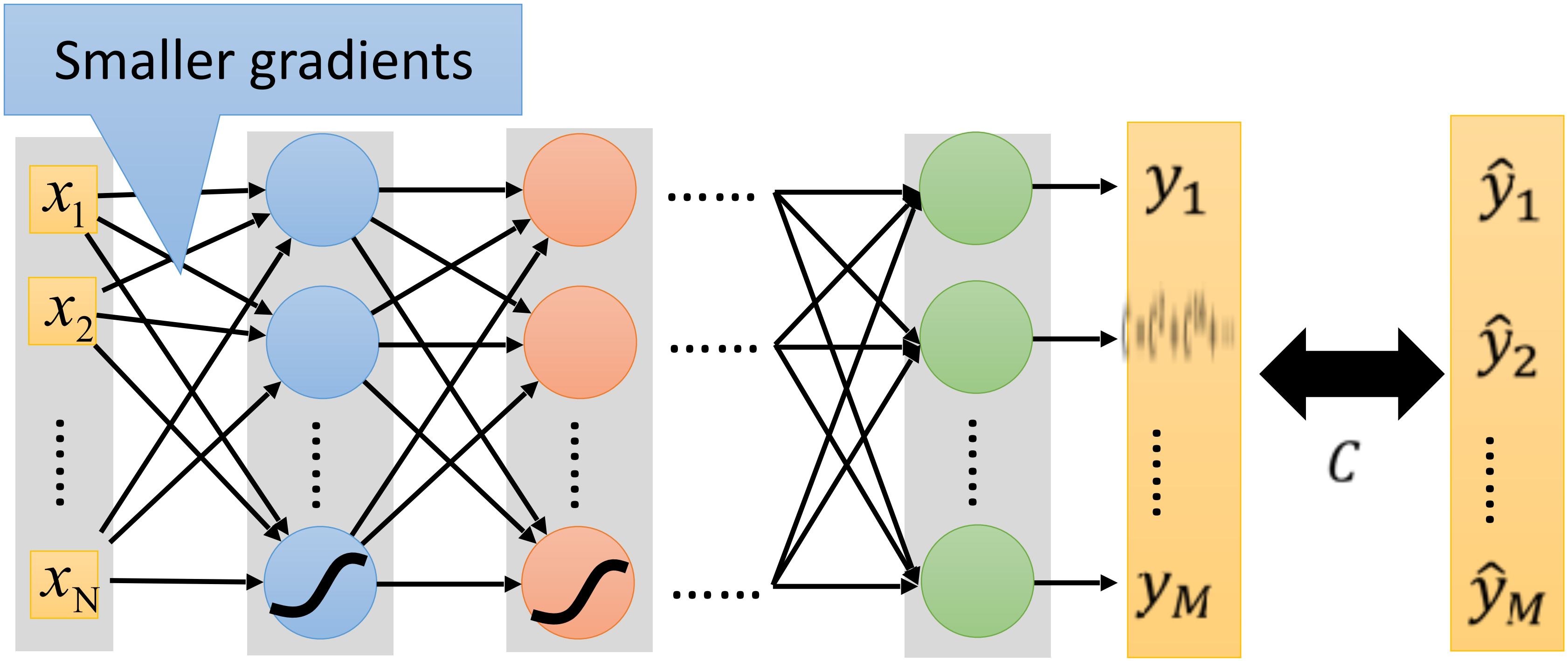
Already converge

based on random!?

Vanishing Gradient Problem

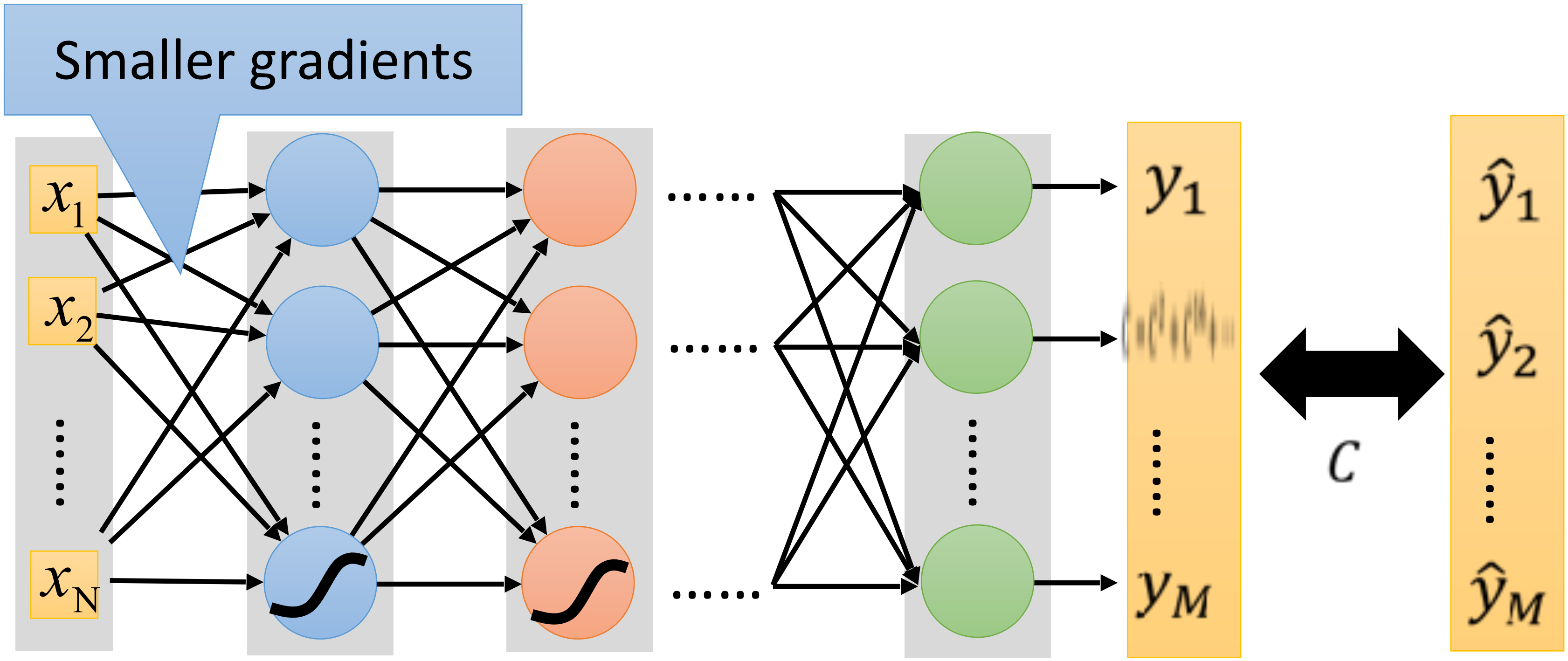


Vanishing Gradient Problem



$$\frac{\partial C}{\partial w} = ?$$

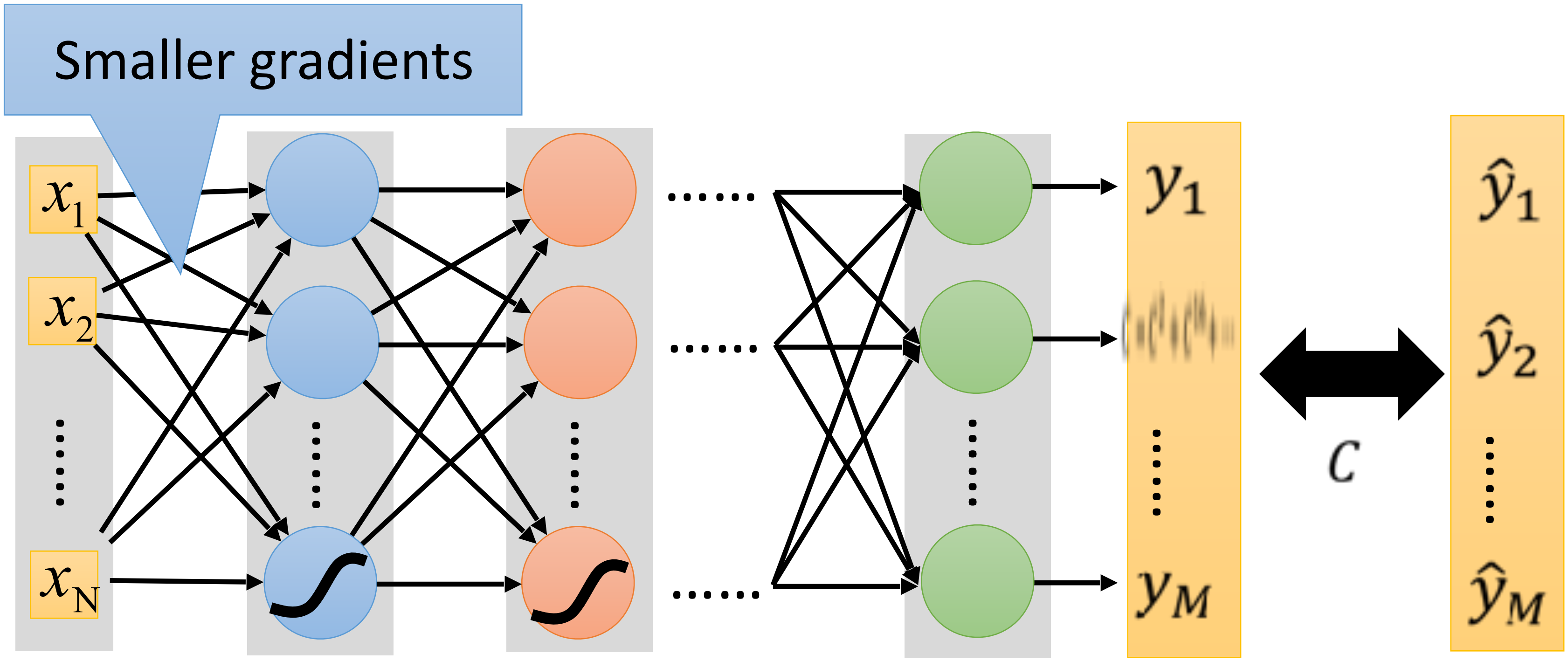
Vanishing Gradient Problem



Intuitive way to compute the gradient …

$$\frac{\partial C}{\partial w} = ?$$

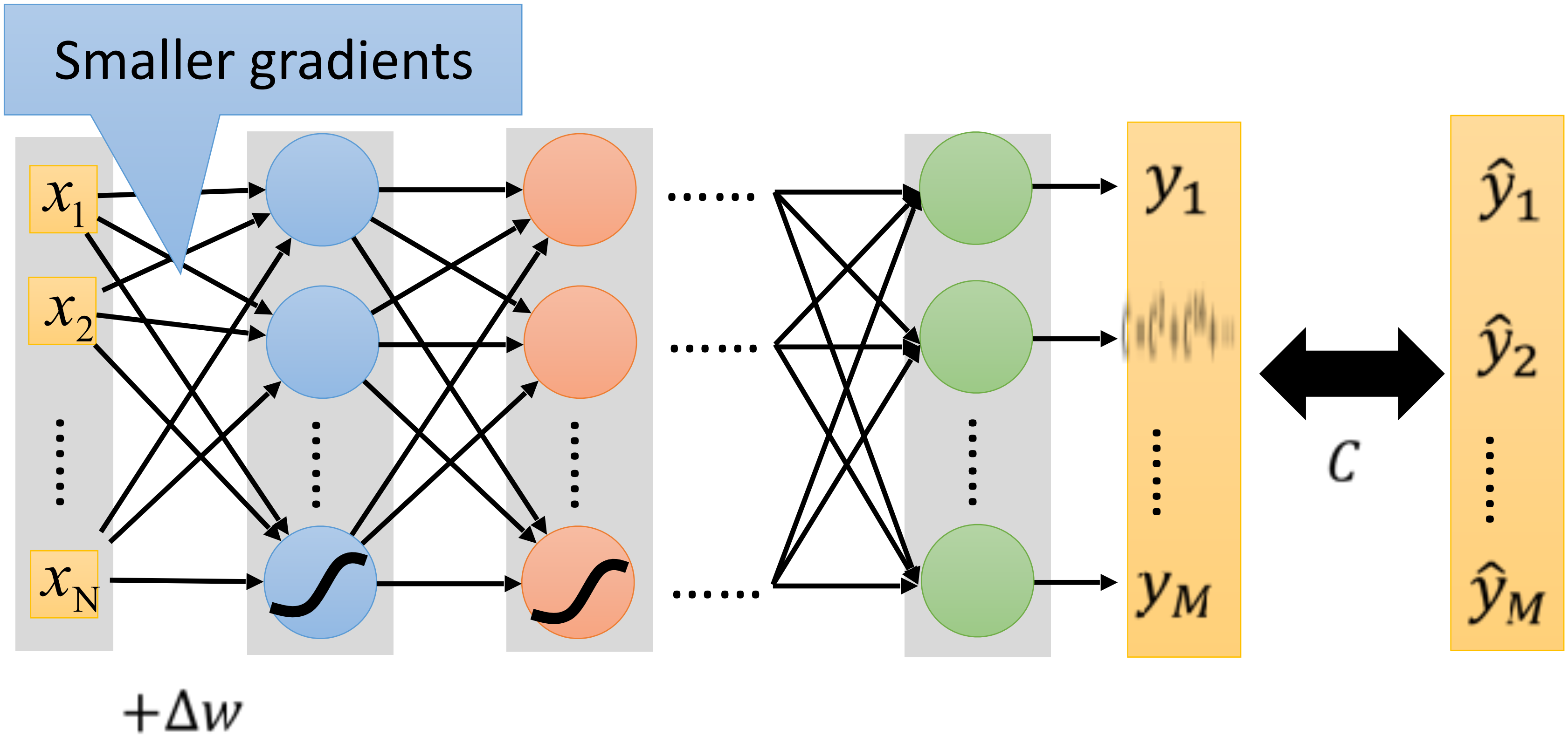
Vanishing Gradient Problem



Intuitive way to compute the gradient …

$$\frac{\partial C}{\partial w} =? \ \frac{\Delta C}{\Delta w}$$

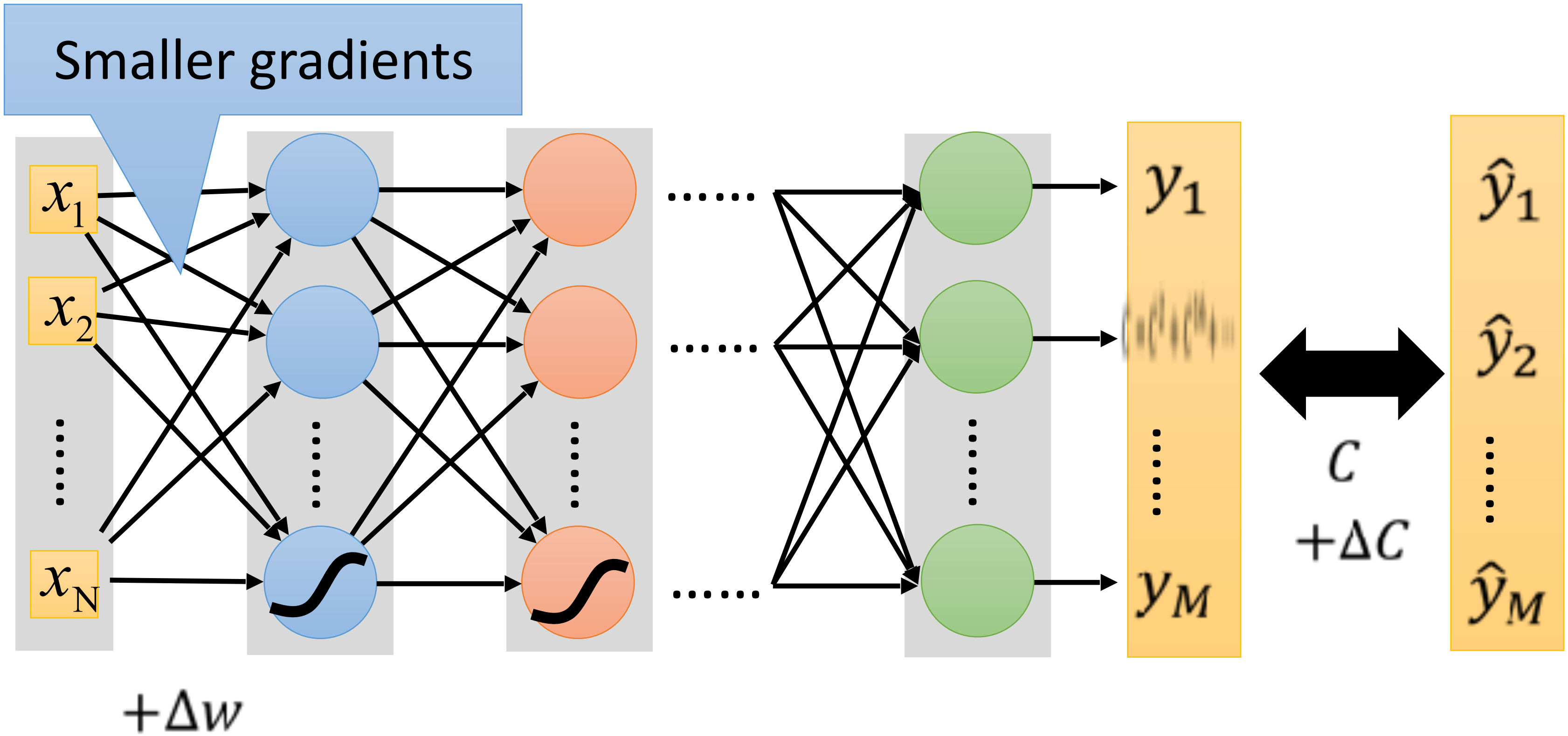
Vanishing Gradient Problem



Intuitive way to compute the gradient …

$$\frac{\partial C}{\partial w} =? \frac{\Delta C}{\Delta w}$$

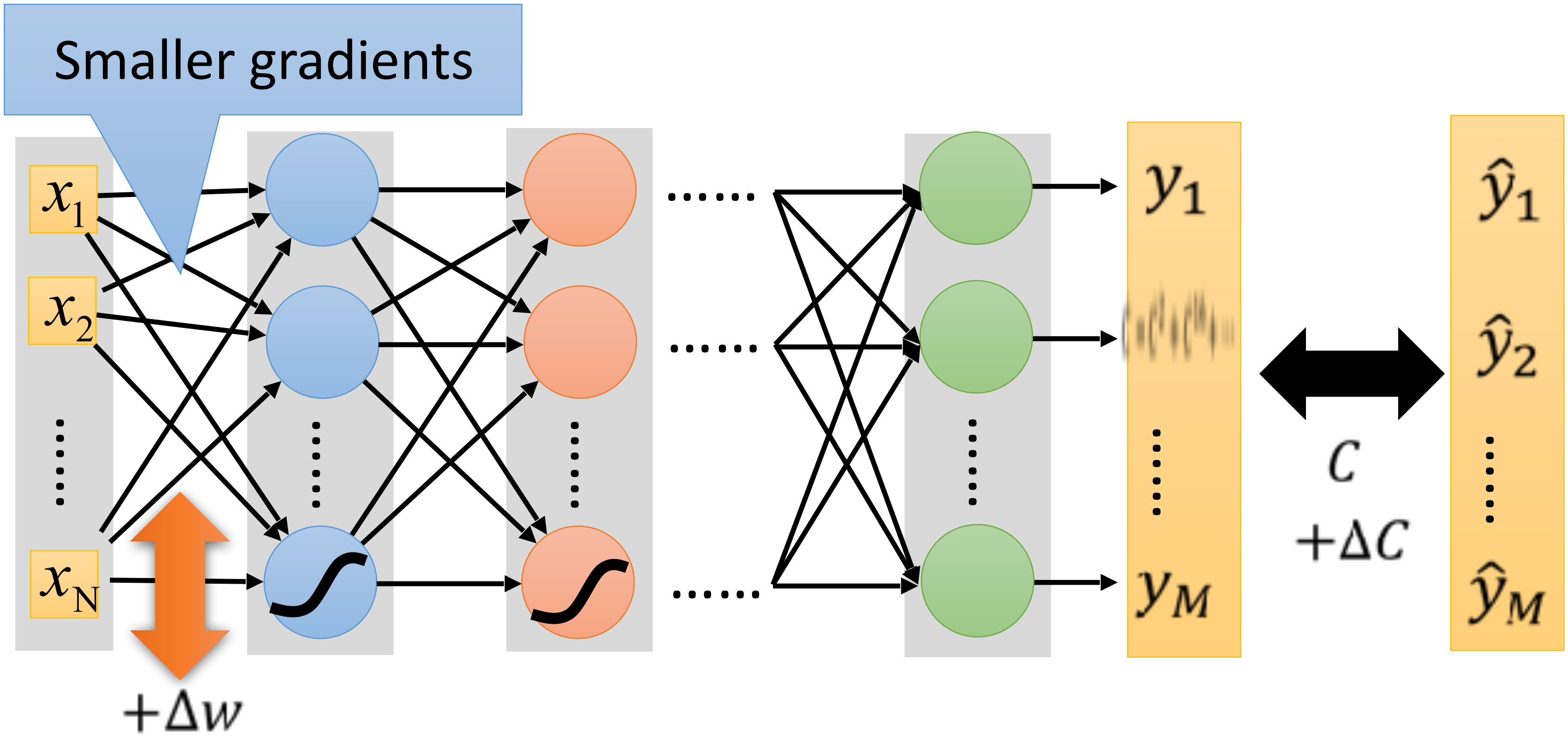
Vanishing Gradient Problem



Intuitive way to compute the gradient …

$$\frac{\partial C}{\partial w} =? \frac{\Delta C}{\Delta w}$$

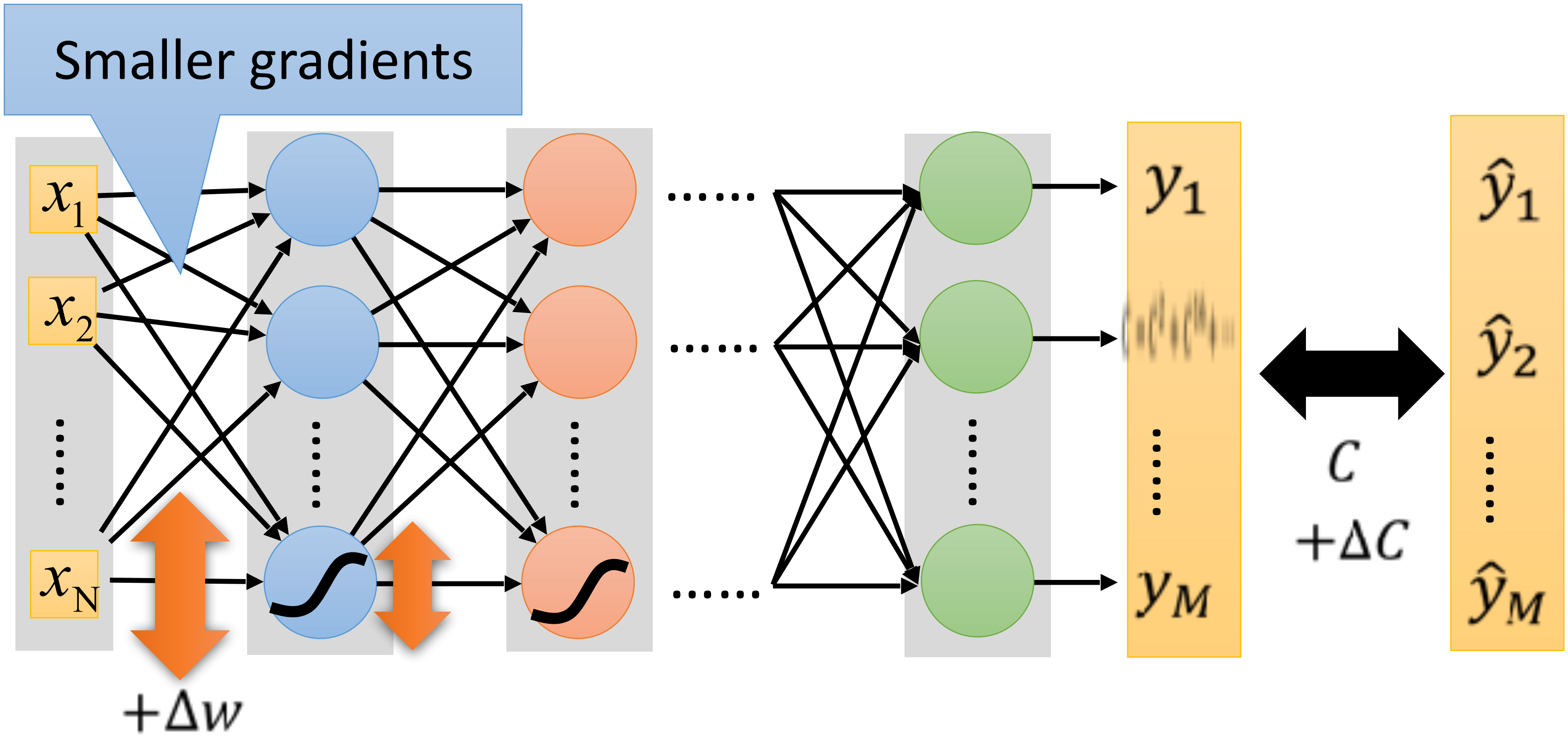
Vanishing Gradient Problem



Intuitive way to compute the gradient …

$$\frac{\partial C}{\partial w} =? \ \frac{\Delta C}{\Delta w}$$

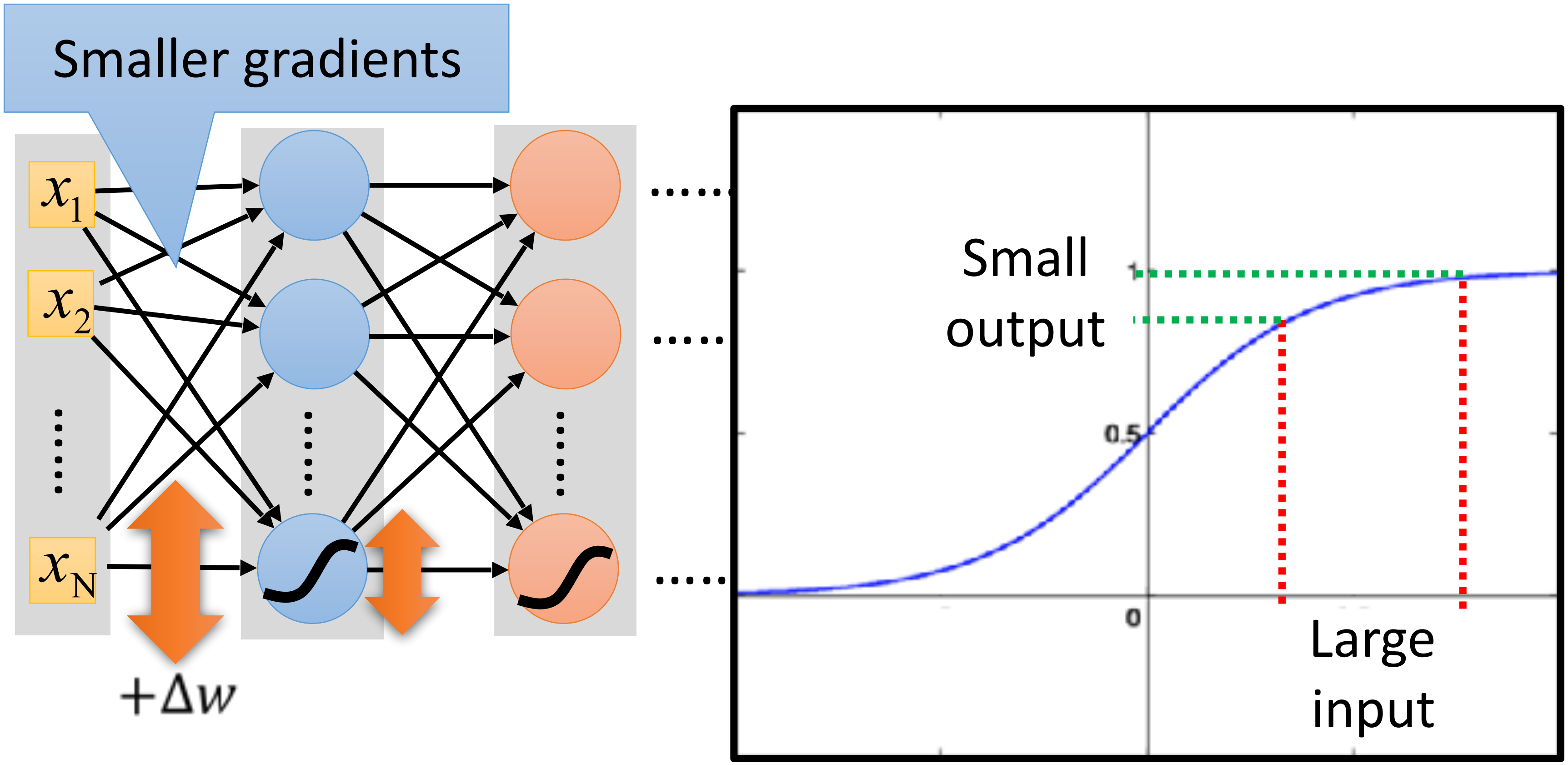
Vanishing Gradient Problem



Intuitive way to compute the gradient …

$$\frac{\partial C}{\partial w} =? \ \frac{\Delta C}{\Delta w}$$

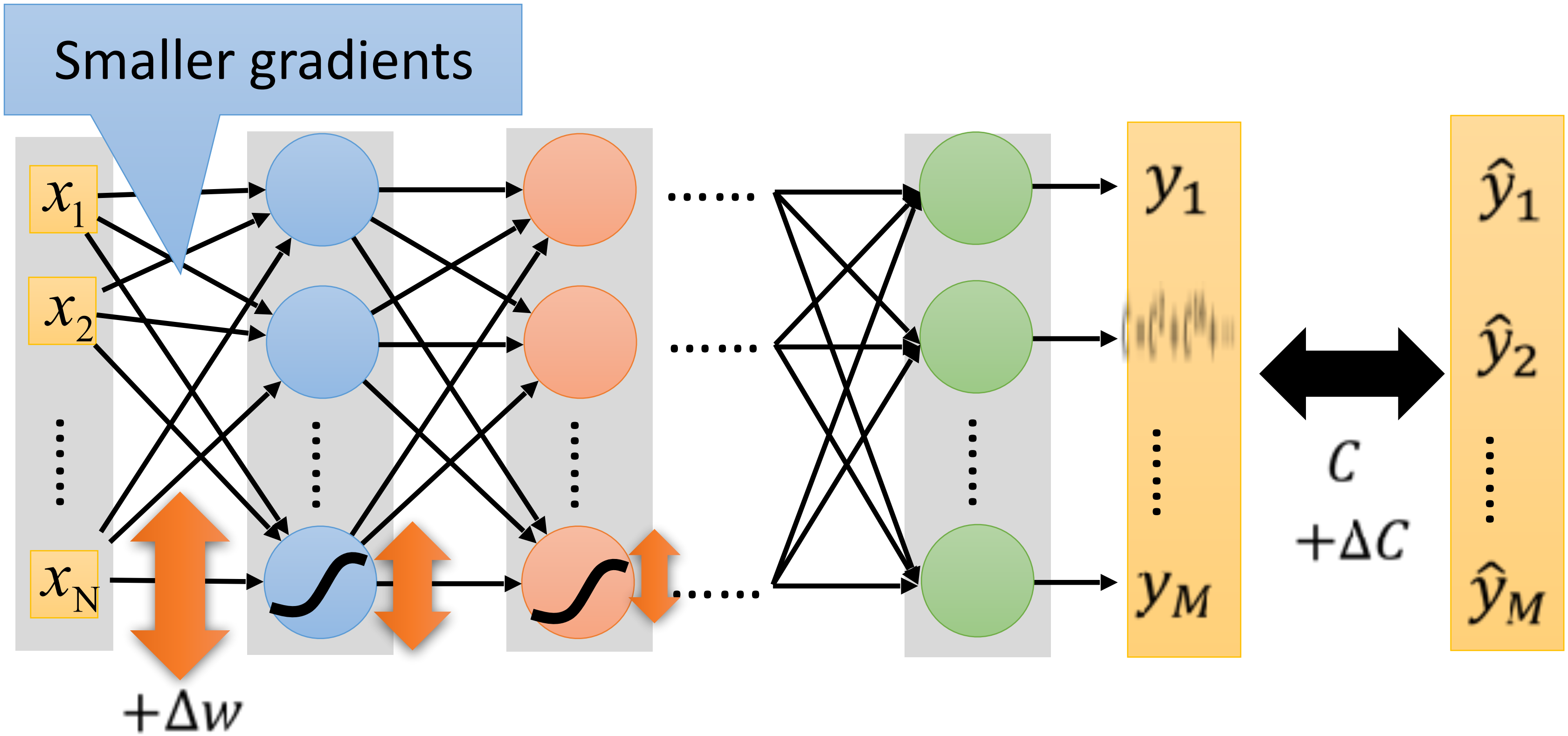
Vanishing Gradient Problem



Intuitive way to compute the gradient …

$$\frac{\partial C}{\partial w} =? \ \frac{\Delta C}{\Delta w}$$

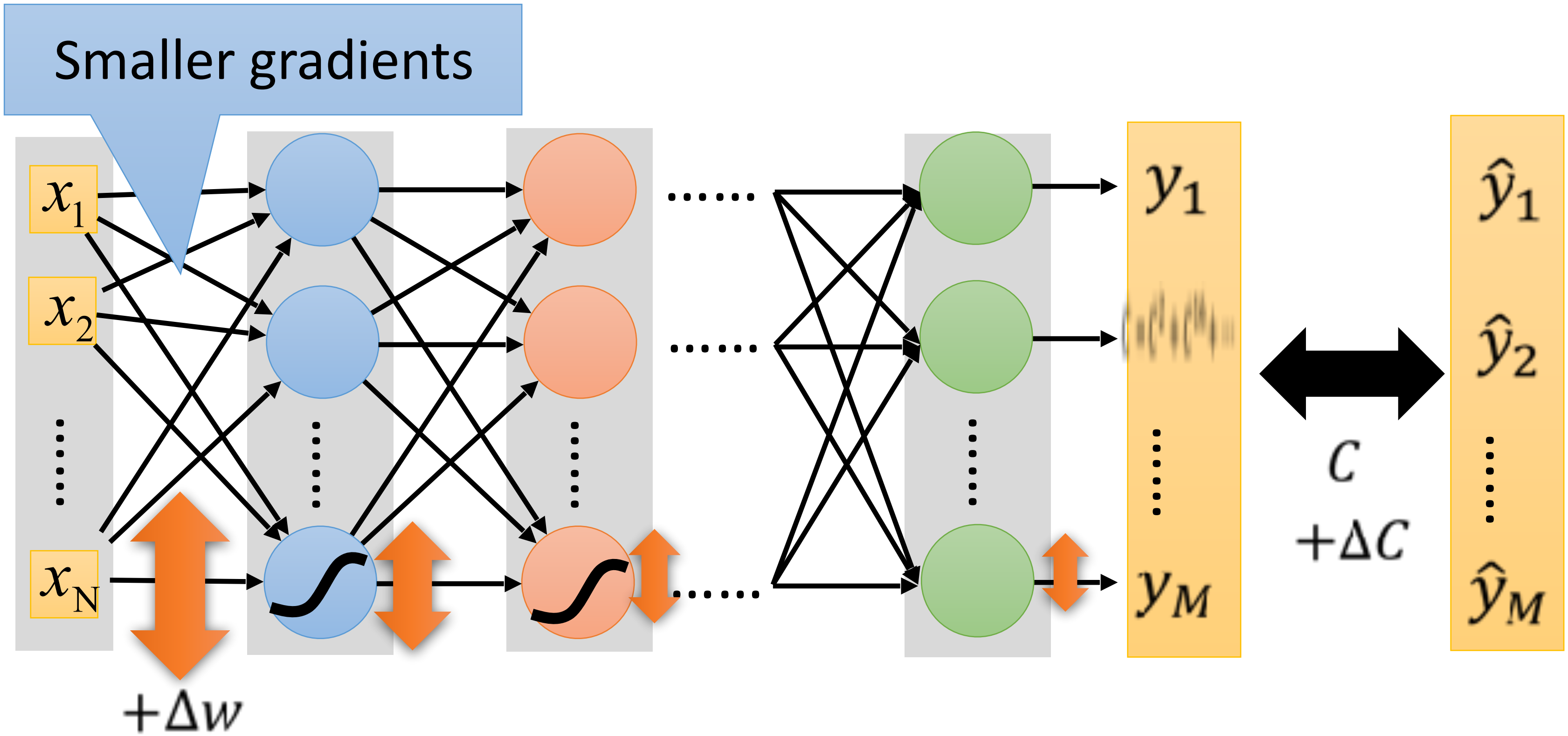
Vanishing Gradient Problem



Intuitive way to compute the gradient …

$$\frac{\partial C}{\partial w} =? \frac{\Delta C}{\Delta w}$$

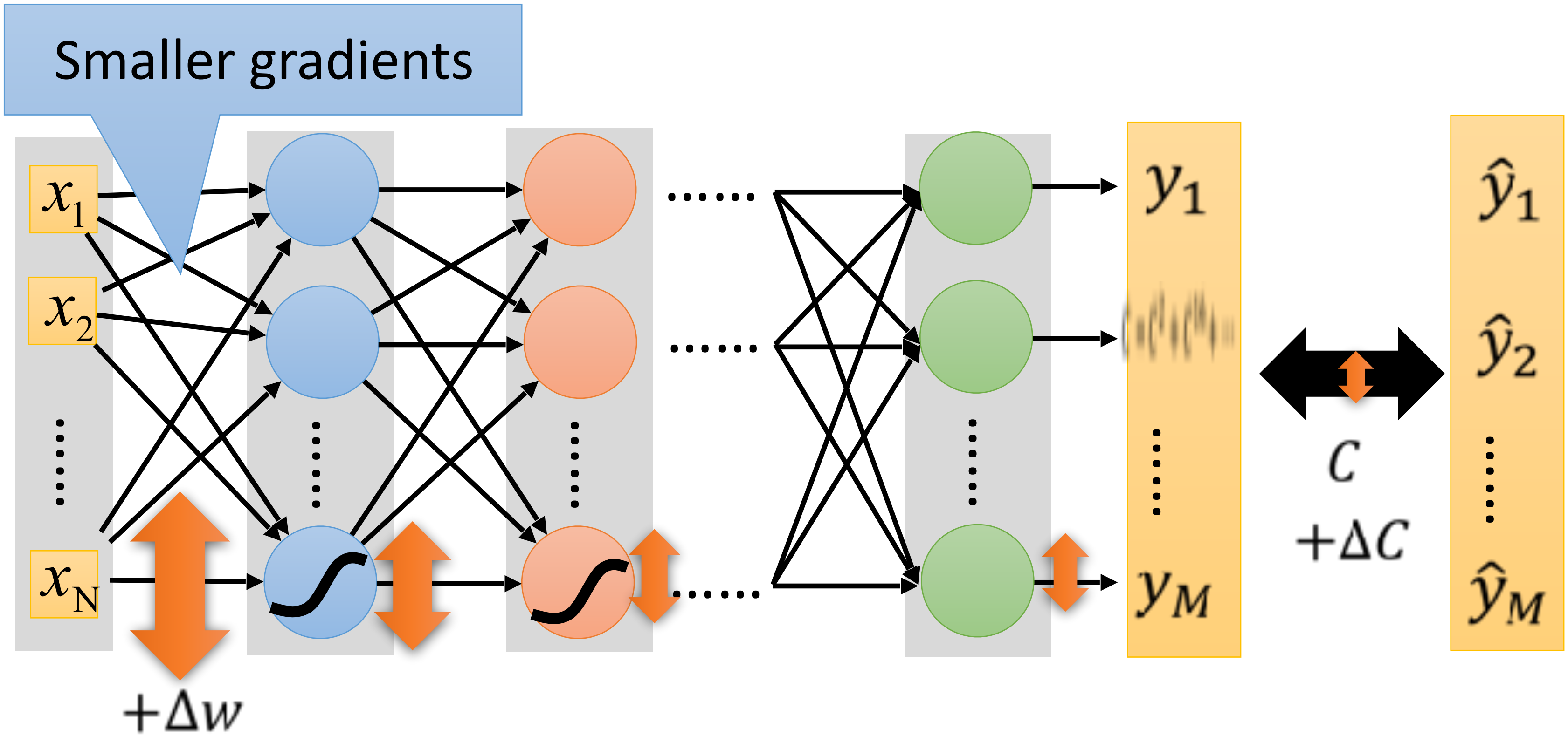
Vanishing Gradient Problem



Intuitive way to compute the gradient …

$$\frac{\partial C}{\partial w} =? \ \frac{\Delta C}{\Delta w}$$

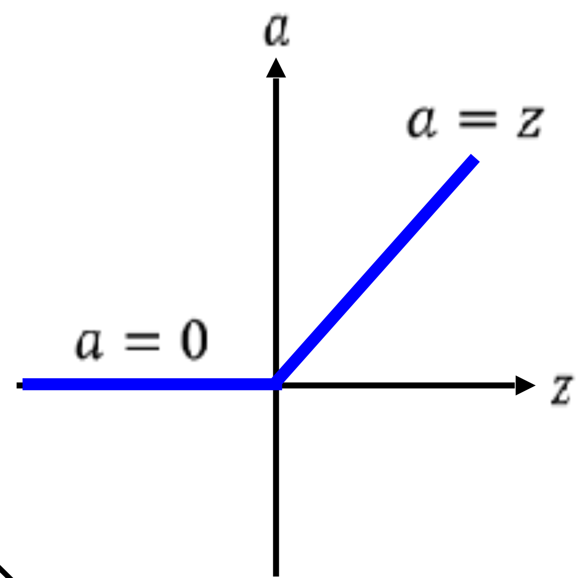
Vanishing Gradient Problem

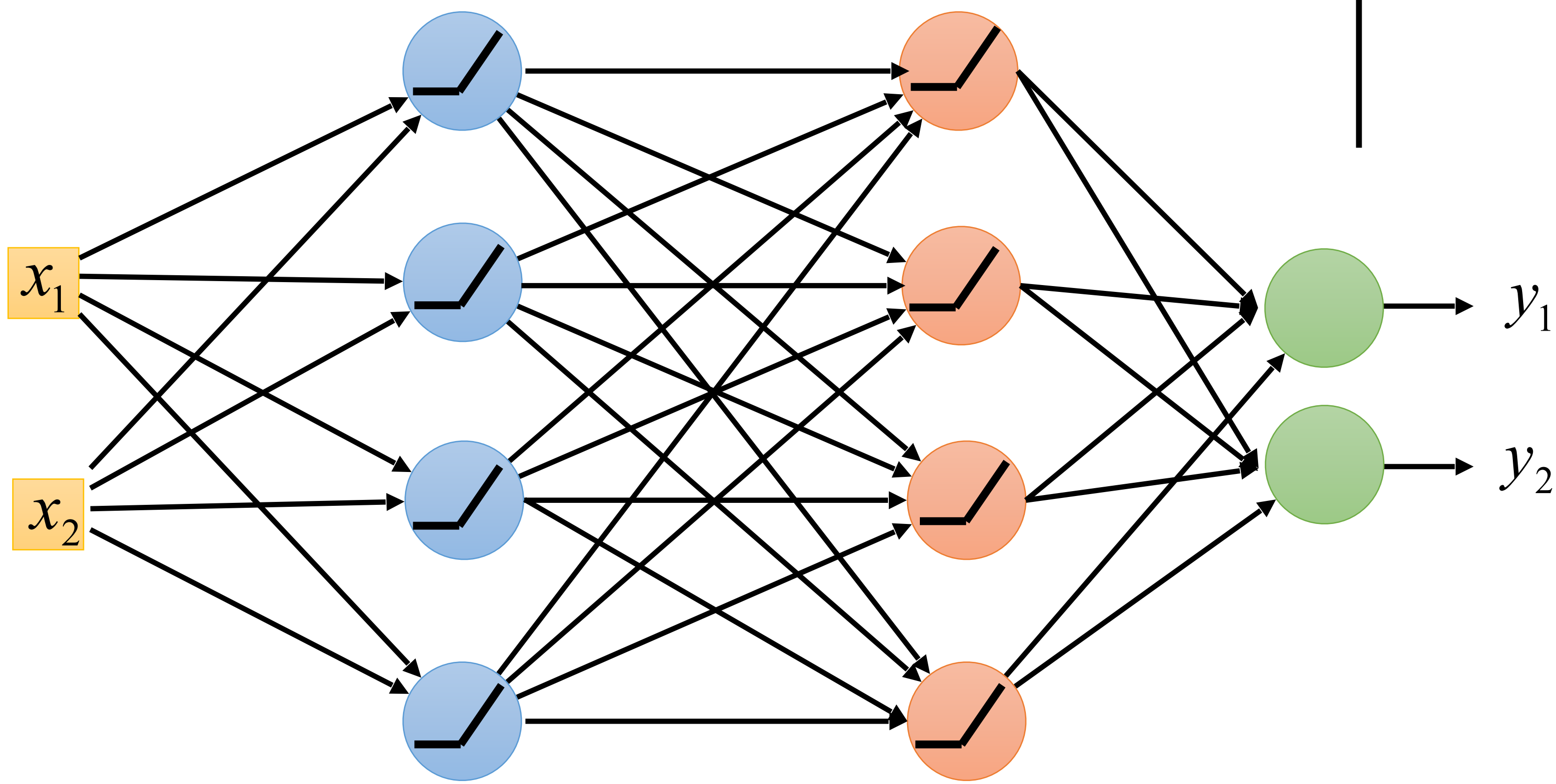


Intuitive way to compute the gradient …

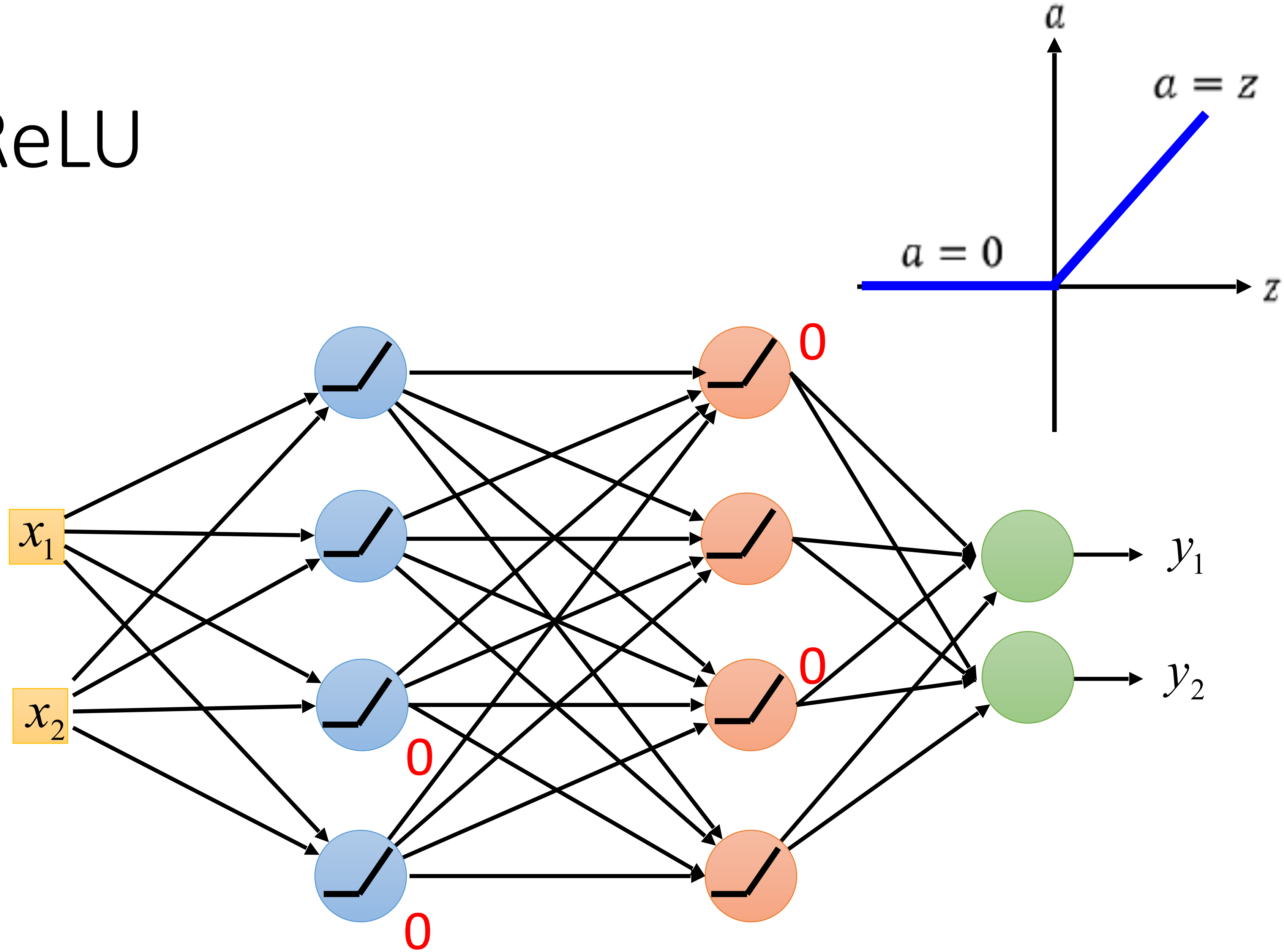
$$\frac{\partial C}{\partial w} =? \ \frac{\Delta C}{\Delta w}$$

ReLU

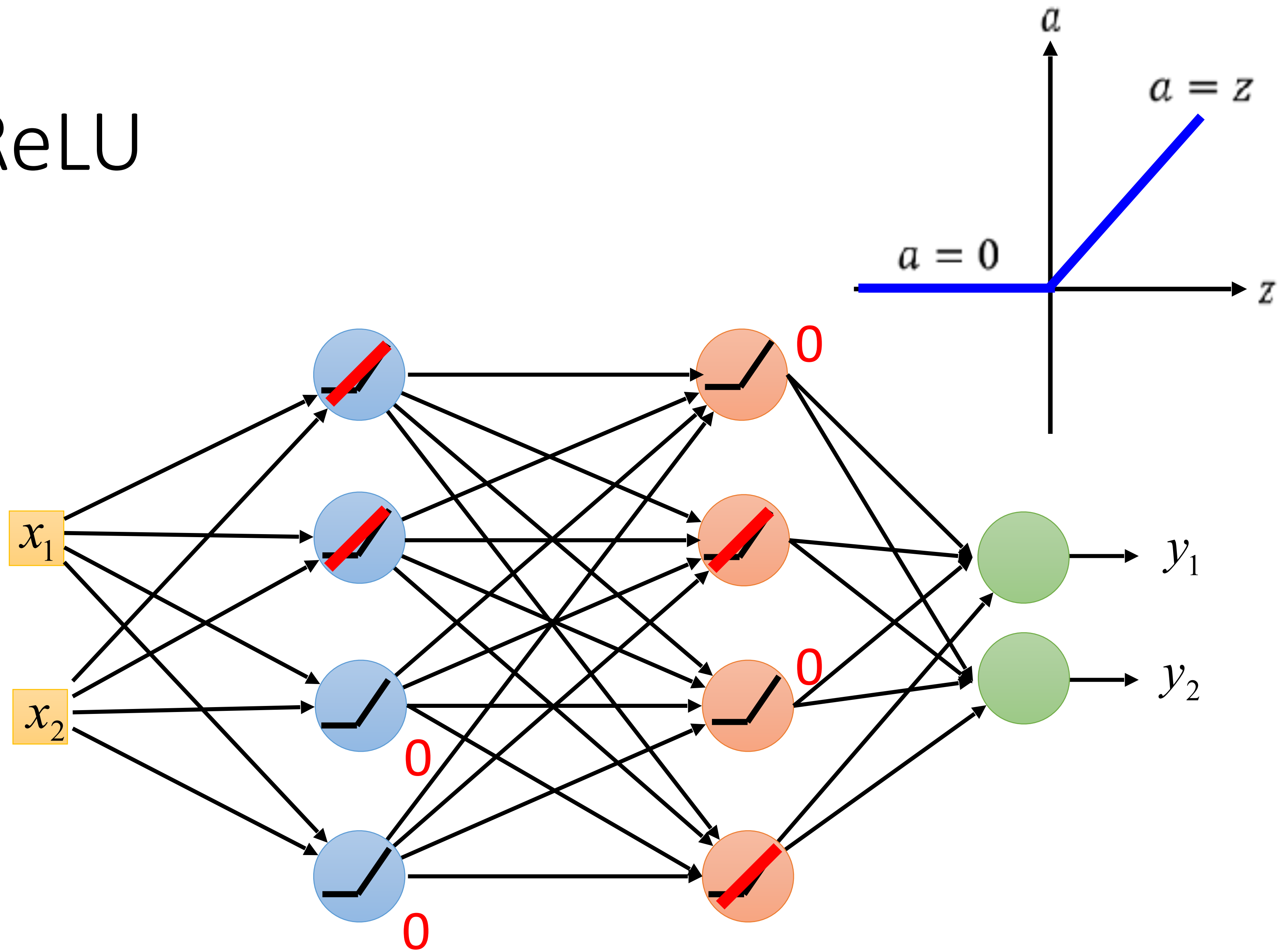




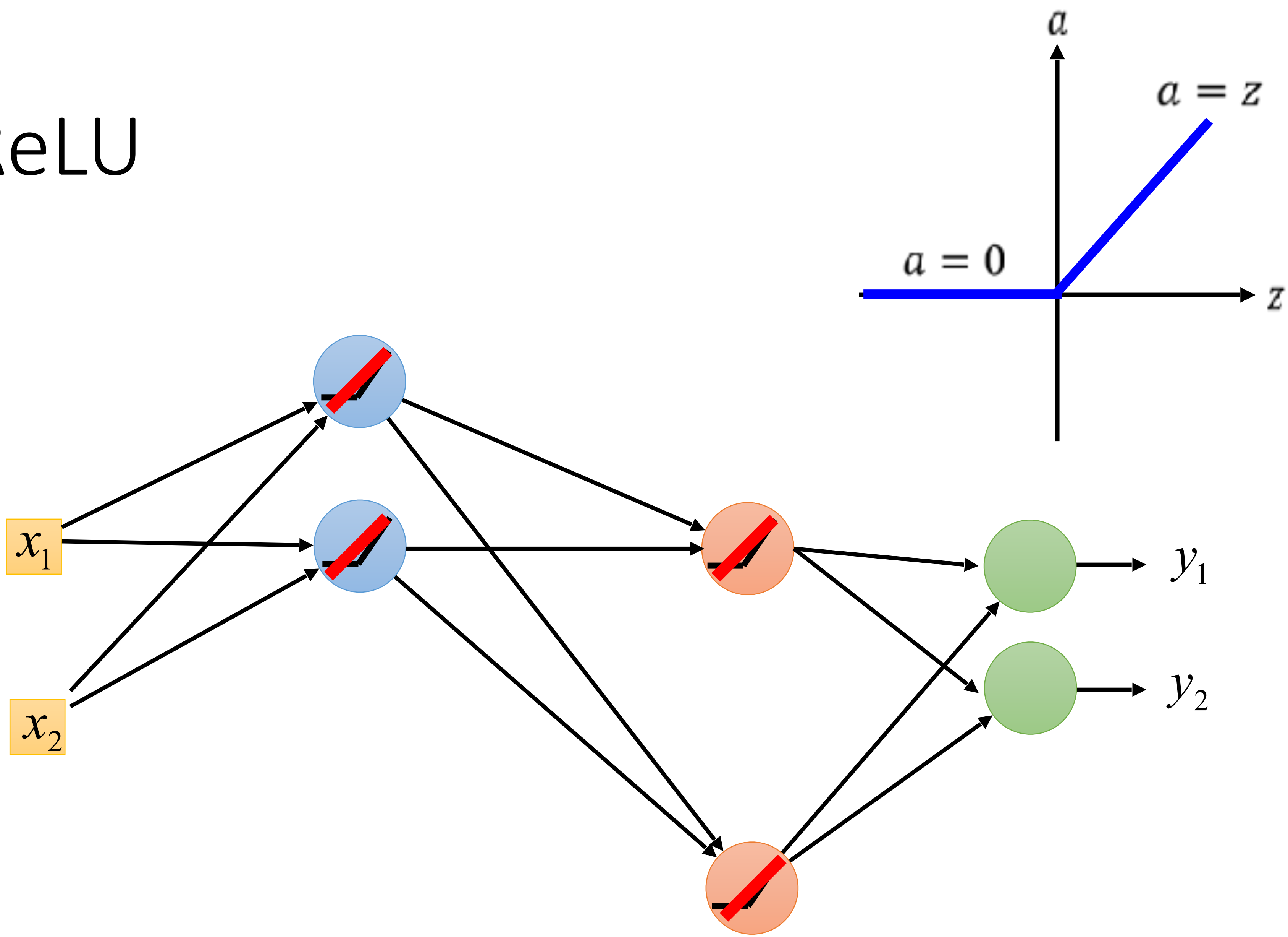
ReLU



ReLU

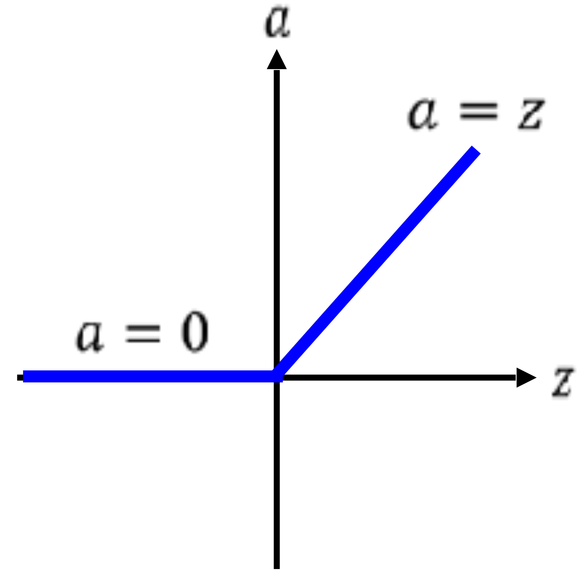


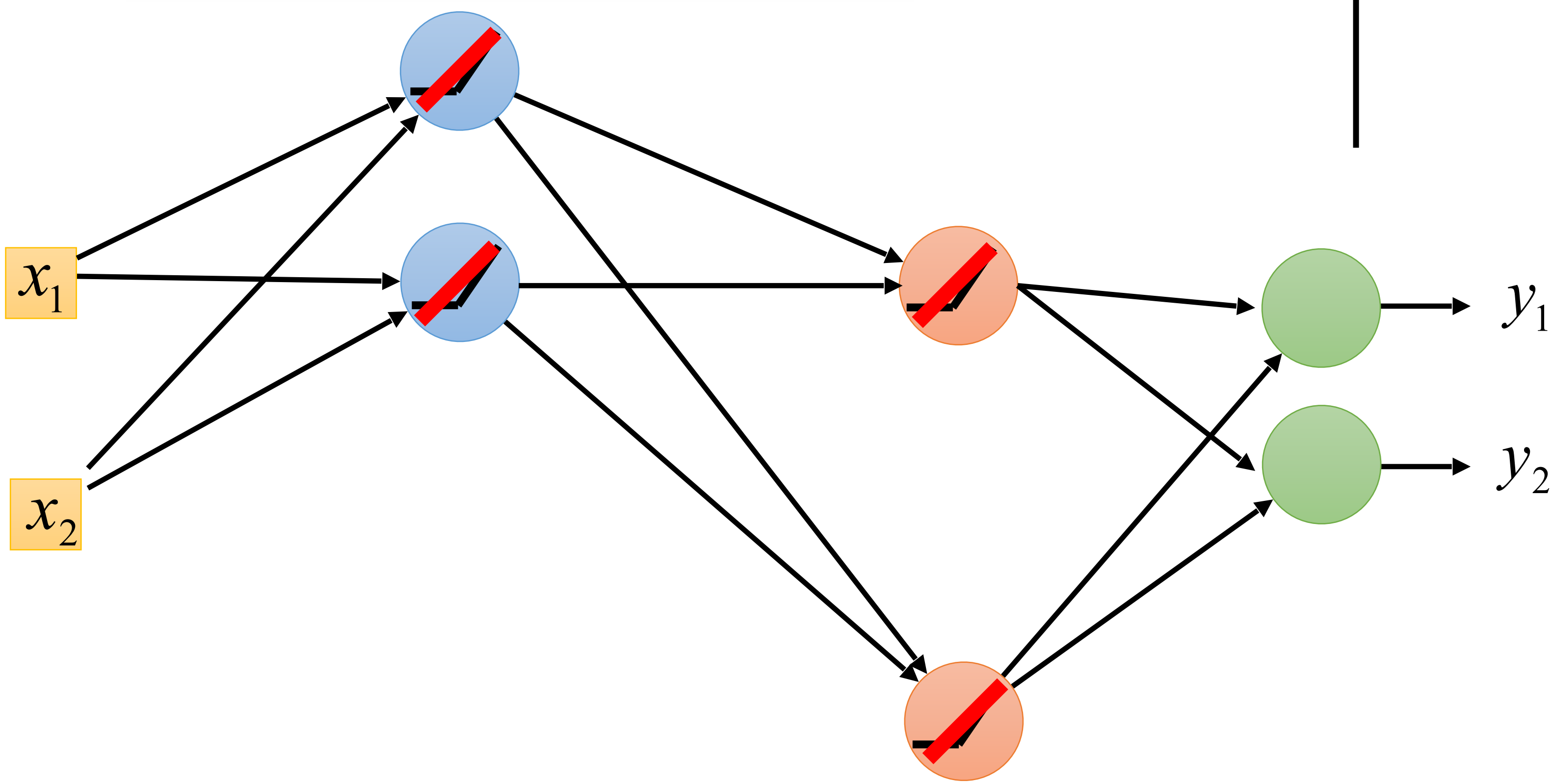
ReLU



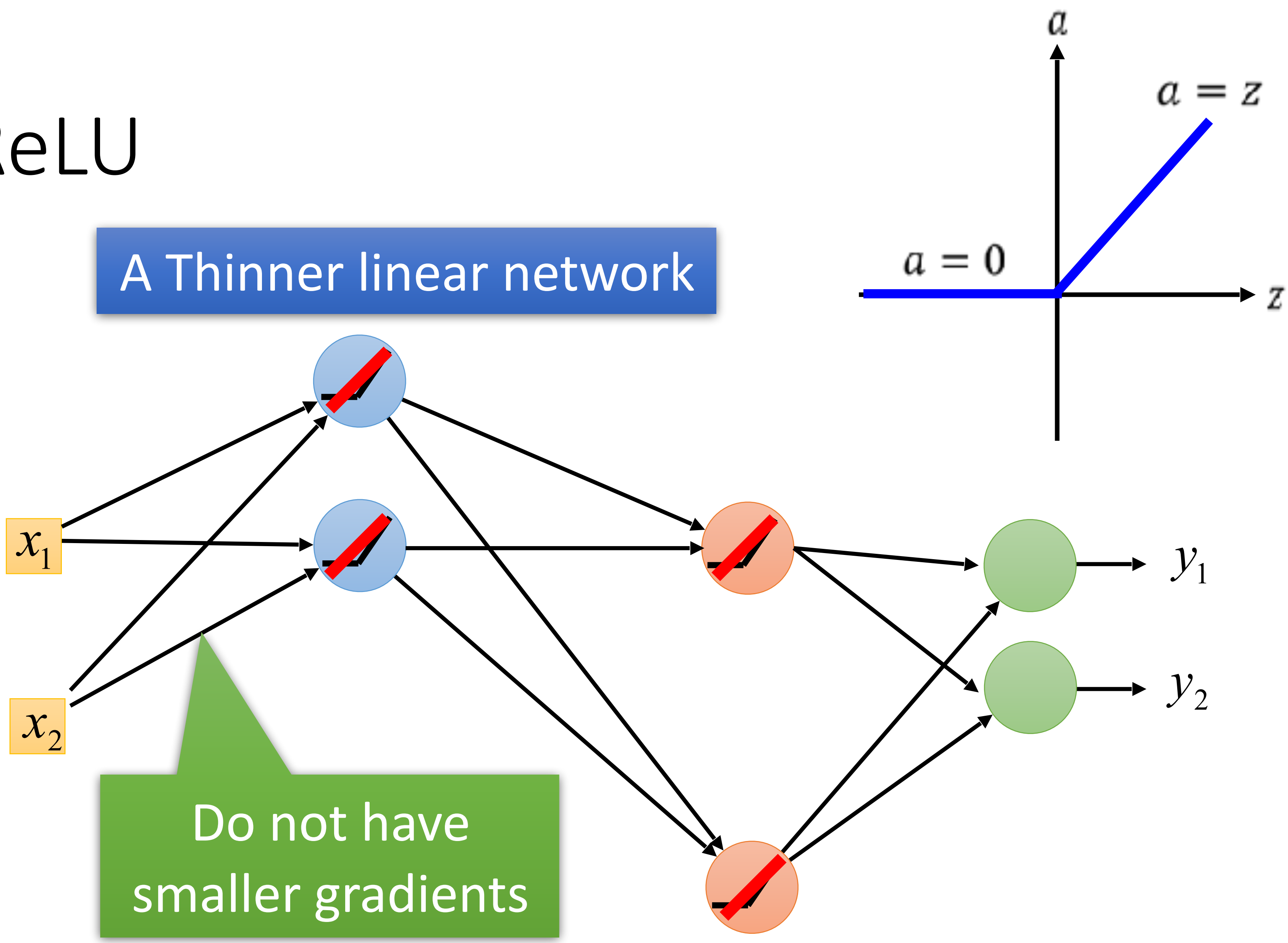
ReLU

A Thinner linear network





ReLU


A Thinner linear network
a
$a = z$
$a = 0$
z
x_1
x_2
y_1
y_2
Do not have smaller gradients

Maxout

ReLU is a special cases of Maxout

- Learnable activation function [Ian J. Goodfellow, ICML'13]

Maxout

ReLU is a special cases of Maxout

- Learnable activation function [Ian J. Goodfellow, ICML'13]

Input

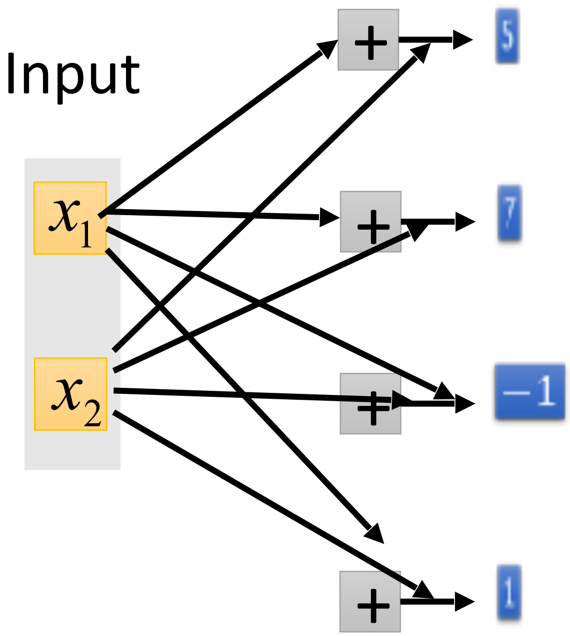
x_1

x_2

Maxout

ReLU is a special cases of Maxout

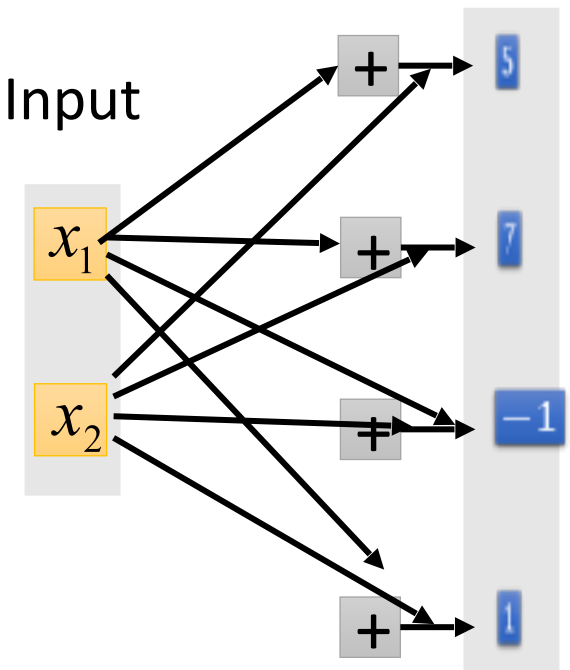
- Learnable activation function [Ian J. Goodfellow, ICML'13]



Maxout

ReLU is a special cases of Maxout

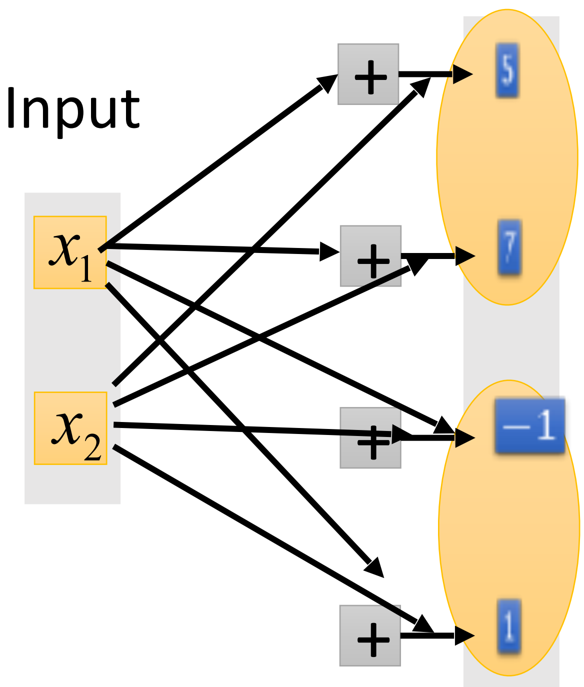
- Learnable activation function [Ian J. Goodfellow, ICML'13]



Maxout

ReLU is a special cases of Maxout

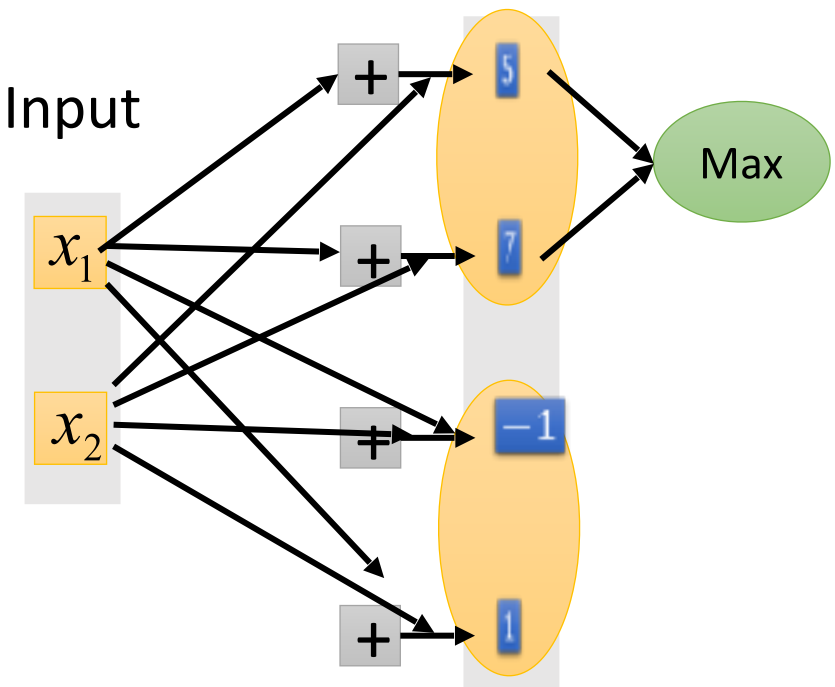
- Learnable activation function [Ian J. Goodfellow, ICML'13]



Maxout

ReLU is a special cases of Maxout

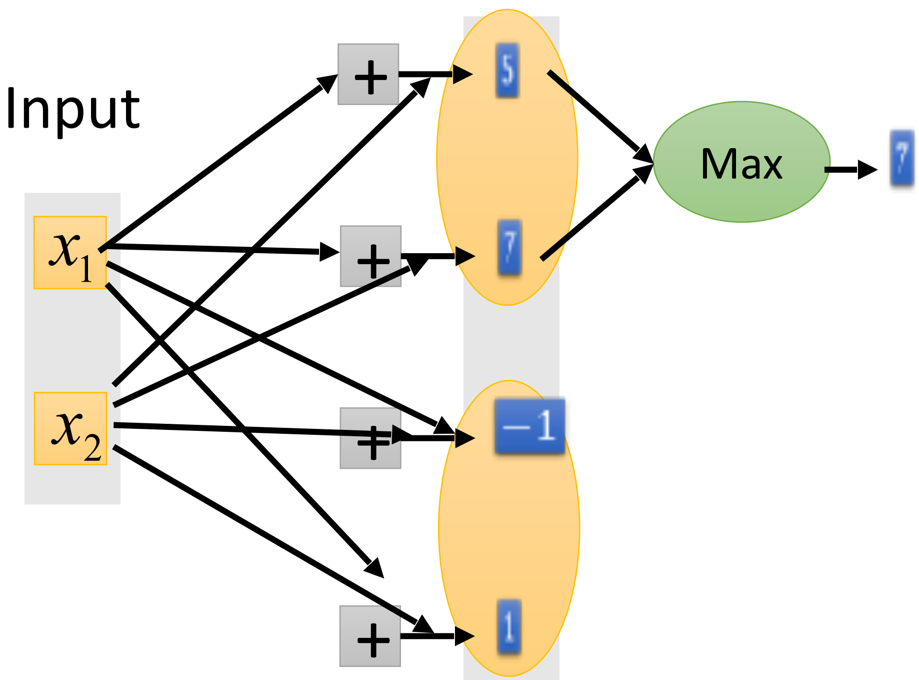
- Learnable activation function [Ian J. Goodfellow, ICML'13]



Maxout

ReLU is a special cases of Maxout

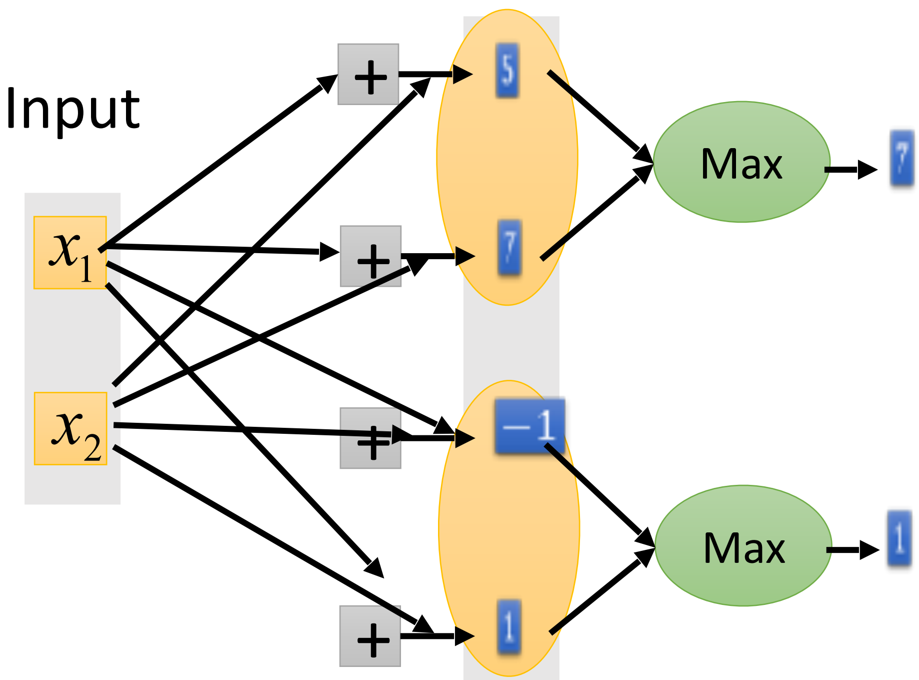
- Learnable activation function [Ian J. Goodfellow, ICML'13]



Maxout

ReLU is a special cases of Maxout

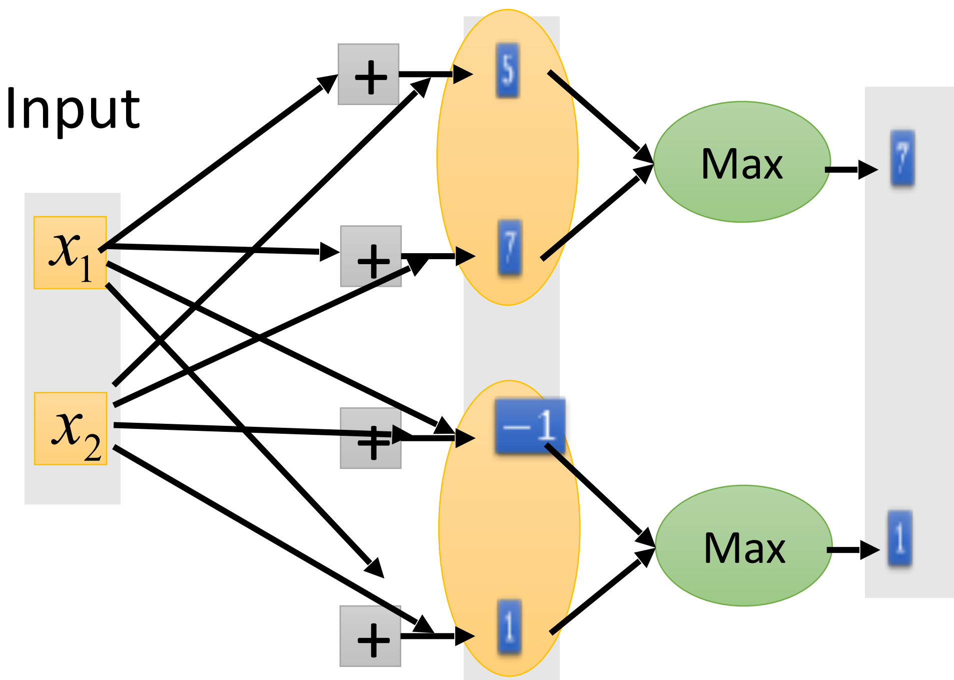
- Learnable activation function [Ian J. Goodfellow, ICML'13]



Maxout

ReLU is a special cases of Maxout

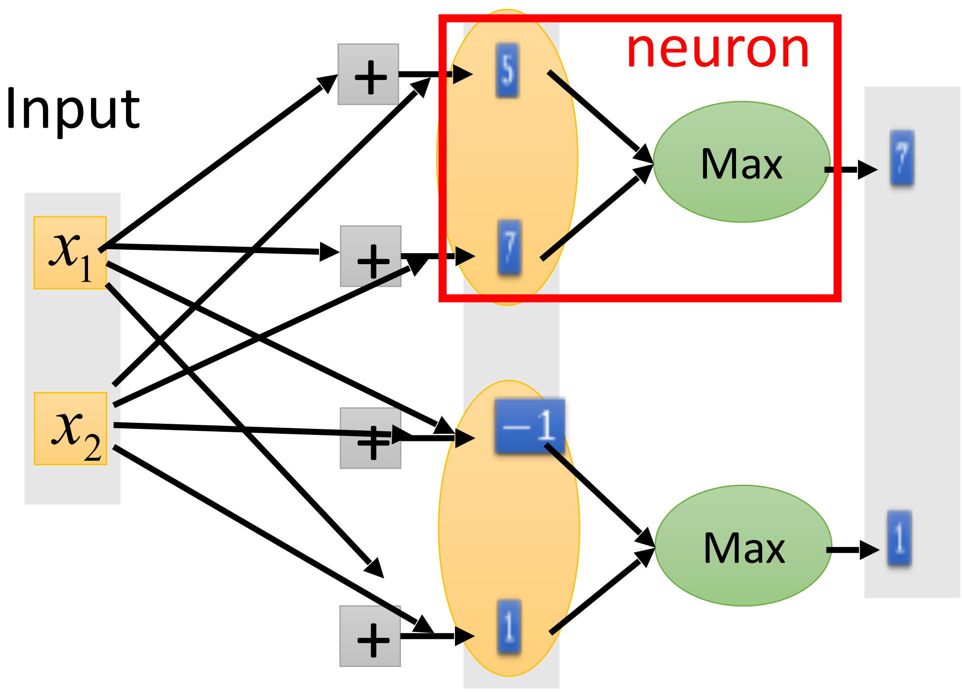
- Learnable activation function [Ian J. Goodfellow, ICML'13]



Maxout

ReLU is a special cases of Maxout

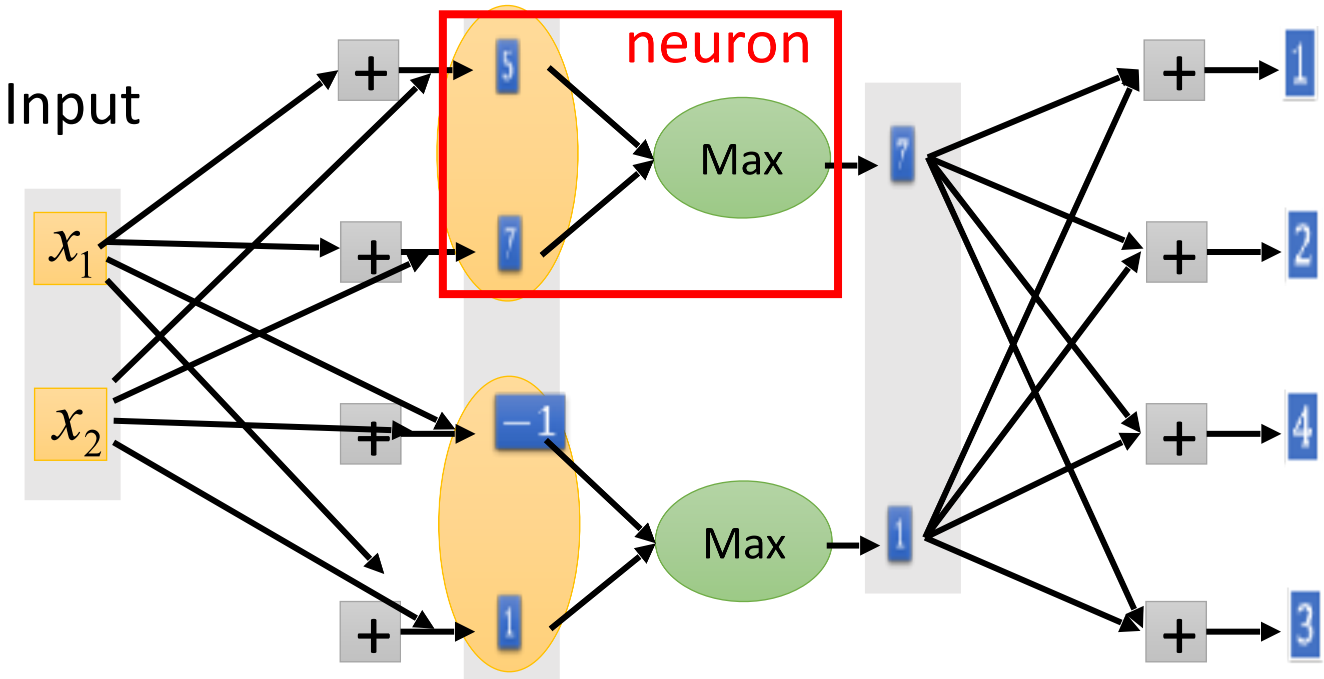
- Learnable activation function [Ian J. Goodfellow, ICML'13]



Maxout

ReLU is a special cases of Maxout

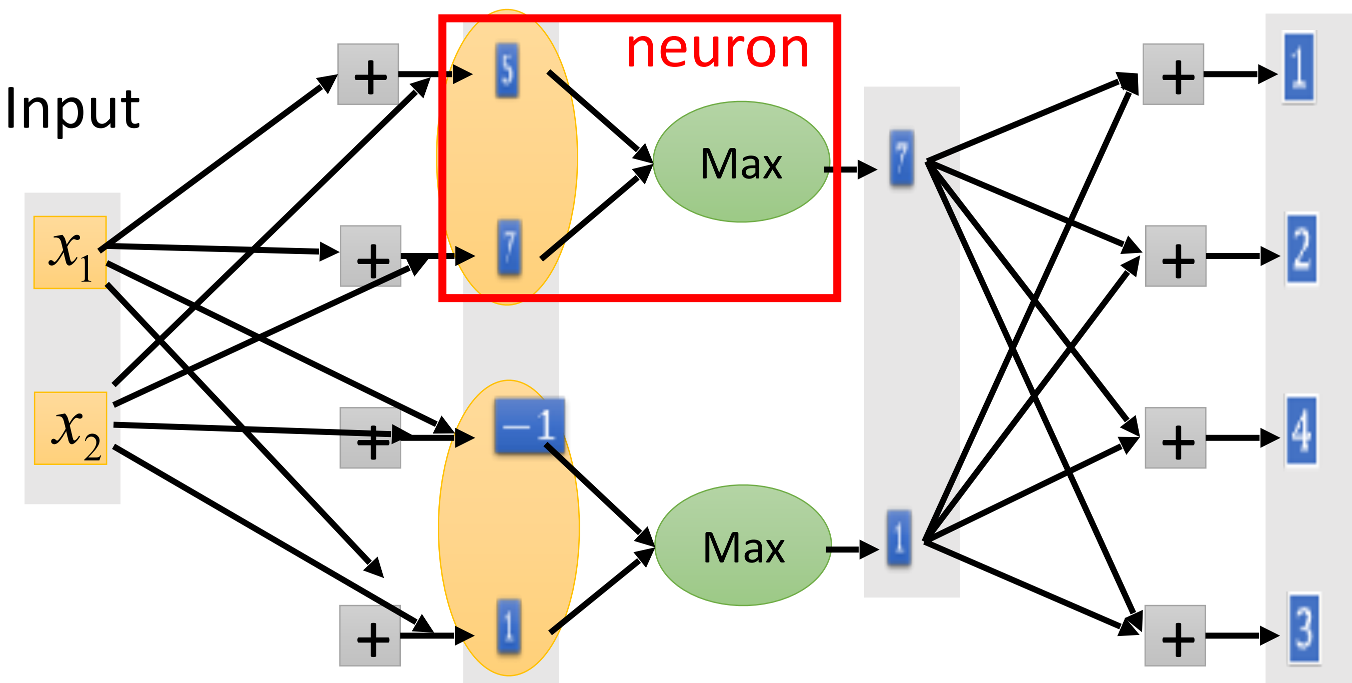
- Learnable activation function [Ian J. Goodfellow, ICML'13]



Maxout

ReLU is a special cases of Maxout

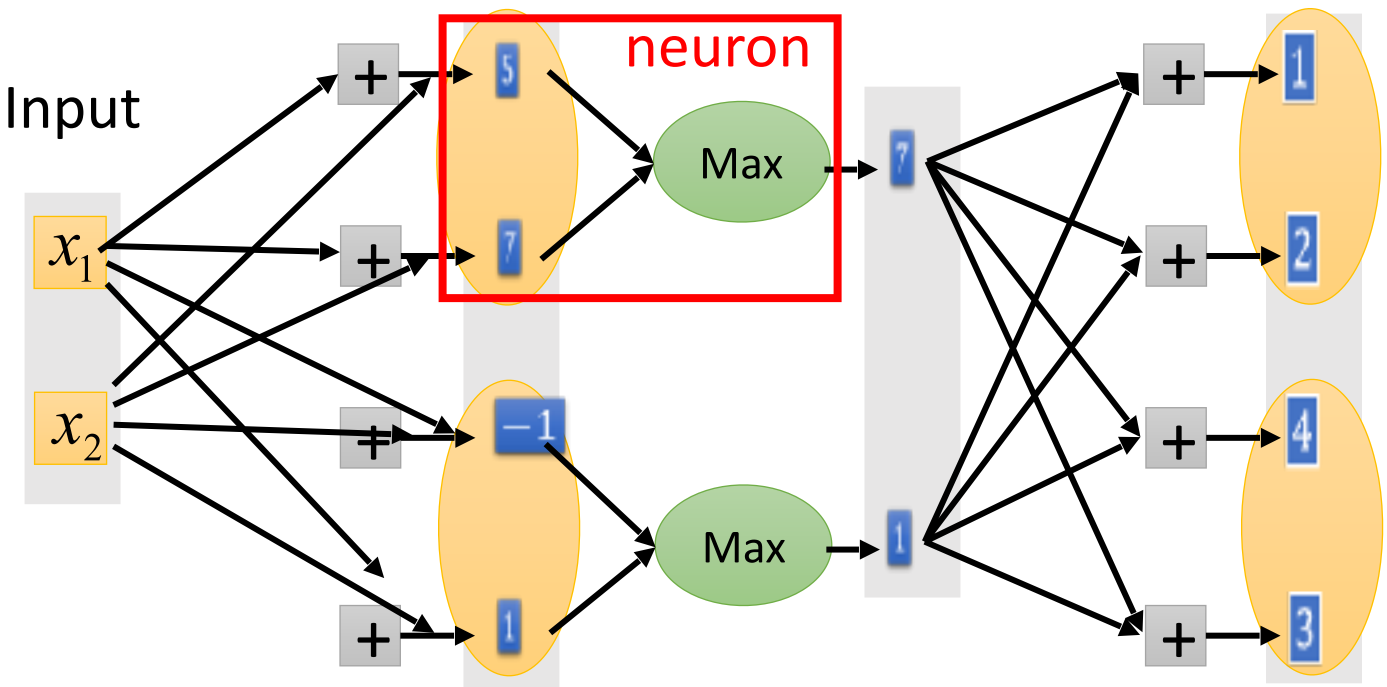
- Learnable activation function [Ian J. Goodfellow, ICML'13]



Maxout

ReLU is a special cases of Maxout

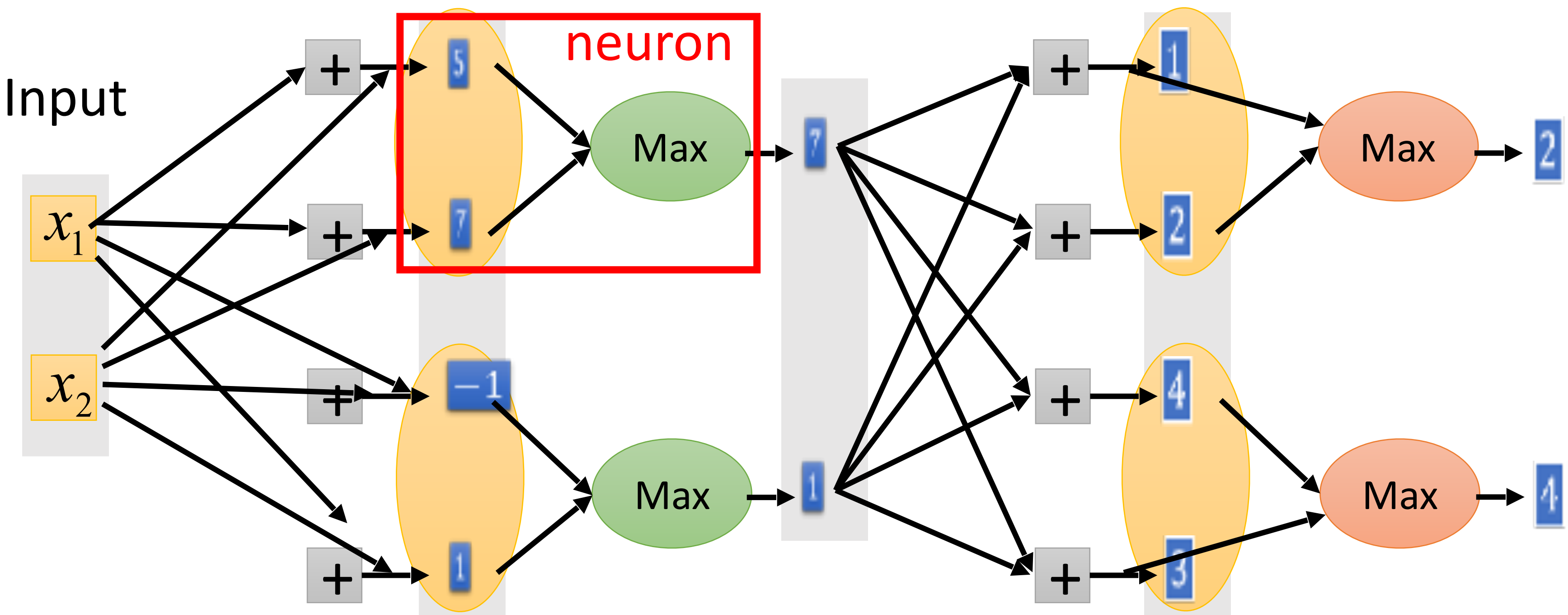
- Learnable activation function [Ian J. Goodfellow, ICML'13]



Maxout

ReLU is a special cases of Maxout

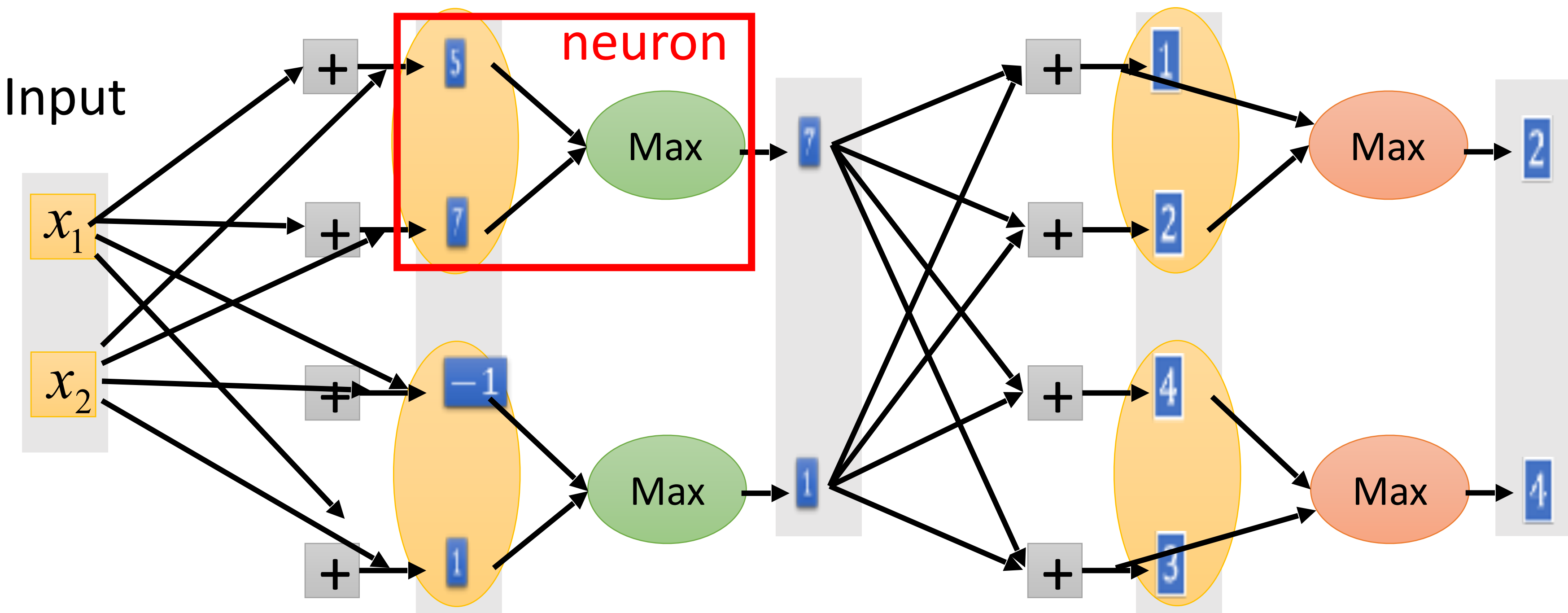
- Learnable activation function [Ian J. Goodfellow, ICML'13]



Maxout

ReLU is a special cases of Maxout

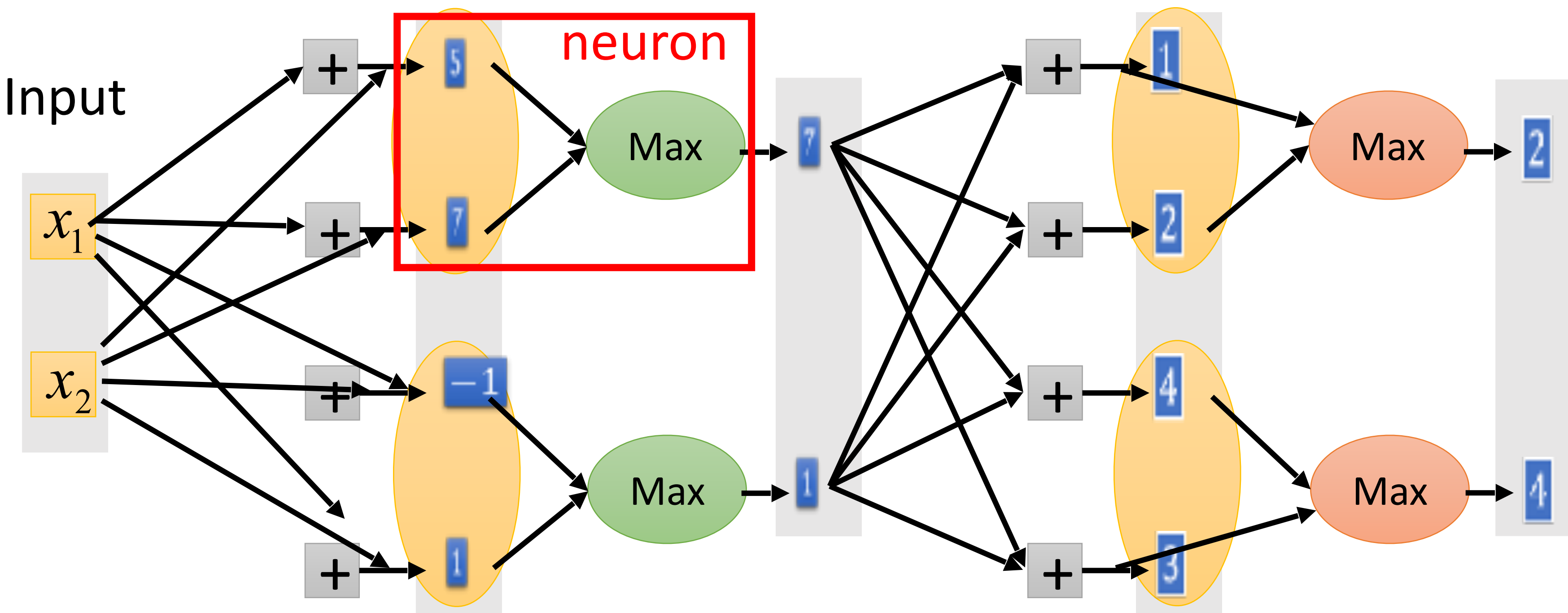
- Learnable activation function [Ian J. Goodfellow, ICML'13]



Maxout

ReLU is a special cases of Maxout

- Learnable activation function [Ian J. Goodfellow, ICML'13]



You can have more than 2 elements in a group.

Maxout

ReLU is a special cases of Maxout

- Learnable activation function [Ian J. Goodfellow, ICML'13]

Maxout

ReLU is a special cases of Maxout

- Learnable activation function [Ian J. Goodfellow, ICML'13]
  - Activation function in maxout network can be any piecewise linear convex function

Maxout

ReLU is a special cases of Maxout

- Learnable activation function [Ian J. Goodfellow, ICML'13]
  - Activation function in maxout network can be any piecewise linear convex function
  - How many pieces depending on how many elements in a group

Maxout

ReLU is a special cases of Maxout

- Learnable activation function [Ian J. Goodfellow, ICML'13]
  - Activation function in maxout network can be any piecewise linear convex function
  - How many pieces depending on how many elements in a group

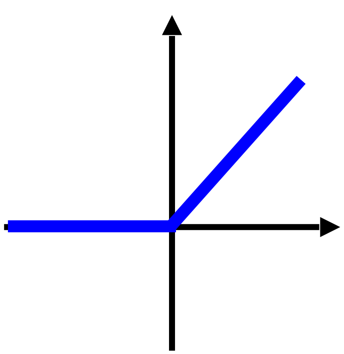
2 elements in a group

Maxout

ReLU is a special cases of Maxout

- Learnable activation function [Ian J. Goodfellow, ICML'13]
  - Activation function in maxout network can be any piecewise linear convex function
  - How many pieces depending on how many elements in a group

2 elements in a group

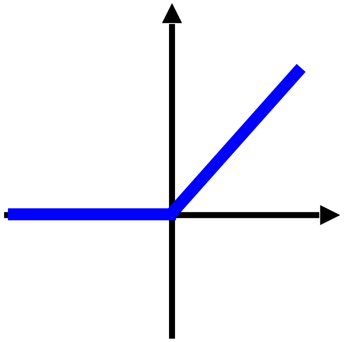


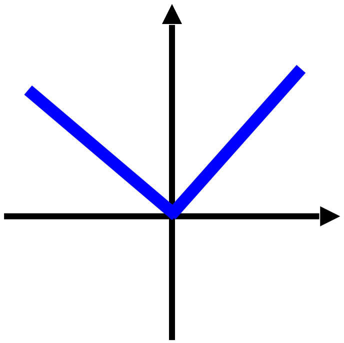
Maxout

ReLU is a special cases of Maxout

- Learnable activation function [Ian J. Goodfellow, ICML'13]
  - Activation function in maxout network can be any piecewise linear convex function
  - How many pieces depending on how many elements in a group

2 elements in a group





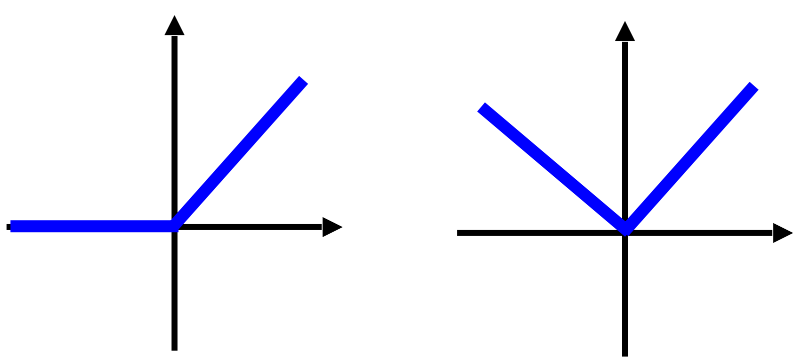
Maxout

ReLU is a special cases of Maxout

- Learnable activation function [Ian J. Goodfellow, ICML'13]
  - Activation function in maxout network can be any piecewise linear convex function
  - How many pieces depending on how many elements in a group

2 elements in a group

3 elements in a group

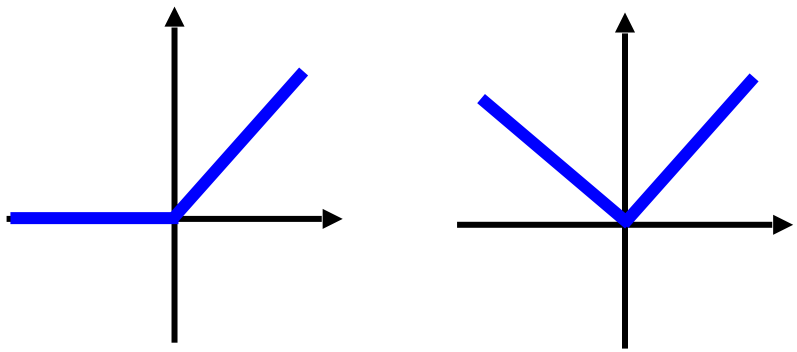


Maxout

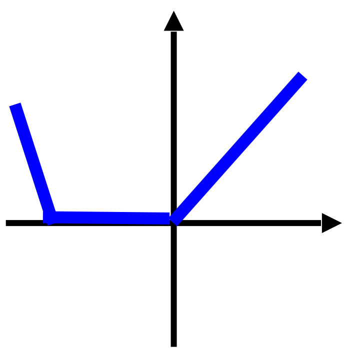
ReLU is a special cases of Maxout

- Learnable activation function [Ian J. Goodfellow, ICML'13]
  - Activation function in maxout network can be any piecewise linear convex function
  - How many pieces depending on how many elements in a group

2 elements in a group



3 elements in a group

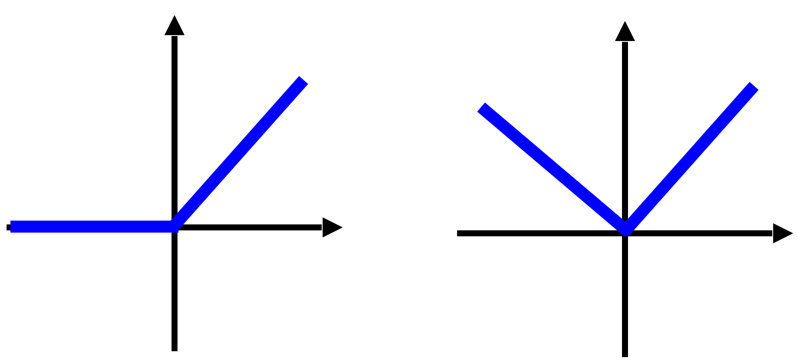


Maxout

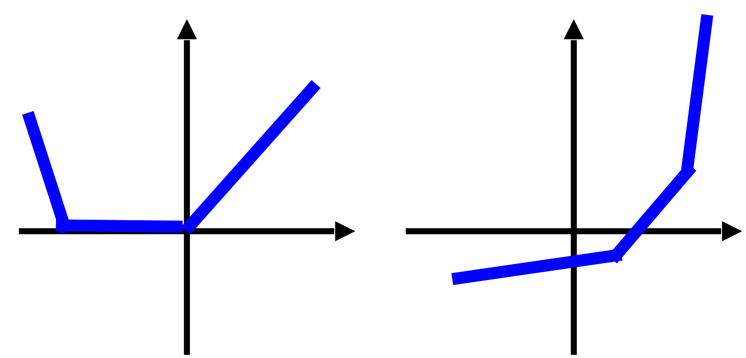
ReLU is a special cases of Maxout

- Learnable activation function [Ian J. Goodfellow, ICML'13]
  - Activation function in maxout network can be any piecewise linear convex function
  - How many pieces depending on how many elements in a group

2 elements in a group



3 elements in a group

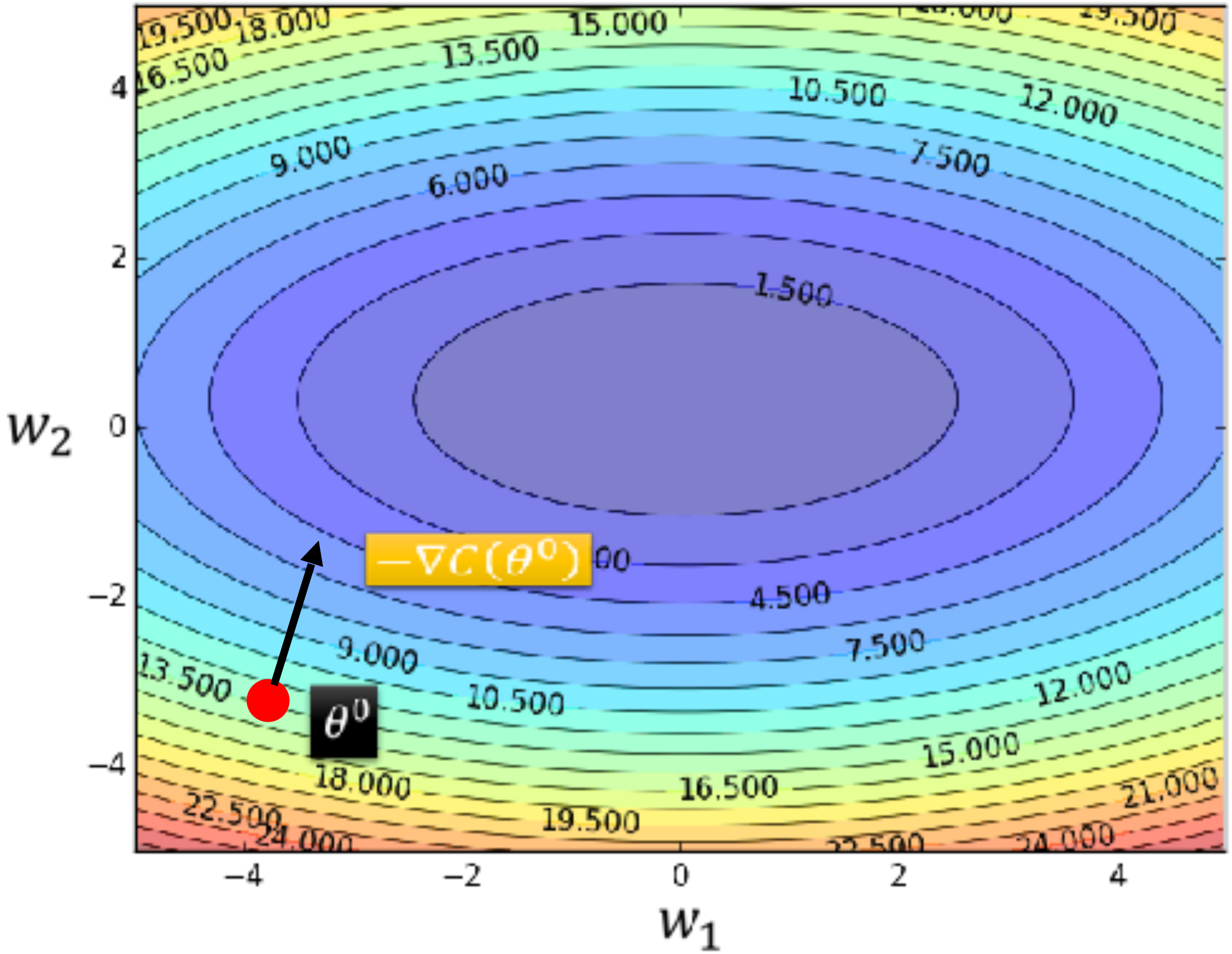


Part III:

Tips for Training DNN

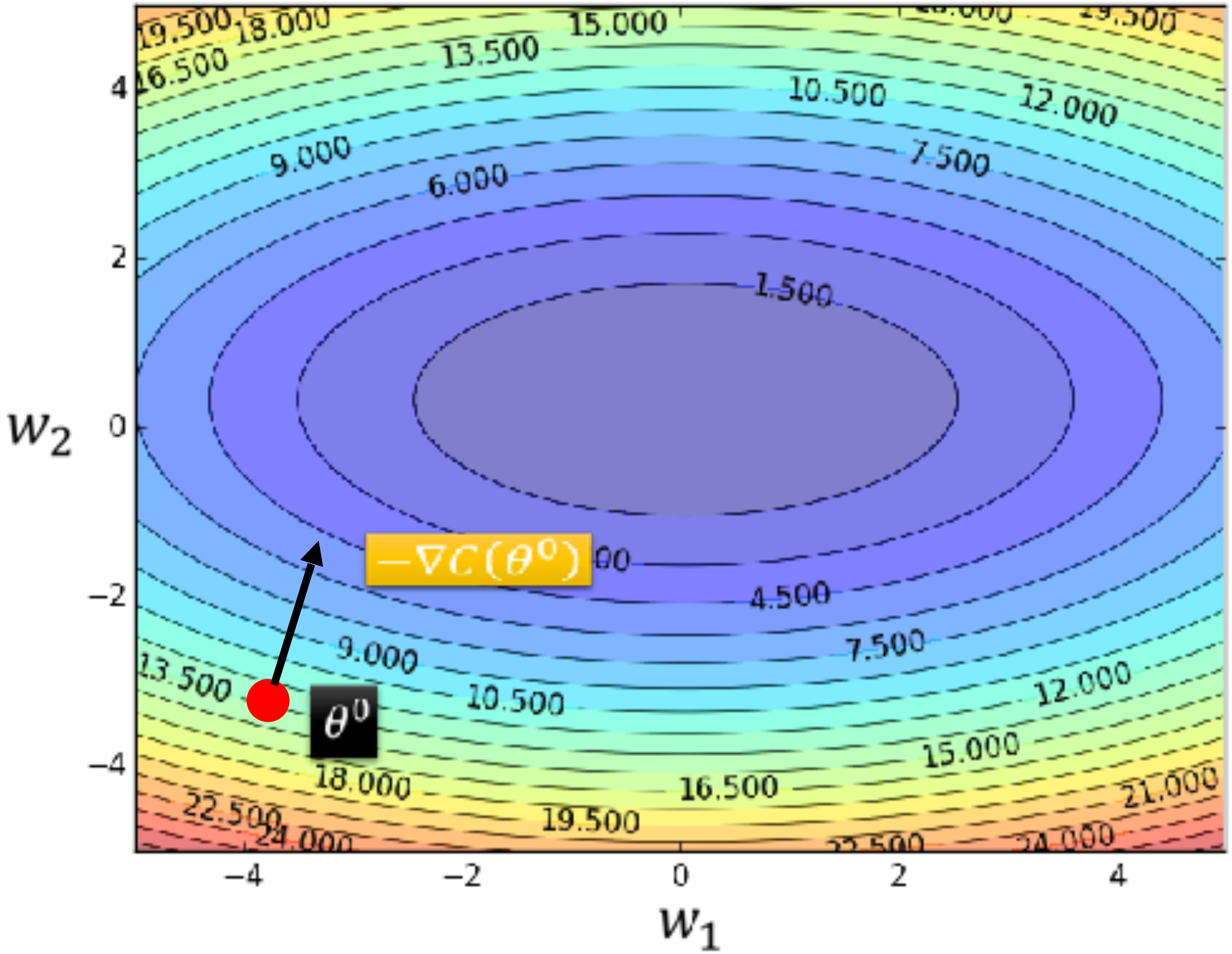
Adaptive Learning Rate

Learning Rate



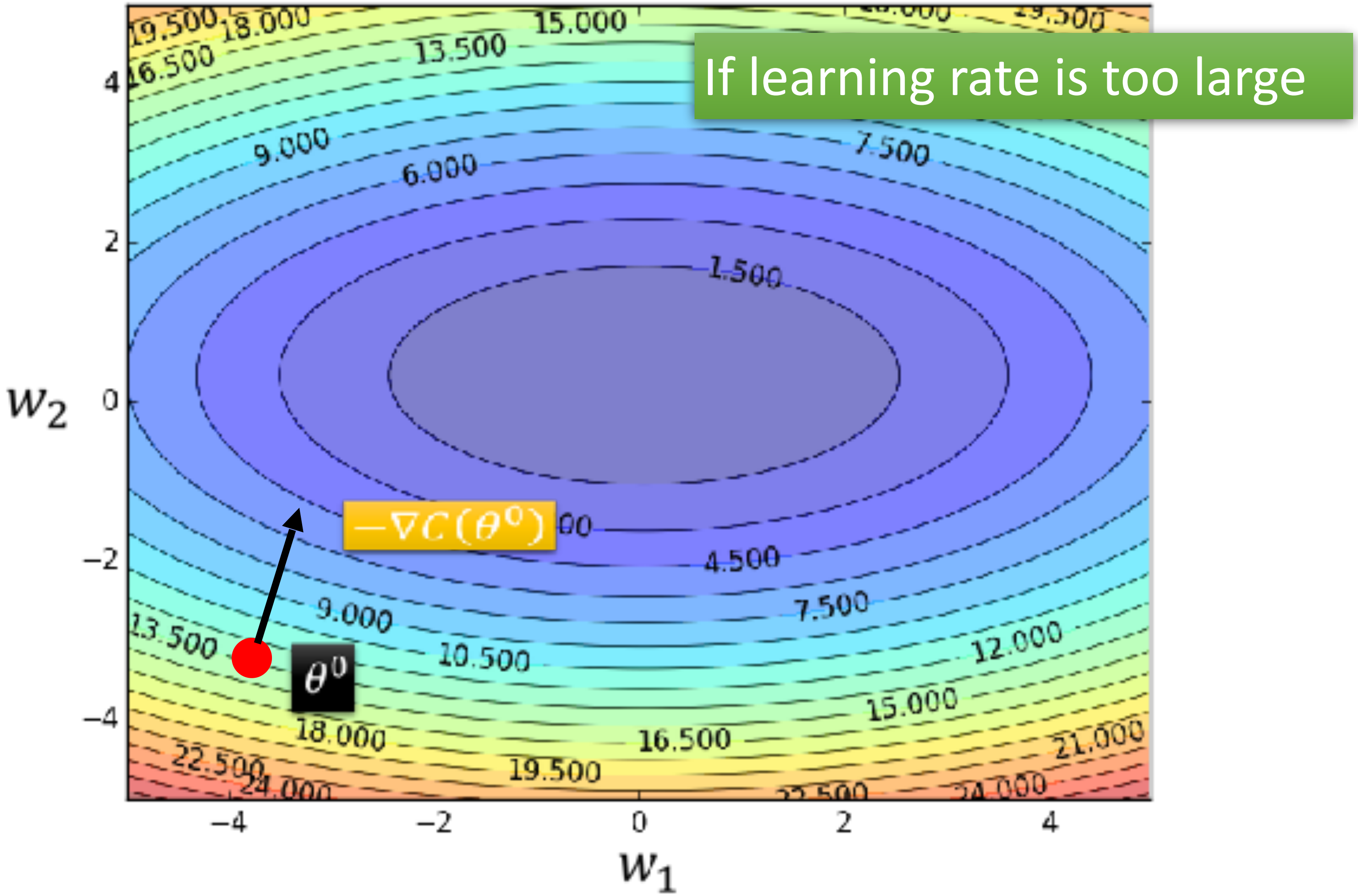
Learning Rate

Set the learning rate η carefully



Learning Rate

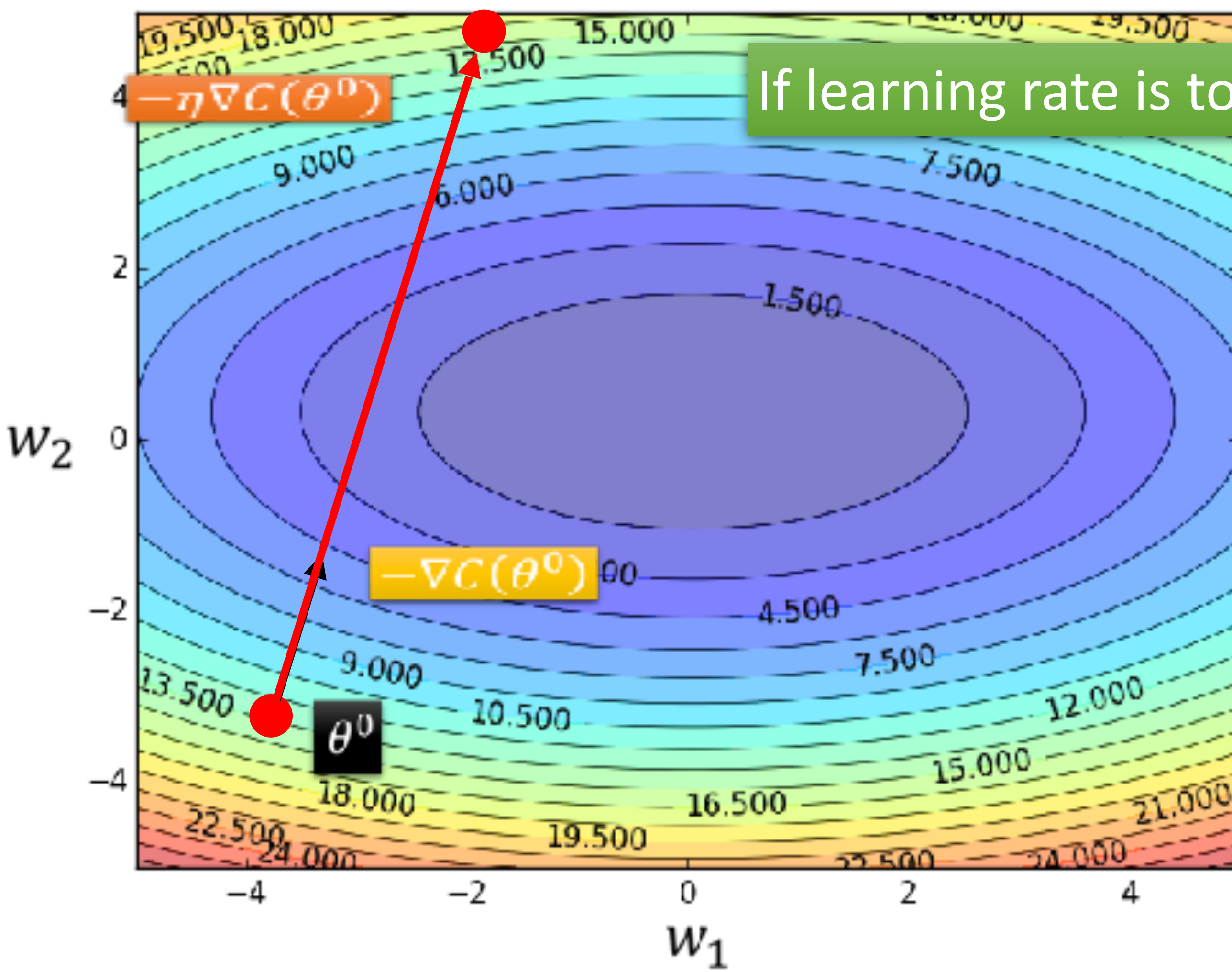
Set the learning rate η carefully



Learning Rate

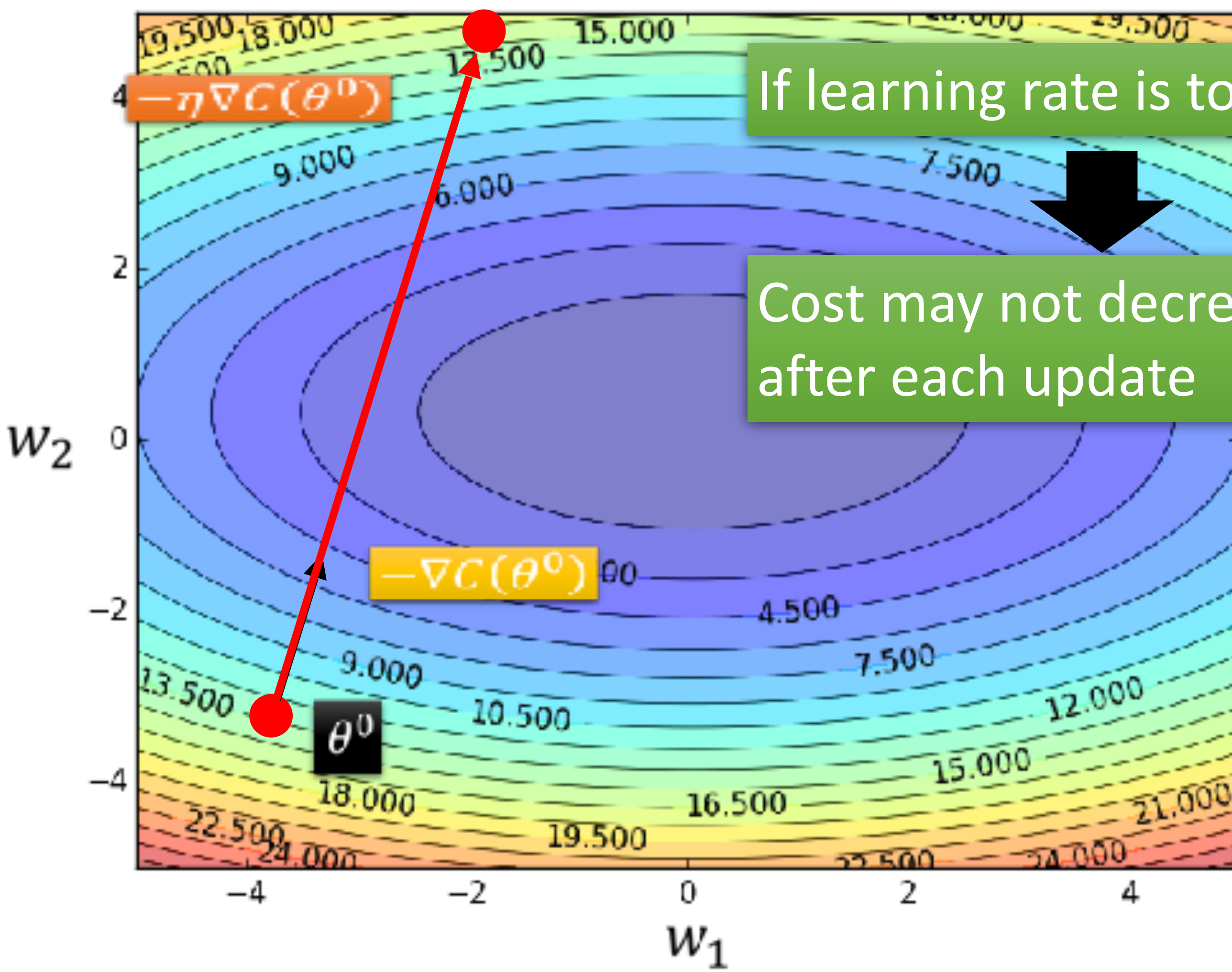
Set the learning rate η carefully

If learning rate is too large



Learning Rate

Set the learning rate η carefully

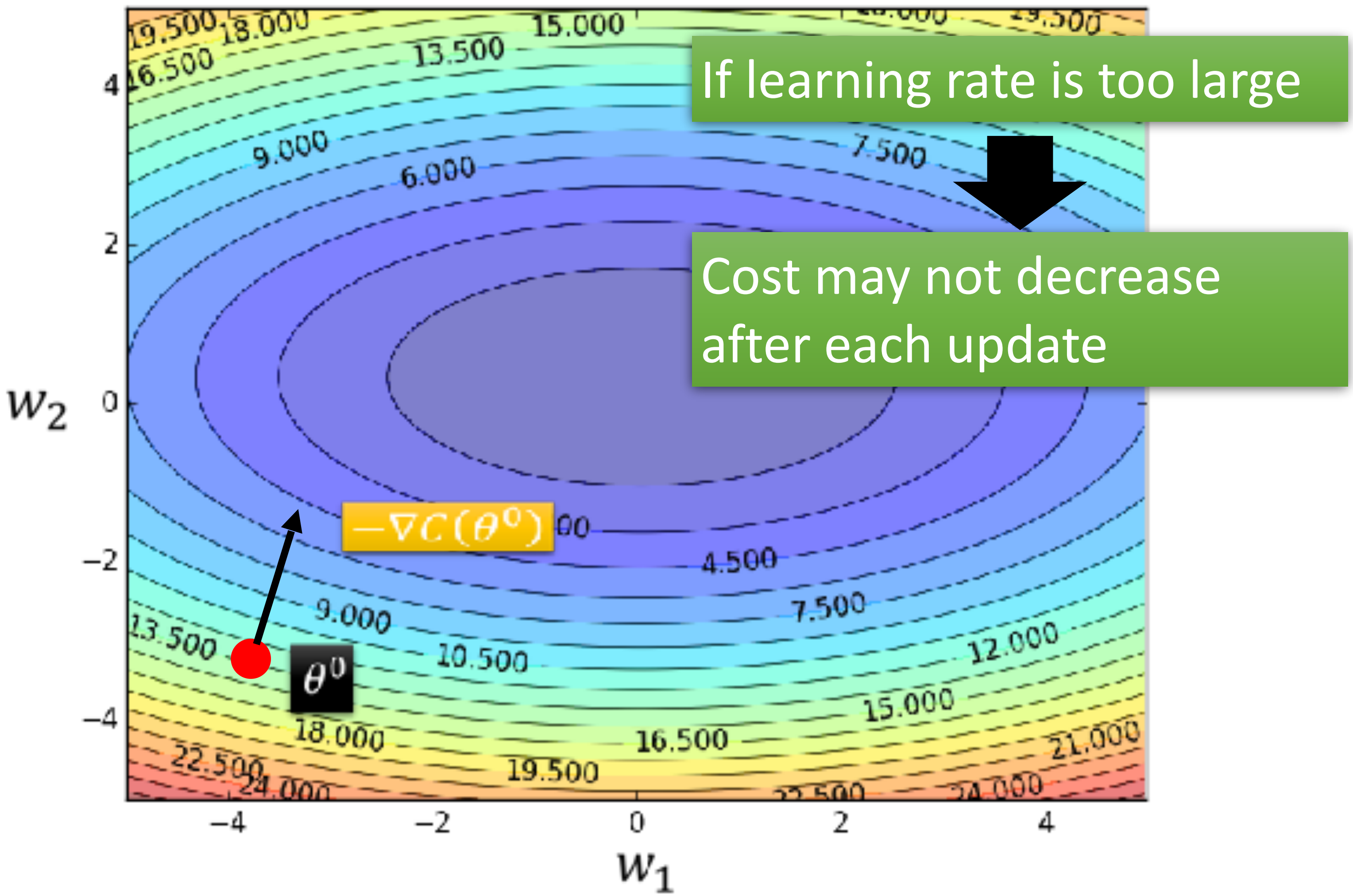


If learning rate is too large

Cost may not decrease after each update

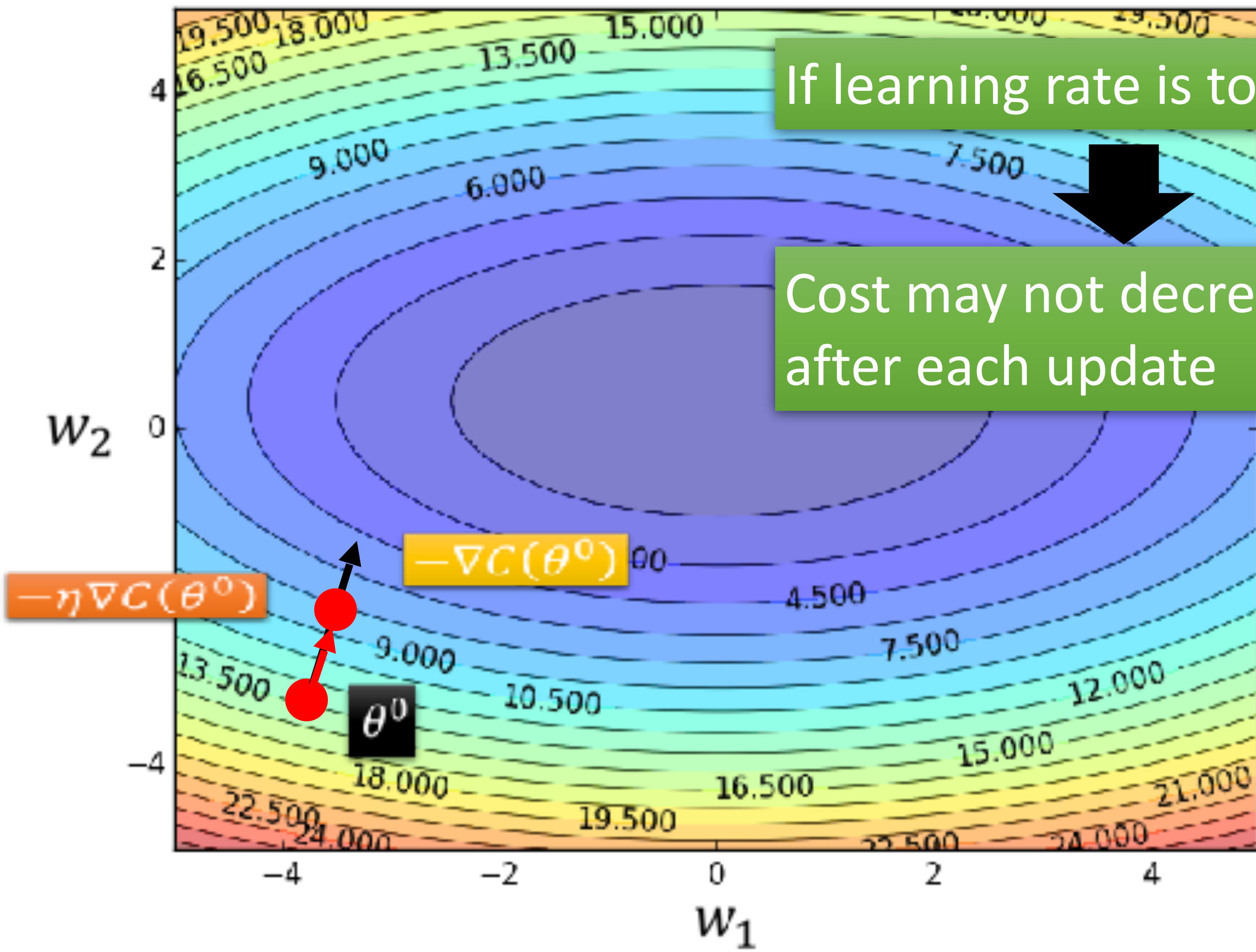
Learning Rate

Set the learning rate η carefully



Learning Rate

Set the learning rate η carefully

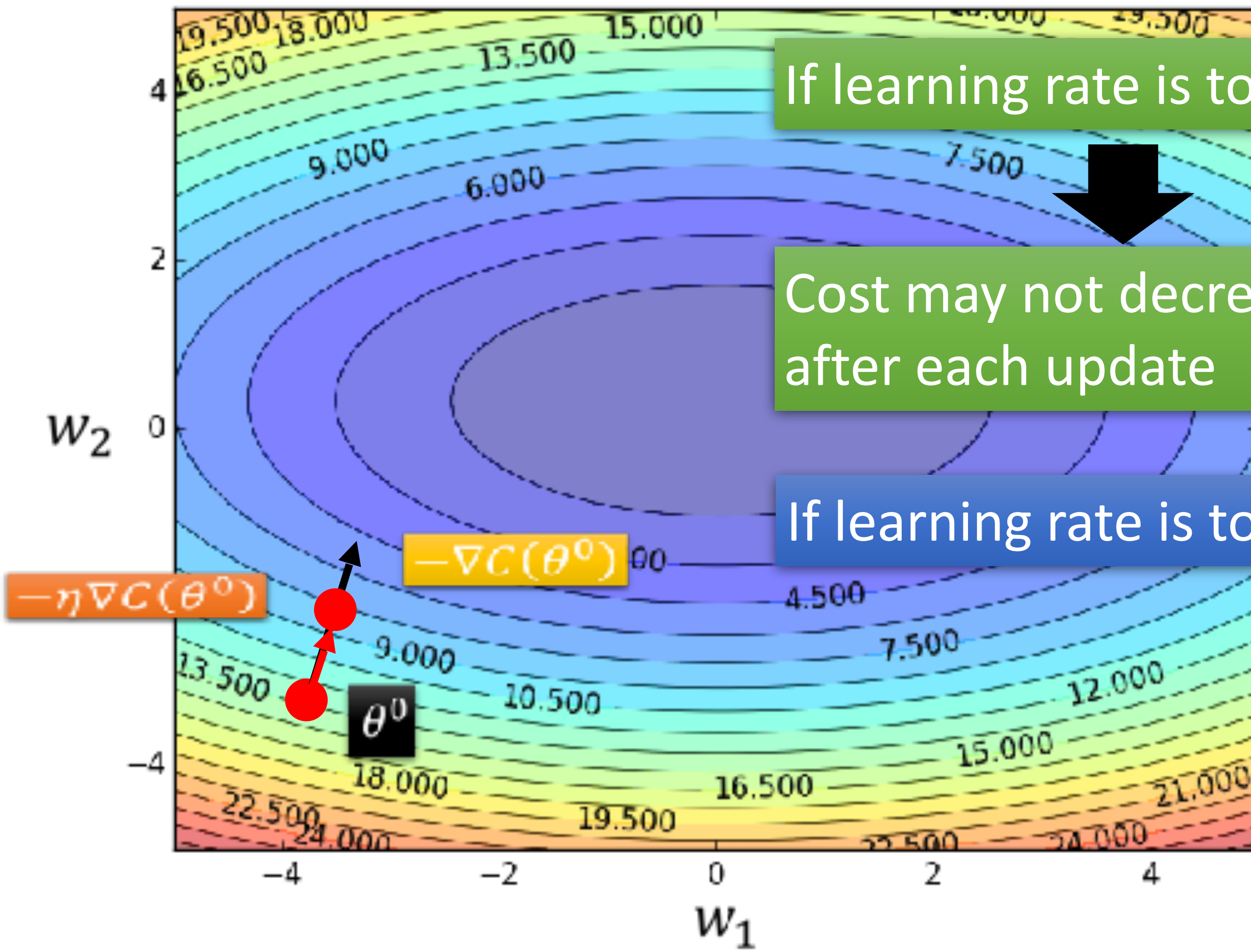


If learning rate is too large

Cost may not decrease after each update

Learning Rate

Set the learning rate η carefully



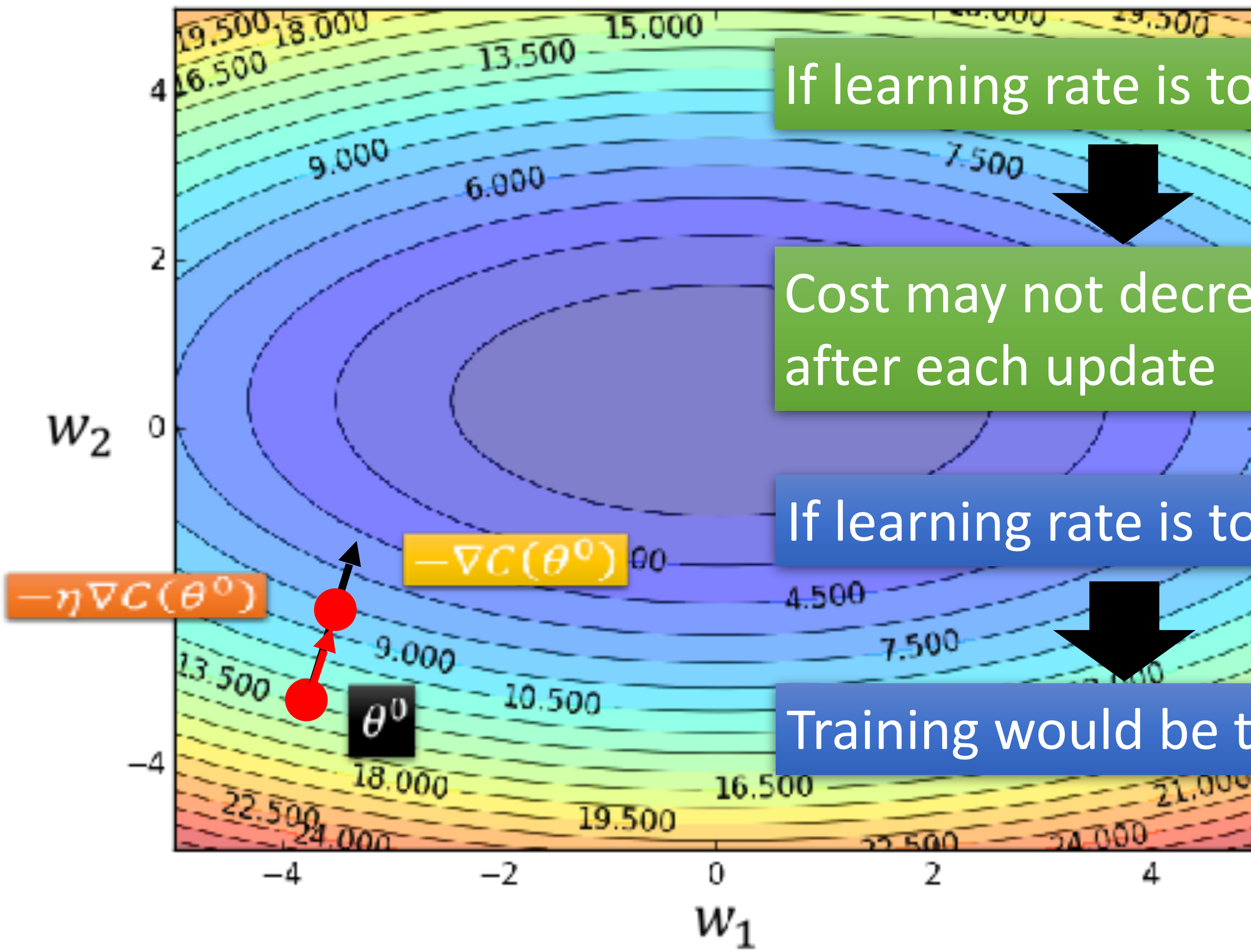
If learning rate is too large

Cost may not decrease after each update

If learning rate is too small

Learning Rate

Set the learning rate η carefully



If learning rate is too large

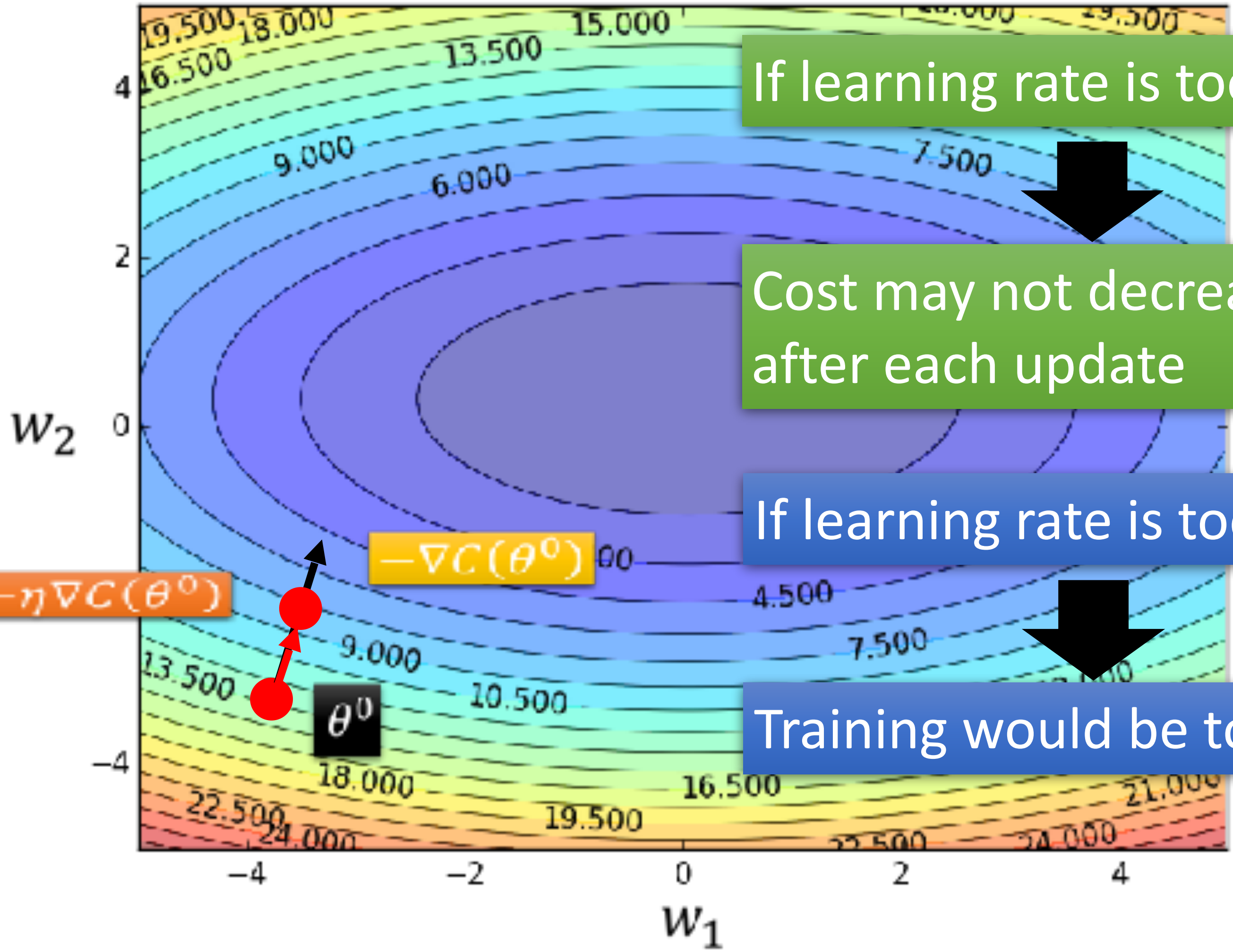
Cost may not decrease after each update

If learning rate is too small

Training would be too slow

Learning Rate

Can we give different parameters different learning rates?



If learning rate is too large

Cost may not decrease after each update

If learning rate is too small

Training would be too slow

Adagrad

Original Gradient Descent

$$\theta^t \leftarrow \theta^{t-1} - \eta \nabla C(\theta^{t-1})$$

Adagrad

Original Gradient Descent

$$\theta^t \leftarrow \theta^{t-1} - \eta \nabla C(\theta^{t-1})$$

Each parameter w are considered separately

$$w^{t+1} \leftarrow w^t - \eta_w g^t$$

Adagrad

Original Gradient Descent

$$\theta^t \leftarrow \theta^{t-1} - \eta \nabla C(\theta^{t-1})$$

Each parameter w are considered separately

$$w^{t+1} \leftarrow w^t - \eta_w \underline{g^t}$$

Adagrad

Original Gradient Descent

$$\theta^t \leftarrow \theta^{t-1} - \eta \nabla C(\theta^{t-1})$$

Each parameter w are considered separately

$$w^{t+1} \leftarrow w^t - \eta_w g^t \qquad g^t = \frac{\partial C(\theta^t)}{\partial w}$$

Adagrad

Original Gradient Descent

$$\theta^t \leftarrow \theta^{t-1} - \eta \nabla C(\theta^{t-1})$$

Each parameter w are considered separately

$$w^{t+1} \leftarrow w^t - \boxed{\eta_w} \underline{g^t} \qquad \underline{g^t} = \frac{\partial C(\theta^t)}{\partial w}$$

Parameter dependent learning rate

Adagrad

Original Gradient Descent

$$\theta^t \leftarrow \theta^{t-1} - \eta \nabla C(\theta^{t-1})$$

Each parameter w are considered separately

$$w^{t+1} \leftarrow w^t - \boxed{\eta_w} \underline{g^t} \qquad \underline{g^t} = \frac{\partial C(\theta^t)}{\partial w}$$

Parameter dependent learning rate

$$\eta_w = \frac{\eta}{\sqrt{\sum_{i=0}^{t} (g^i)^2}}$$

Adagrad

Original Gradient Descent

$$\theta^t \leftarrow \theta^{t-1} - \eta \nabla C(\theta^{t-1})$$

Each parameter w are considered separately

$$w^{t+1} \leftarrow w^t - \boxed{\eta_w} \underline{g^t} \qquad \underline{g^t} = \frac{\partial C(\theta^t)}{\partial w}$$

Parameter dependent learning rate

$$\eta_w = \frac{\eta}{\sqrt{\sum_{i=0}^{t} (g^i)^2}}$$

constant

Adagrad

Original Gradient Descent

$$\theta^t \leftarrow \theta^{t-1} - \eta \nabla C(\theta^{t-1})$$

Each parameter w are considered separately

$$w^{t+1} \leftarrow w^t - \boxed{\eta_w} \underline{g^t} \qquad \underline{g^t} = \frac{\partial C(\theta^t)}{\partial w}$$

Parameter dependent learning rate

$$\eta_w = \frac{\eta}{\sqrt{\boxed{\sum_{i=0}^{t} (g^i)^2}}}$$

constant

Summation of the square of the previous derivatives

Adagrad

$$\eta_w = \boxed{\frac{\eta}{\sqrt{\sum_{i=0}^{t}(g^i)^2}}}$$

Adagrad

$$\eta_w = \boxed{\frac{\eta}{\sqrt{\sum_{i=0}^{t}(g^i)^2}}}$$

	g^0		
w_1	0.1		

Adagrad

$$\eta_w = \boxed{\frac{\eta}{\sqrt{\sum_{i=0}^{t}(g^i)^2}}}$$

	g^0		
w_1	0.1		

Learning rate:

$$\frac{\eta}{\sqrt{0.1^2}}$$

Adagrad

$$\eta_w = \boxed{\frac{\eta}{\sqrt{\sum_{i=0}^{t}(g^i)^2}}}$$

	g^0
w_1	0.1

Learning rate:

$$\frac{\eta}{\sqrt{0.1^2}} = \frac{\eta}{0.1}$$

Adagrad

$$\eta_w = \boxed{\frac{\eta}{\sqrt{\sum_{i=0}^{t}(g^i)^2}}}$$

	g^0	g^1	……
w_1	0.1	0.2	……

Learning rate:

$$\frac{\eta}{\sqrt{0.1^2}} \qquad = \frac{\eta}{0.1}$$

Adagrad

$$\eta_w = \boxed{\frac{\eta}{\sqrt{\sum_{i=0}^{t}(g^i)^2}}}$$

w_1	g^0	g^1	……
	0.1	0.2	……

Learning rate:

$$\frac{\eta}{\sqrt{0.1^2}} = \frac{\eta}{0.1}$$

$$\frac{\eta}{\sqrt{0.1^2 + 0.2^2}}$$

Adagrad

$$\eta_w = \boxed{\frac{\eta}{\sqrt{\sum_{i=0}^{t}(g^i)^2}}}$$

	g^0	g^1	……
w_1	0.1	0.2	……

Learning rate:

$$\frac{\eta}{\sqrt{0.1^2}} = \frac{\eta}{0.1}$$

$$\frac{\eta}{\sqrt{0.1^2 + 0.2^2}} = \frac{\eta}{0.22}$$

Adagrad

$$\eta_w = \boxed{\frac{\eta}{\sqrt{\sum_{i=0}^{t}(g^i)^2}}}$$

w_1

g^0	g^1	……
0.1	0.2	……

w_2

g^0
20.0

Learning rate:

$$\frac{\eta}{\sqrt{0.1^2}} = \frac{\eta}{0.1}$$

$$\frac{\eta}{\sqrt{0.1^2 + 0.2^2}} = \frac{\eta}{0.22}$$

Adagrad

$$\eta_w = \boxed{\frac{\eta}{\sqrt{\sum_{i=0}^{t}(g^i)^2}}}$$

w_1

g^0	g^1	……
0.1	0.2	……

Learning rate:

$$\frac{\eta}{\sqrt{0.1^2}} = \frac{\eta}{0.1}$$

$$\frac{\eta}{\sqrt{0.1^2 + 0.2^2}} = \frac{\eta}{0.22}$$

w_2

g^0
20.0

Learning rate:

$$\frac{\eta}{\sqrt{20^2}}$$

Adagrad

$$\eta_w = \boxed{\frac{\eta}{\sqrt{\sum_{i=0}^{t}(g^i)^2}}}$$

w_1

g^0	g^1	……
0.1	0.2	……

Learning rate:

$$\frac{\eta}{\sqrt{0.1^2}} = \frac{\eta}{0.1}$$

$$\frac{\eta}{\sqrt{0.1^2 + 0.2^2}} = \frac{\eta}{0.22}$$

w_2

g^0
20.0

Learning rate:

$$\frac{\eta}{\sqrt{20^2}} = \frac{\eta}{20}$$

Adagrad

$$\eta_w = \boxed{\frac{\eta}{\sqrt{\sum_{i=0}^{t}(g^i)^2}}}$$

w_1

g^0	g^1	……
0.1	0.2	……

Learning rate:

$$\frac{\eta}{\sqrt{0.1^2}} = \frac{\eta}{0.1}$$

$$\frac{\eta}{\sqrt{0.1^2 + 0.2^2}} = \frac{\eta}{0.22}$$

w_2

g^0	g^1	……
20.0	10.0	……

Learning rate:

$$\frac{\eta}{\sqrt{20^2}} = \frac{\eta}{20}$$

Adagrad

$$\eta_w = \boxed{\frac{\eta}{\sqrt{\sum_{i=0}^{t}(g^i)^2}}}$$

w_1

g^0	g^1	……
0.1	0.2	……

Learning rate:

$$\frac{\eta}{\sqrt{0.1^2}} = \frac{\eta}{0.1}$$

$$\frac{\eta}{\sqrt{0.1^2 + 0.2^2}} = \frac{\eta}{0.22}$$

w_2

g^0	g^1	……
20.0	10.0	……

Learning rate:

$$\frac{\eta}{\sqrt{20^2}} = \frac{\eta}{20}$$

$$\frac{\eta}{\sqrt{20^2 + 10^2}}$$

Adagrad

$$\eta_w = \boxed{\frac{\eta}{\sqrt{\sum_{i=0}^{t}(g^i)^2}}}$$

w_1

g^0	g^1	……
0.1	0.2	……

Learning rate:

$$\frac{\eta}{\sqrt{0.1^2}} = \frac{\eta}{0.1}$$

$$\frac{\eta}{\sqrt{0.1^2 + 0.2^2}} = \frac{\eta}{0.22}$$

w_2

g^0	g^1	……
20.0	10.0	……

Learning rate:

$$\frac{\eta}{\sqrt{20^2}} = \frac{\eta}{20}$$

$$\frac{\eta}{\sqrt{20^2 + 10^2}} = \frac{\eta}{22}$$

Adagrad

$$\eta_w = \boxed{\frac{\eta}{\sqrt{\sum_{i=0}^{t}(g^i)^2}}}$$

w_1

g^0	g^1	……
0.1	0.2	……

Learning rate:

$$\frac{\eta}{\sqrt{0.1^2}} = \frac{\eta}{0.1}$$

$$\frac{\eta}{\sqrt{0.1^2 + 0.2^2}} = \frac{\eta}{0.22}$$

w_2

g^0	g^1	……
20.0	10.0	……

Learning rate:

$$\frac{\eta}{\sqrt{20^2}} = \frac{\eta}{20}$$

$$\frac{\eta}{\sqrt{20^2 + 10^2}} = \frac{\eta}{22}$$

<u>Observation:</u>

Adagrad

$$\eta_w = \boxed{\frac{\eta}{\sqrt{\sum_{i=0}^{t}(g^i)^2}}}$$

w_1

g^0	g^1	……
0.1	0.2	……

Learning rate:

$$\frac{\eta}{\sqrt{0.1^2}} = \frac{\eta}{0.1}$$

$$\frac{\eta}{\sqrt{0.1^2 + 0.2^2}} = \frac{\eta}{0.22}$$

w_2

g^0	g^1	……
20.0	10.0	……

Learning rate:

$$\frac{\eta}{\sqrt{20^2}} = \frac{\eta}{20}$$

$$\frac{\eta}{\sqrt{20^2 + 10^2}} = \frac{\eta}{22}$$

Observation: 1. Learning rate is smaller and smaller for all parameters

Adagrad

$$\eta_w = \boxed{\frac{\eta}{\sqrt{\sum_{i=0}^{t}(g^i)^2}}}$$

w_1

g^0	g^1	……
0.1	0.2	……

w_2

g^0	g^1	……
20.0	10.0	……

Learning rate:

$$\frac{\eta}{\sqrt{0.1^2}} = \frac{\eta}{0.1}$$

$$\frac{\eta}{\sqrt{0.1^2 + 0.2^2}} = \frac{\eta}{0.22}$$

Learning rate:

$$\frac{\eta}{\sqrt{20^2}} = \frac{\eta}{20}$$

$$\frac{\eta}{\sqrt{20^2 + 10^2}} = \frac{\eta}{22}$$

<u>Observation:</u> 1. Learning rate is smaller and smaller for all parameters

2. Smaller derivatives, larger learning rate, and vice versa

Adagrad

$$\eta_w = \boxed{\frac{\eta}{\sqrt{\sum_{i=0}^{t}(g^i)^2}}}$$

w_1

g^0	g^1	……
0.1	0.2	……

w_2

g^0	g^1	……
20.0	10.0	……

Learning rate:

$$\frac{\eta}{\sqrt{0.1^2}} = \frac{\eta}{0.1}$$

$$\frac{\eta}{\sqrt{0.1^2 + 0.2^2}} = \frac{\eta}{0.22}$$

Learning rate:

$$\frac{\eta}{\sqrt{20^2}} = \frac{\eta}{20}$$

$$\frac{\eta}{\sqrt{20^2 + 10^2}} = \frac{\eta}{22}$$

<u>Observation:</u> 1. Learning rate is smaller and smaller for all parameters

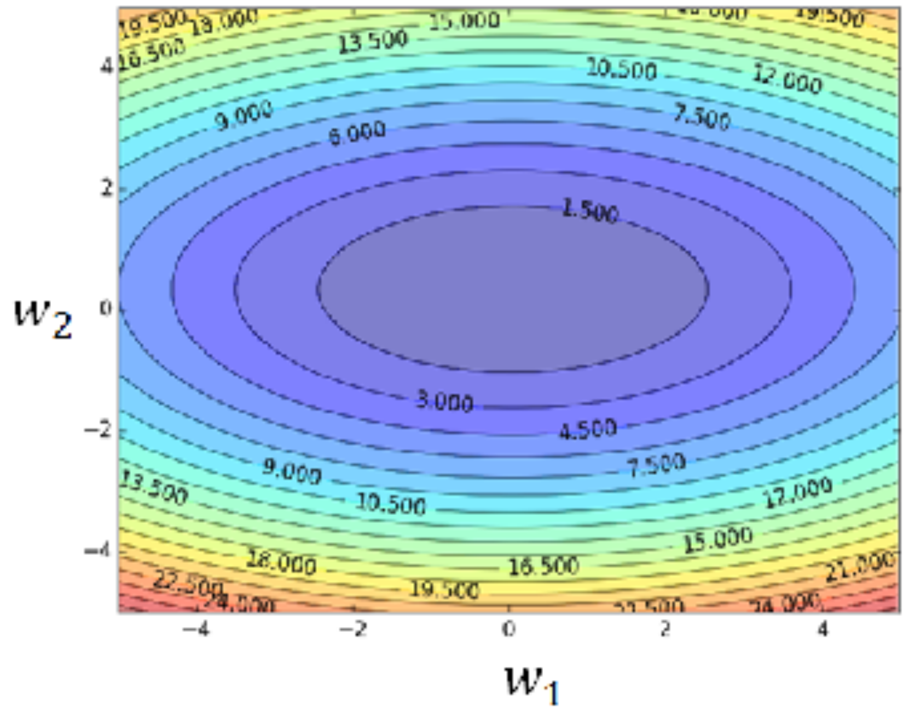
2. Smaller derivatives, larger le

Why?

rate, and vice versa

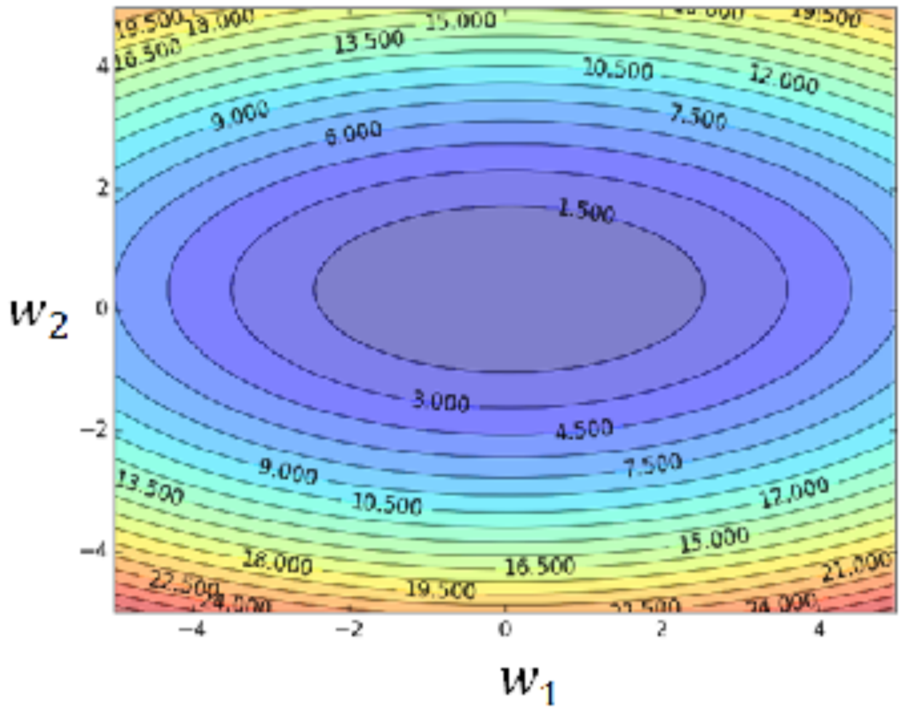
2. Smaller derivatives, larger le
rate, and vice versa

Why?



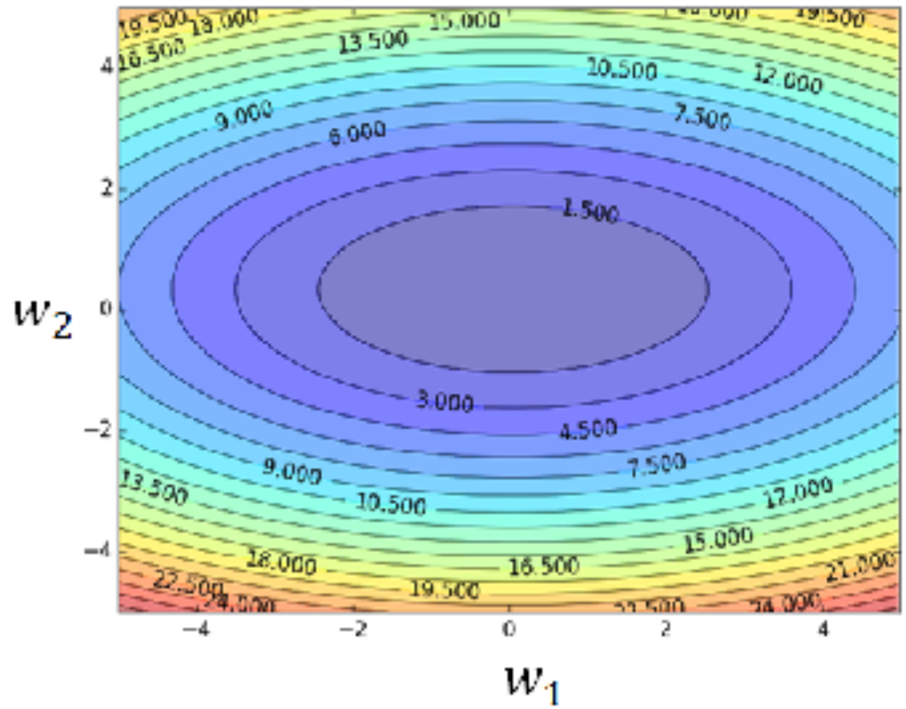
2. Smaller derivatives, larger learning rate, and vice versa

Why?



2. Smaller derivatives, larger le
rate, and vice versa

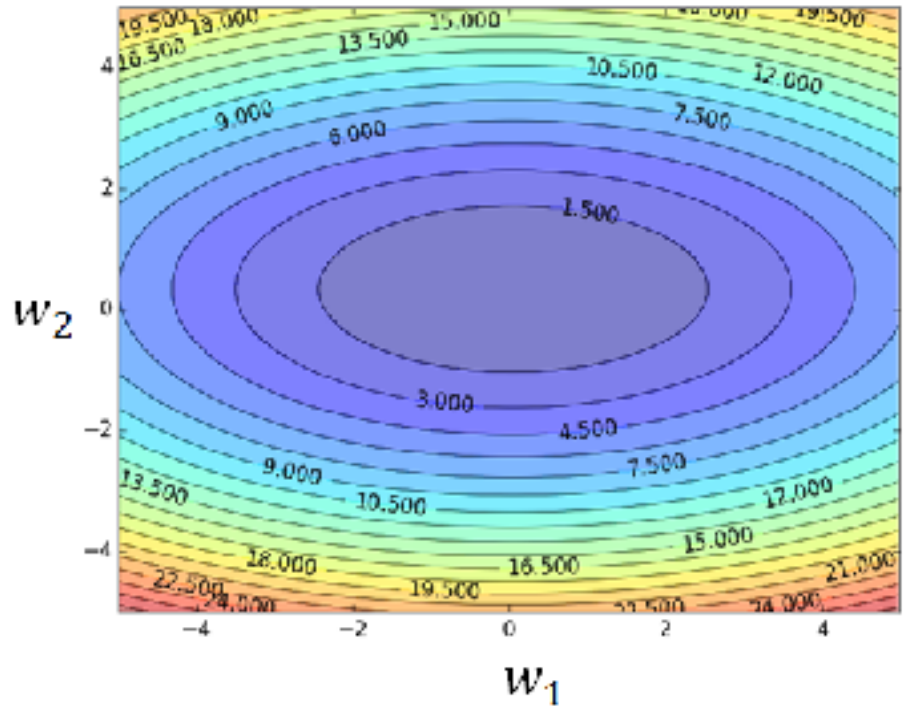
Why?



Smaller Derivatives

2. Smaller derivatives, larger le
rate, and vice versa

Why?



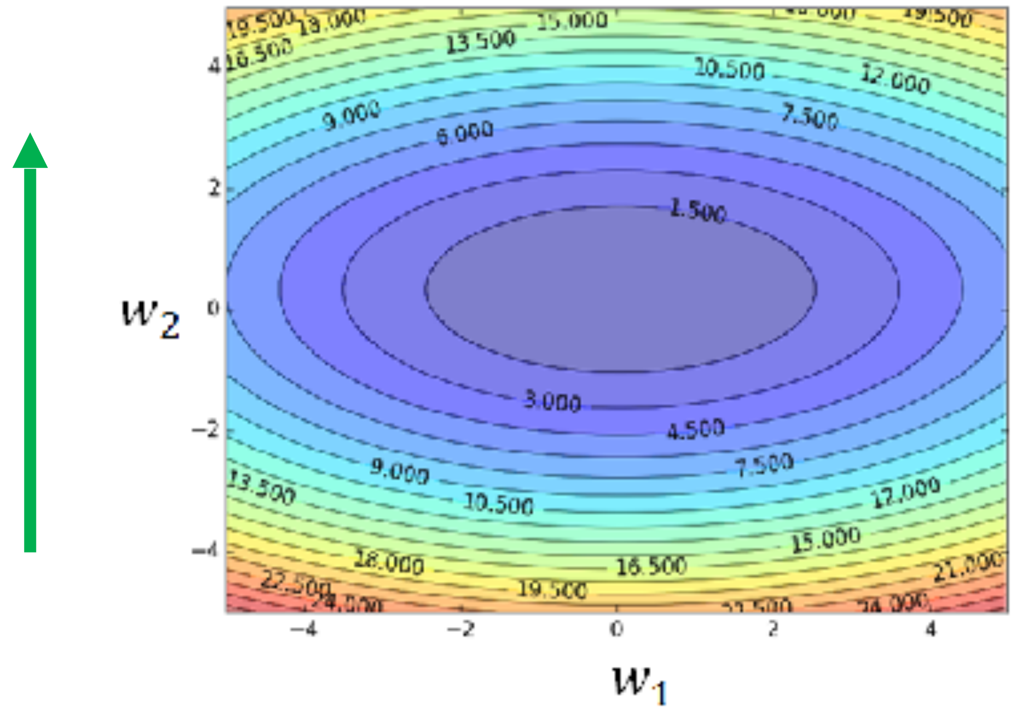
Smaller Derivatives

Larger Learning Rate

2. Smaller derivatives, larger le

Why?

rate, and vice versa



Smaller Derivatives

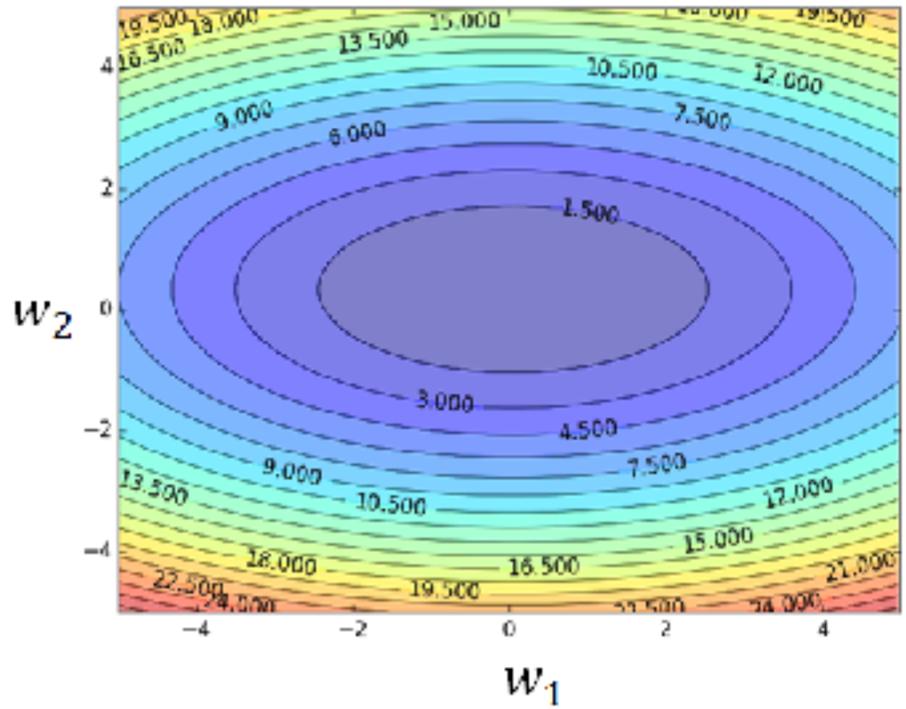
Larger Learning Rate

2. Smaller derivatives, larger le

Why?

rate, and vice versa

Larger derivatives



Smaller Derivatives

Larger Learning Rate

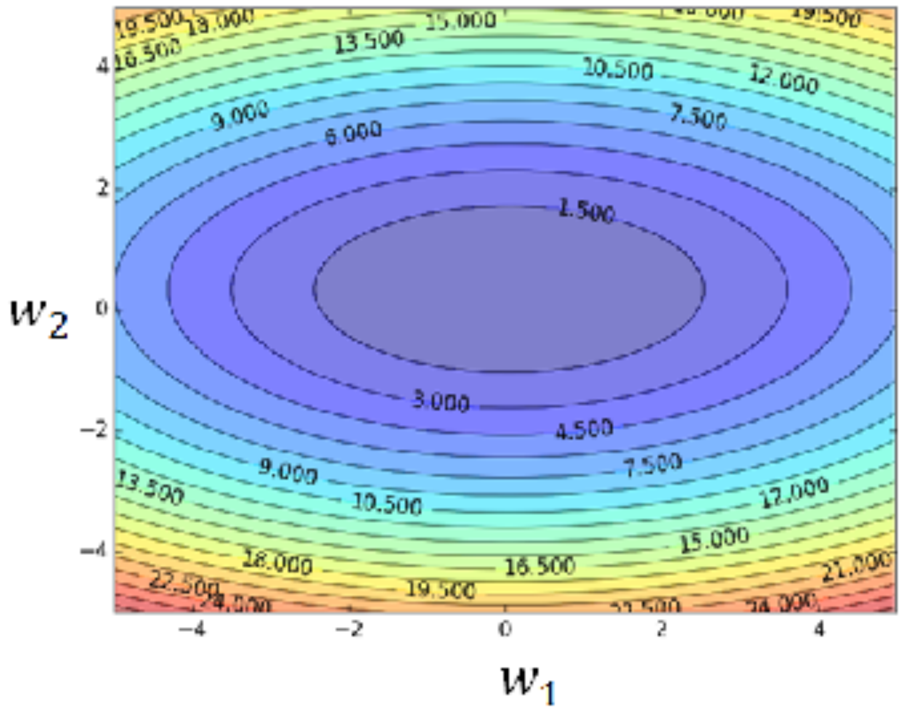
2. Smaller derivatives, larger le

Why?

rate, and vice versa

Larger derivatives

Smaller Learning Rate



Smaller Derivatives

Larger Learning Rate

2. Smaller derivatives, larger le
rate, and vice versa

Why?

Not the whole story ……

- Adagrad [John Duchi, JMLR'11]
- RMSprop
  - https://www.youtube.com/watch?v=O3sxAc4hxZU
- Adadelta [Matthew D. Zeiler, arXiv'12]
- Adam [Diederik P. Kingma, ICLR'15]
- AdaSecant [Caglar Gulcehre, arXiv'14]
- "No more pesky learning rates" [Tom Schaul, arXiv'12]

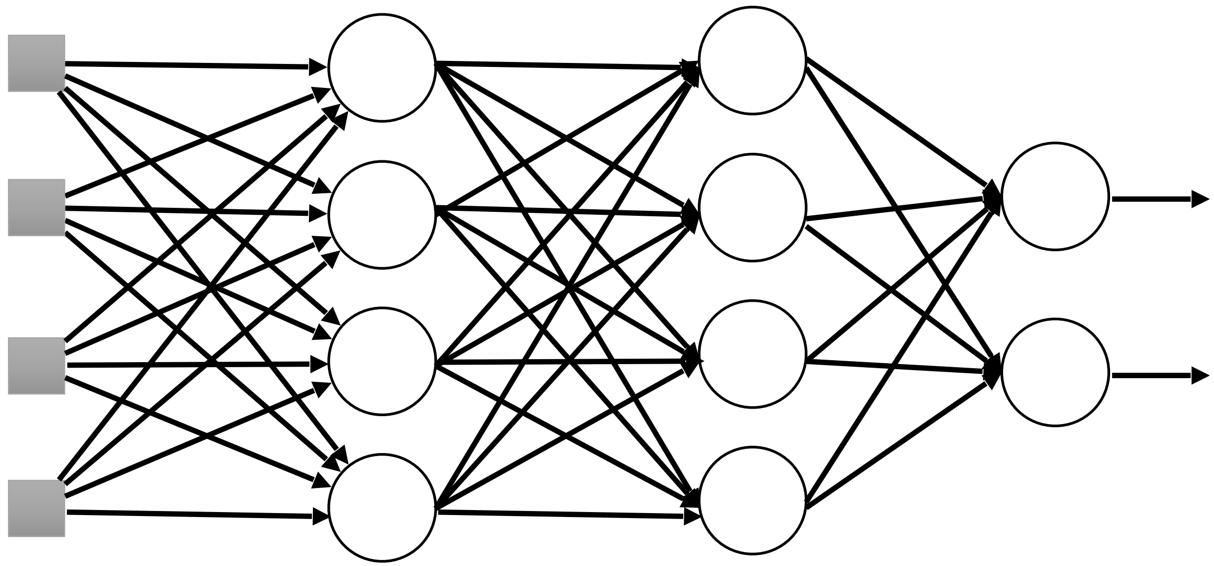
Part III:
Tips for Training DNN

Dropout

Dropout

Pick a mini-batch

$$\theta^t \leftarrow \theta^{t-1} - \eta \nabla C(\theta^{t-1})$$

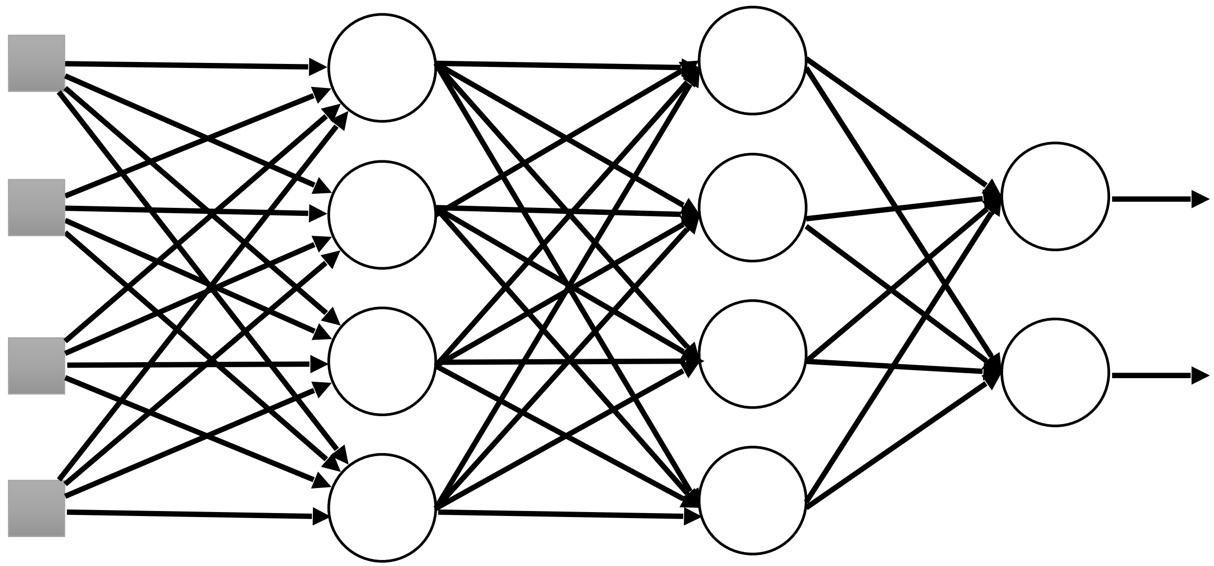


Dropout

Pick a mini-batch

$$\theta^t \leftarrow \theta^{t-1} - \eta \nabla C(\theta^{t-1})$$

Training:

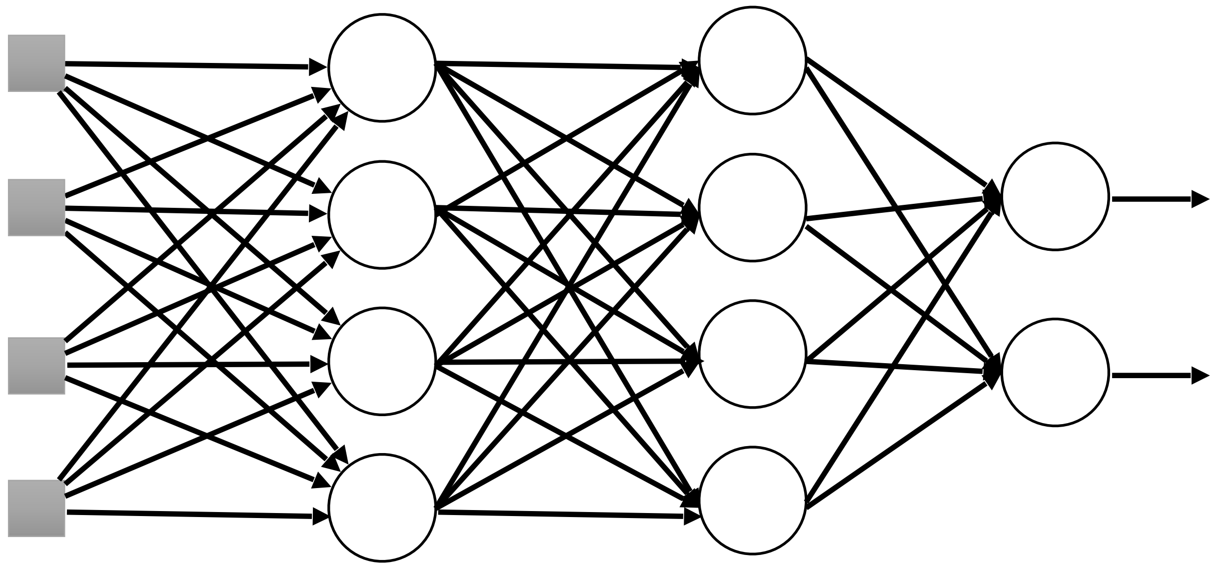


Dropout

Pick a mini-batch

$$\theta^t \leftarrow \theta^{t-1} - \eta \nabla C(\theta^{t-1})$$

Training:



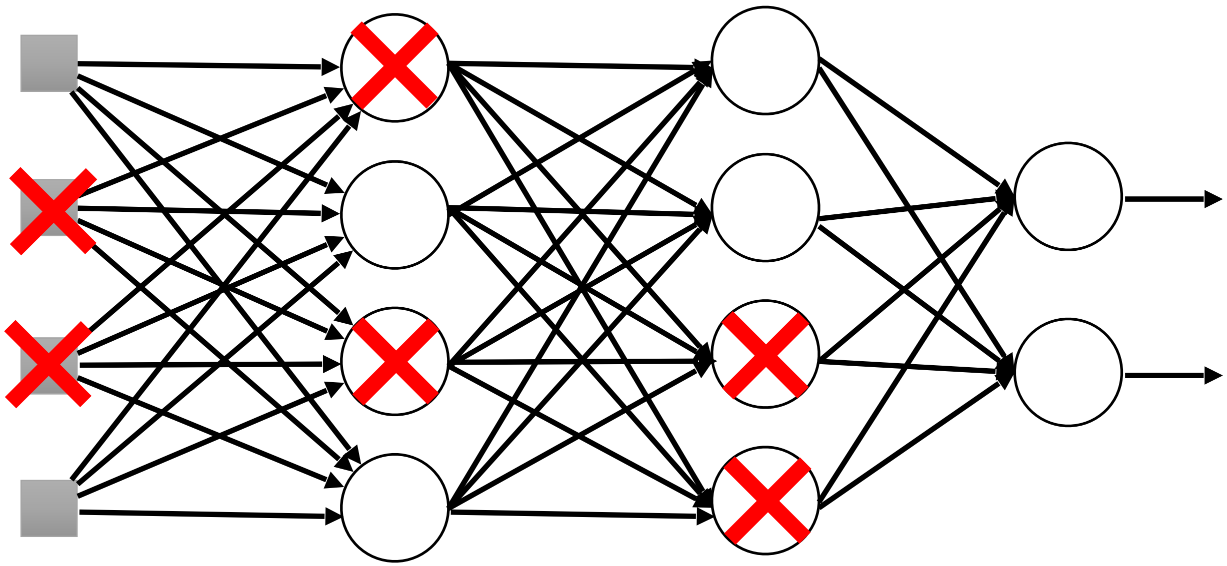
➢ **Each time before computing the gradients**

Dropout

Pick a mini-batch

$$\theta^t \leftarrow \theta^{t-1} - \eta \nabla C(\theta^{t-1})$$

Training:



➢ **Each time before computing the gradients**

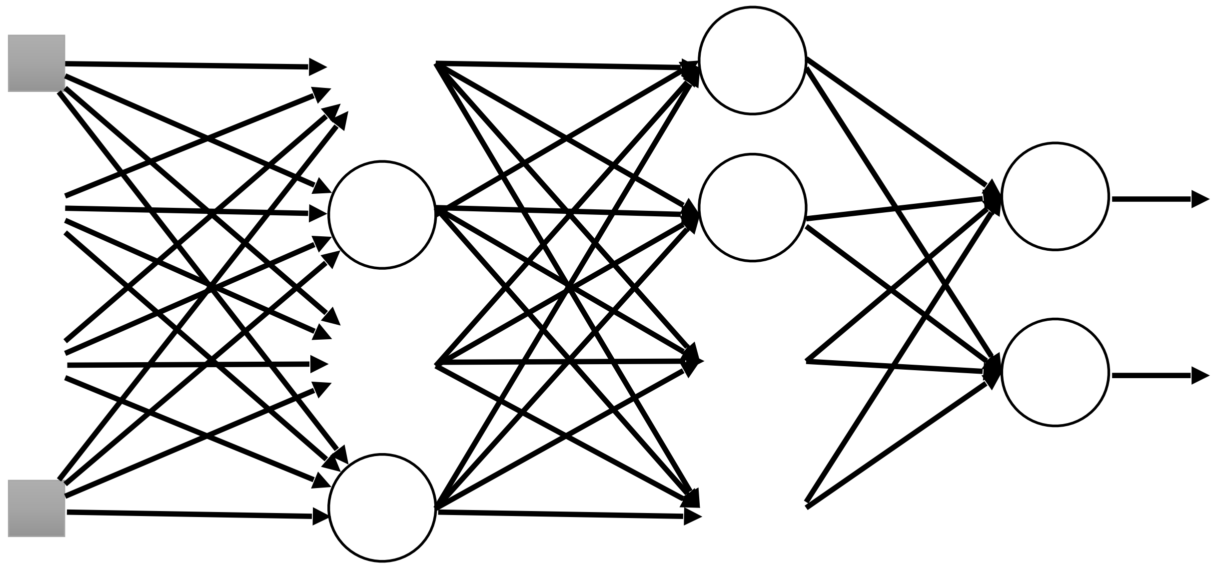
- Each neuron has p% to dropout

Dropout

Pick a mini-batch

$$\theta^t \leftarrow \theta^{t-1} - \eta \nabla C(\theta^{t-1})$$

Training:



➢ **Each time before computing the gradients**

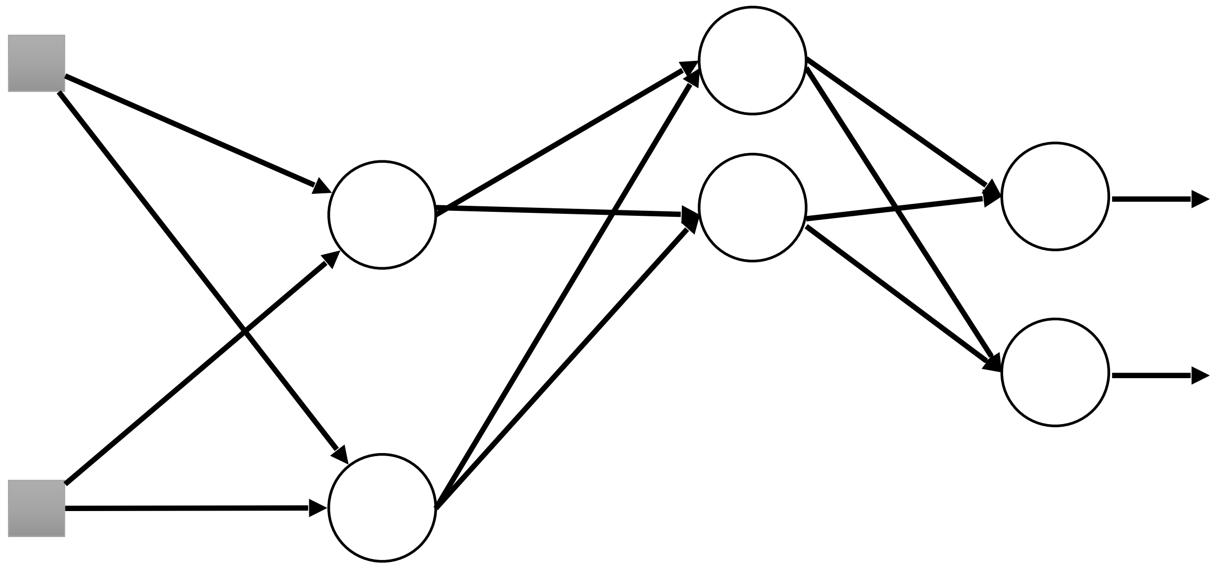
- Each neuron has p% to dropout

Dropout

Pick a mini-batch

$$\theta^t \leftarrow \theta^{t-1} - \eta \nabla C(\theta^{t-1})$$

Training:



➢ **Each time before computing the gradients**

- Each neuron has p% to dropout

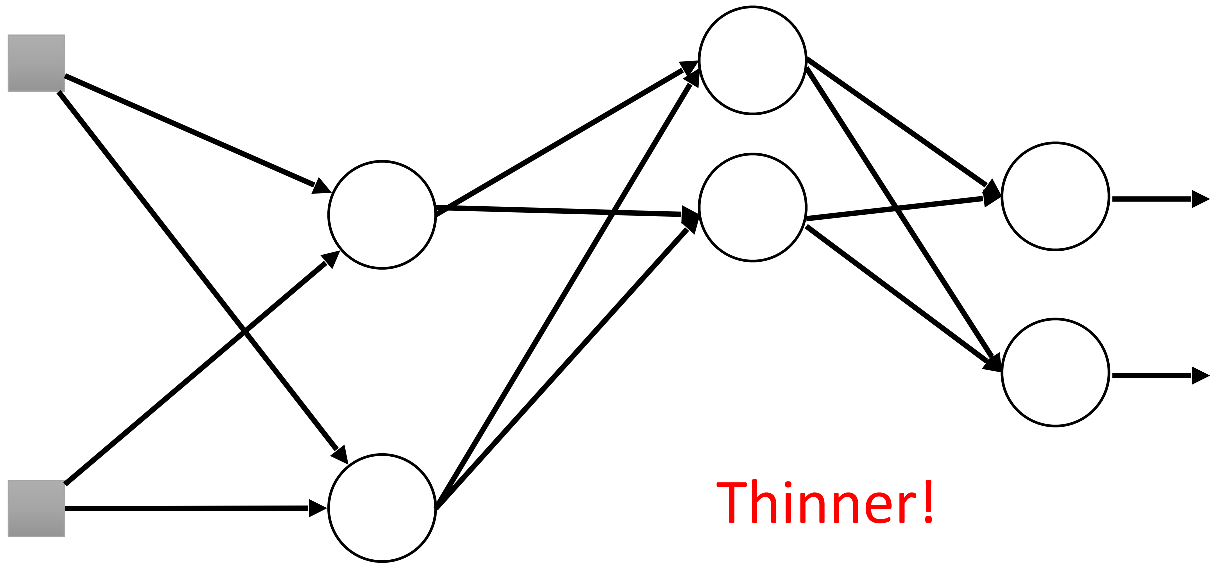
The structure of the network is changed.

Dropout

Pick a mini-batch

$$\theta^t \leftarrow \theta^{t-1} - \eta \nabla C(\theta^{t-1})$$

Training:



- ➢ **Each time before computing the gradients**
  - Each neuron has p% to dropout

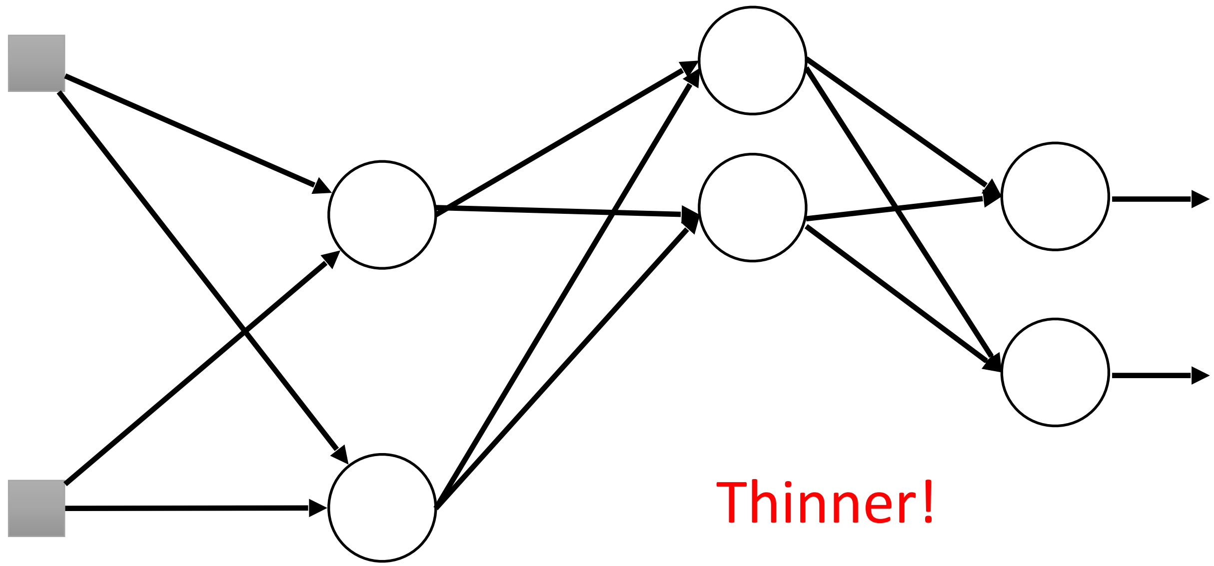
The structure of the network is changed.

Dropout

Pick a mini-batch

$$\theta^t \leftarrow \theta^{t-1} - \eta \nabla C(\theta^{t-1})$$

Training:



➢ **Each time before computing the gradients**

- Each neuron has p% to dropout

  ➡ **The structure of the network is changed.**

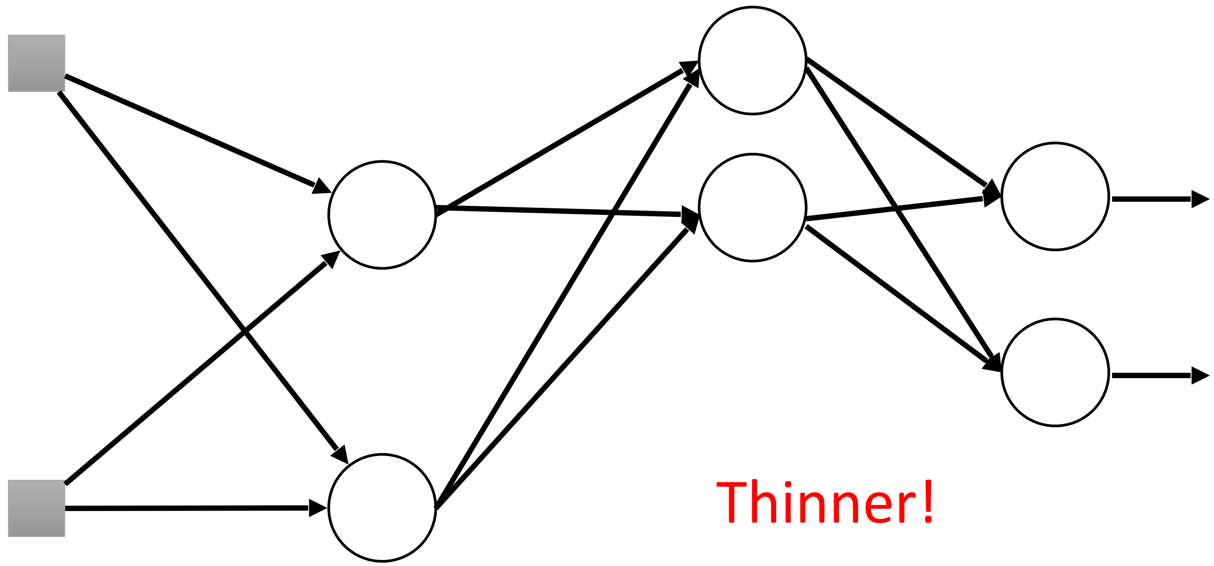
- Using the new network for training

Dropout

Pick a mini-batch

$$\theta^t \leftarrow \theta^{t-1} - \eta \nabla C(\theta^{t-1})$$

Training:



Thinner!

➢ **Each time before computing the gradients**

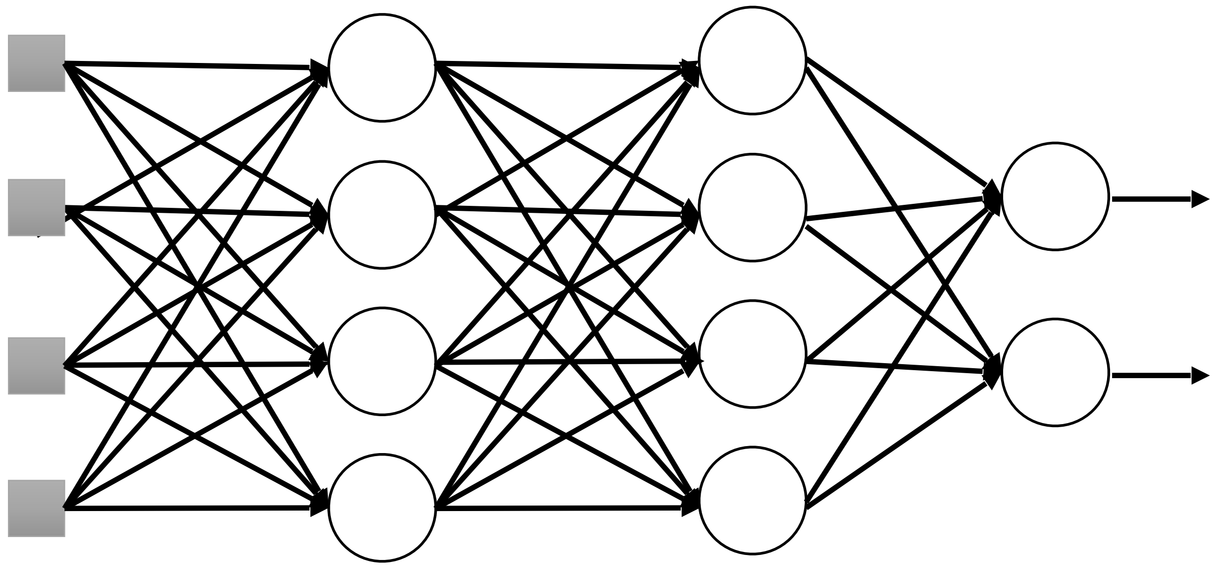
- Each neuron has p% to dropout

  The structure of the network is changed.

- Using the new network for training

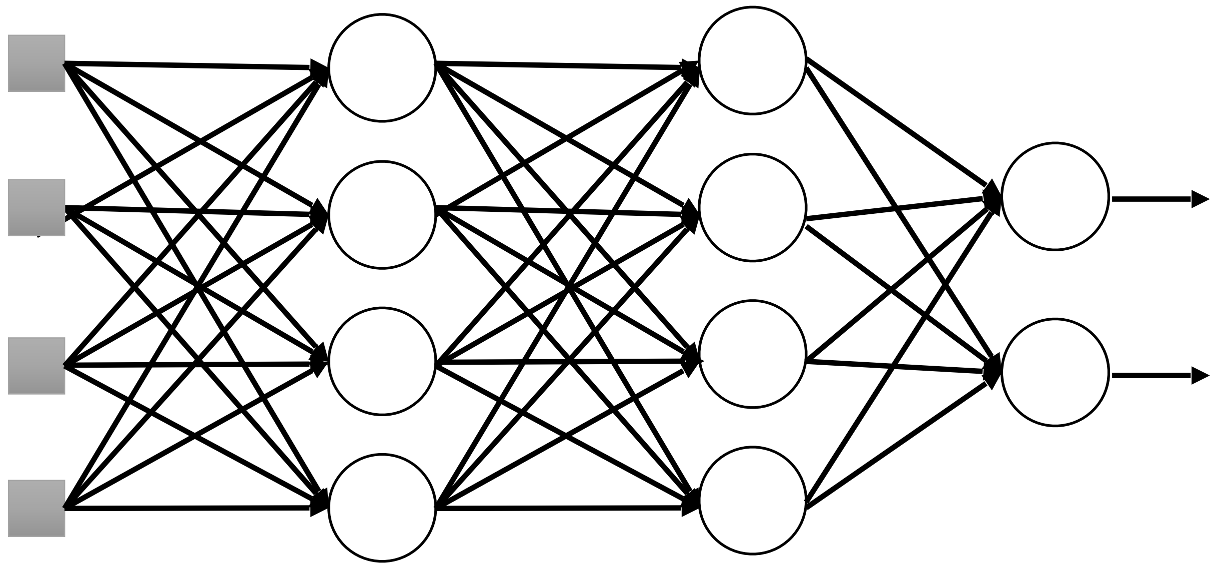
For each mini-batch, we resample the dropout neurons

Dropout



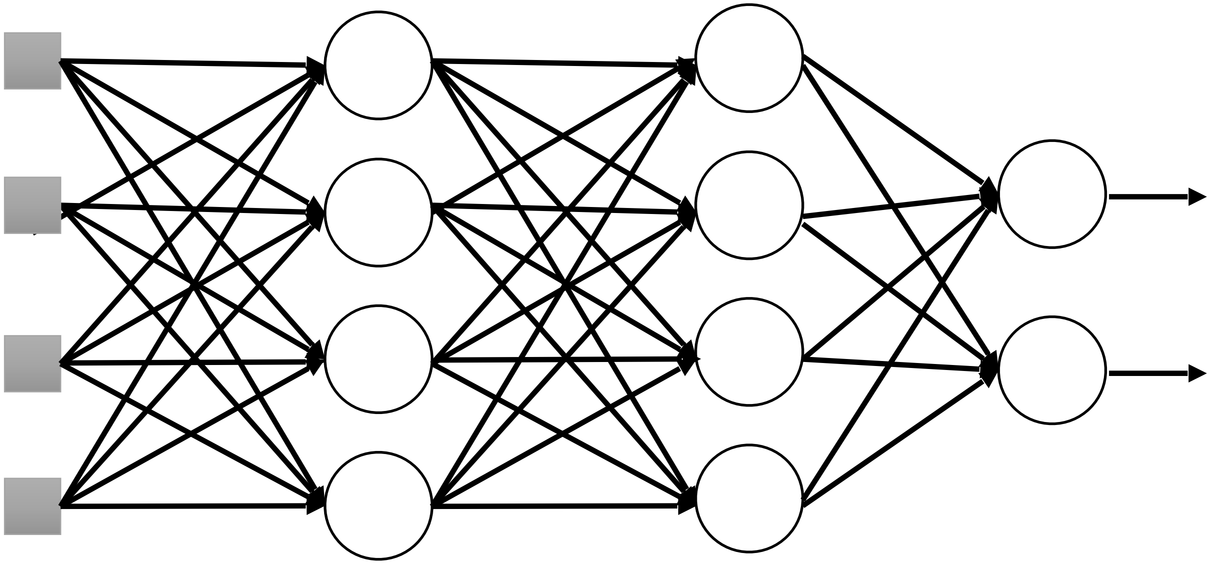
Dropout

Testing:



Dropout

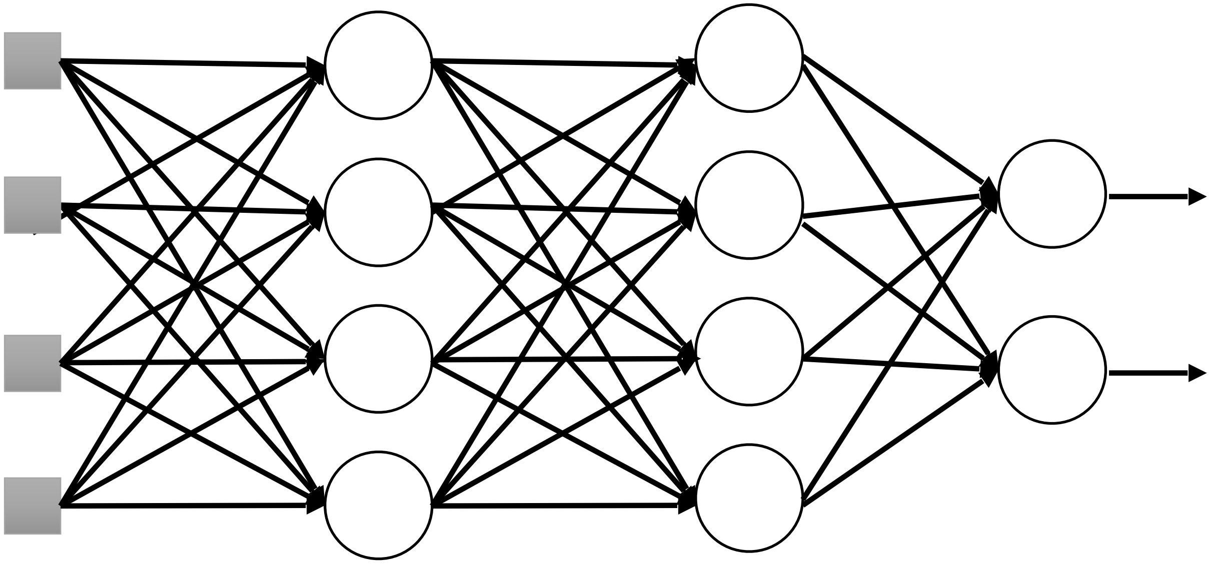
Testing:



➢ **No dropout**

Dropout

Testing:

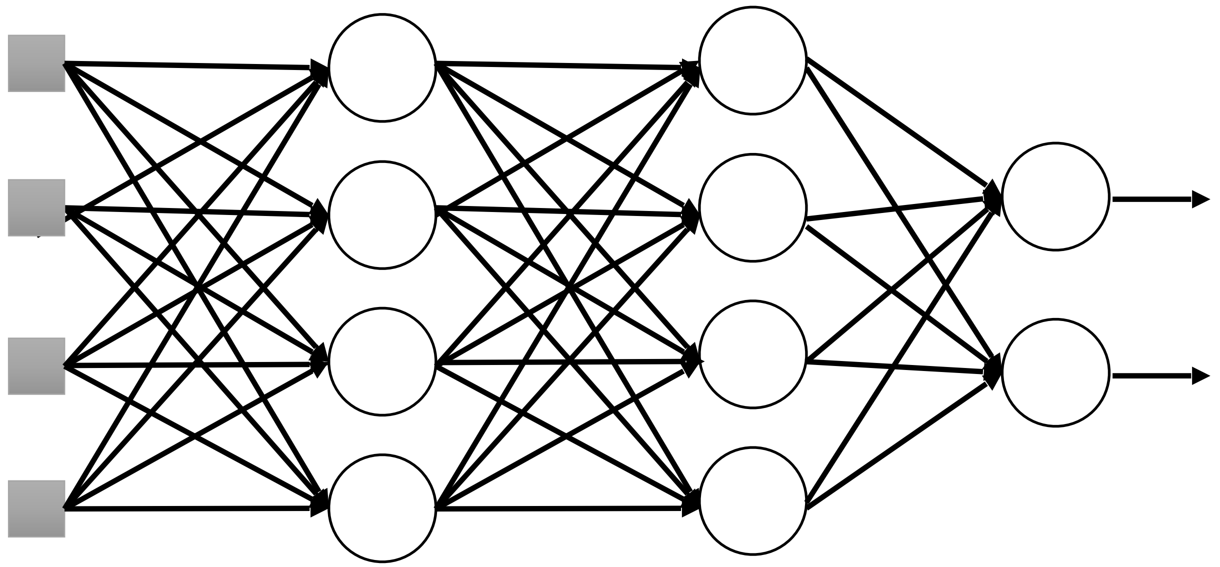


➢ **No dropout**

- If the dropout rate at training is p%, all the weights times (1-p)%

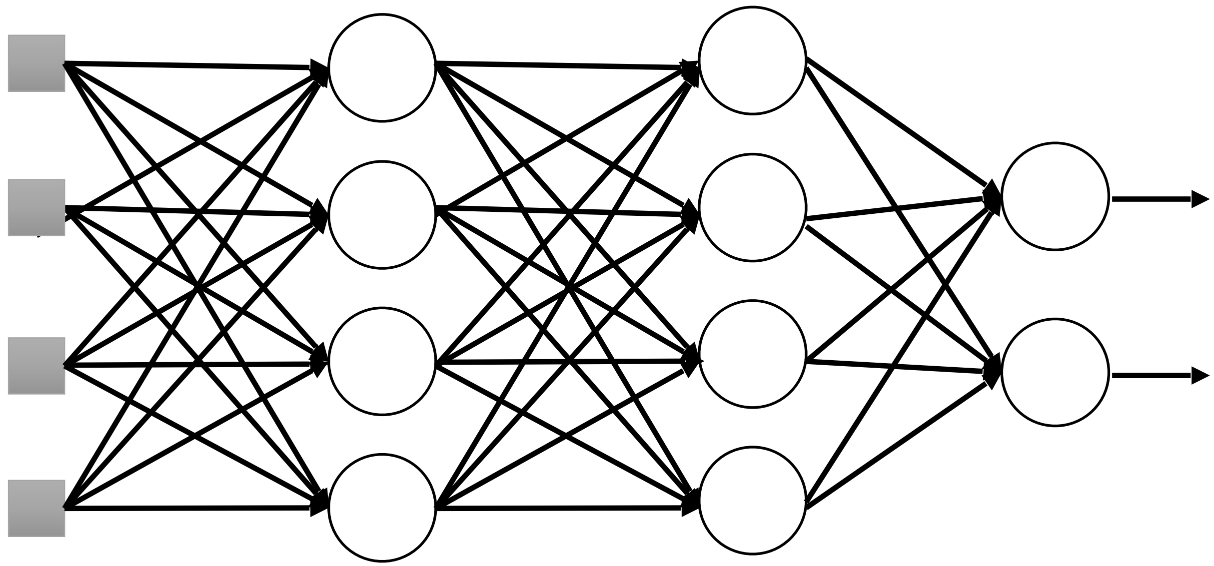
Dropout

Testing:

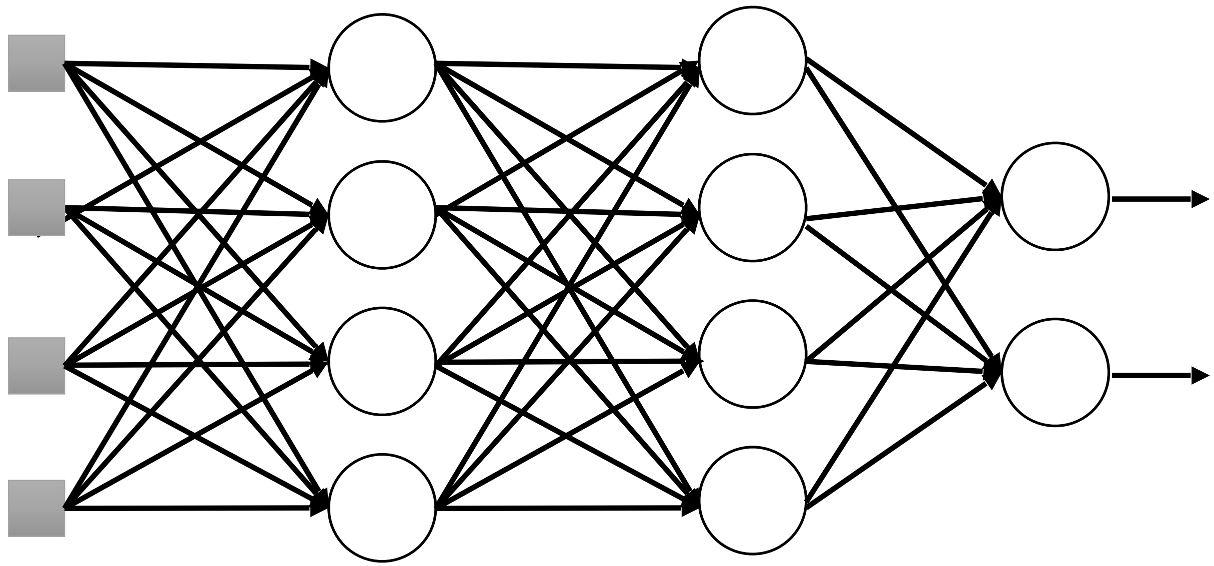


- ➢ **No dropout**
  - If the dropout rate at training is p%, all the weights times (1-p)%
  - Assume that the dropout rate is 50%. If a weight $w = 1$ by training, set $w = 0.5$ for testing.

Dropout - Intuitive Reason

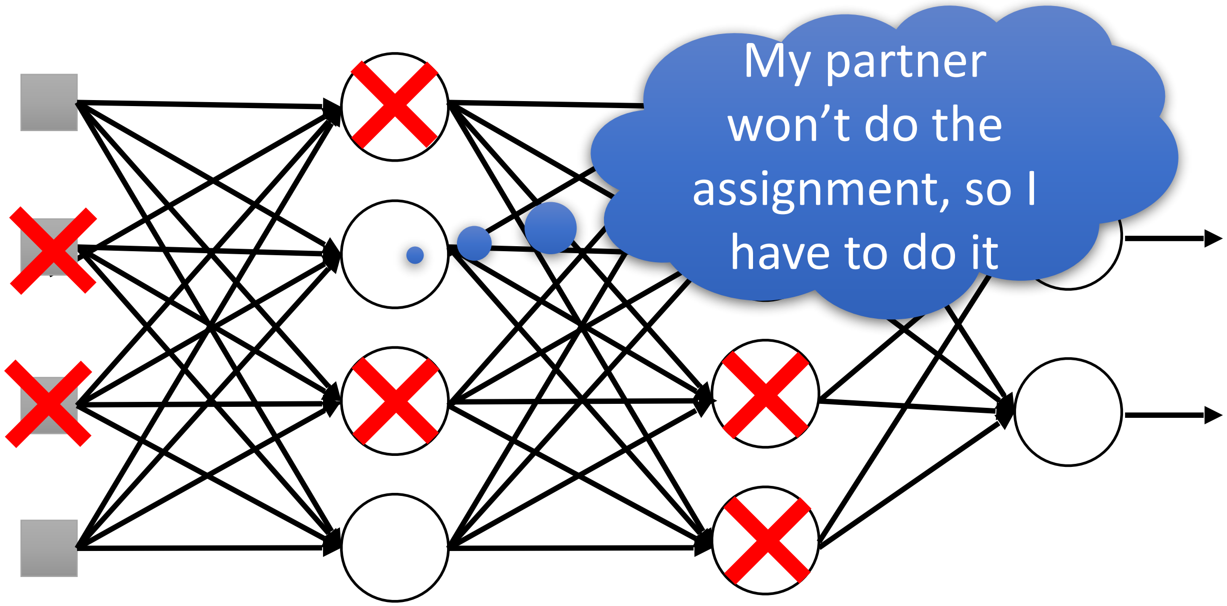


Dropout - Intuitive Reason



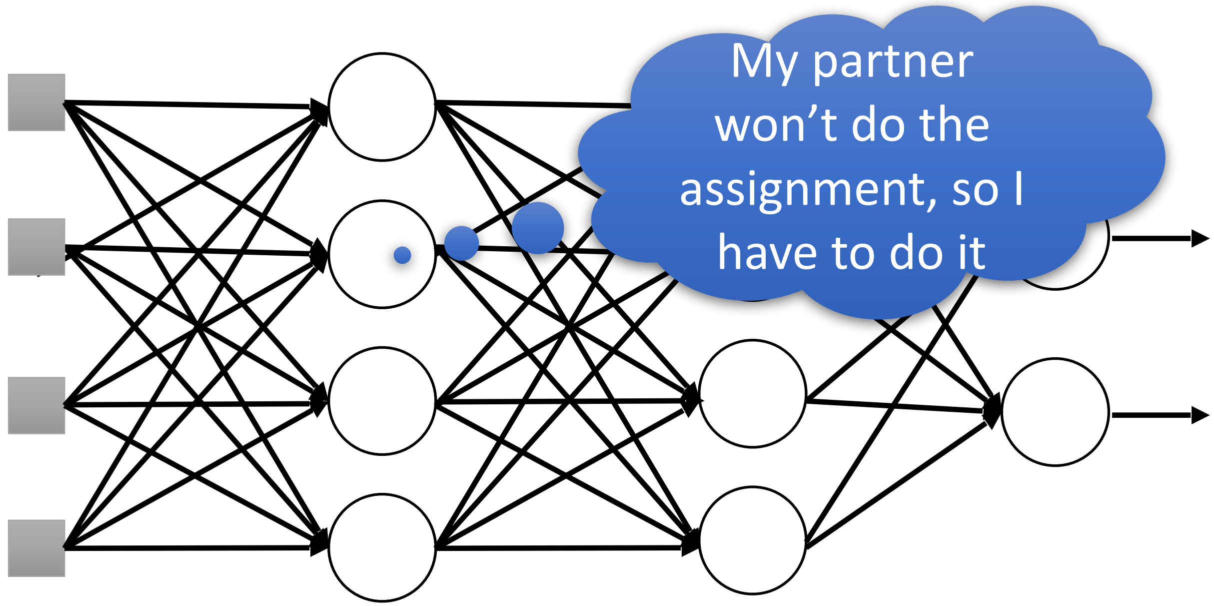
➢ When teams up, if everyone expect the partner will do the work, nothing will be done finally.

Dropout - Intuitive Reason



- ➢ When teams up, if everyone expect the partner will do the work, nothing will be done finally.
- ➢ However, if you know your partner will dropout, you will do better.

Dropout - Intuitive Reason



- When teams up, if everyone expect the partner will do the work, nothing will be done finally.
- However, if you know your partner will dropout, you will do better.
- When testing, no one dropout actually, so obtaining good results eventually.

Dropout - Intuitive Reason

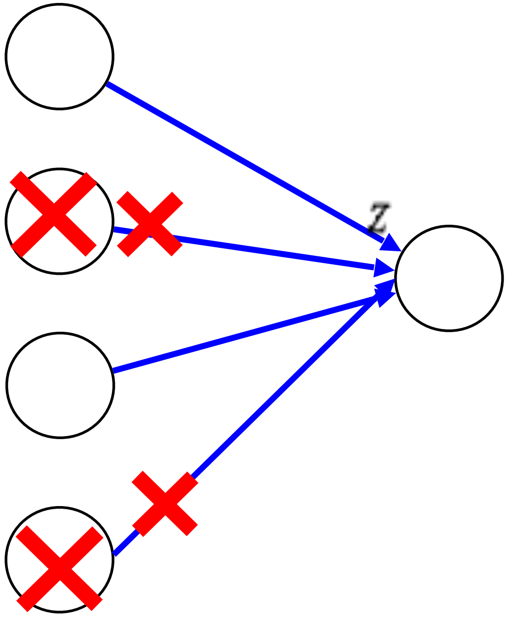
- Why the weights should multiply (1-p)% (dropout rate) when testing?

Dropout - Intuitive Reason

- Why the weights should multiply (1-p)% (dropout rate) when testing?

<u>Training of Dropout</u>

Assume dropout rate is 50%

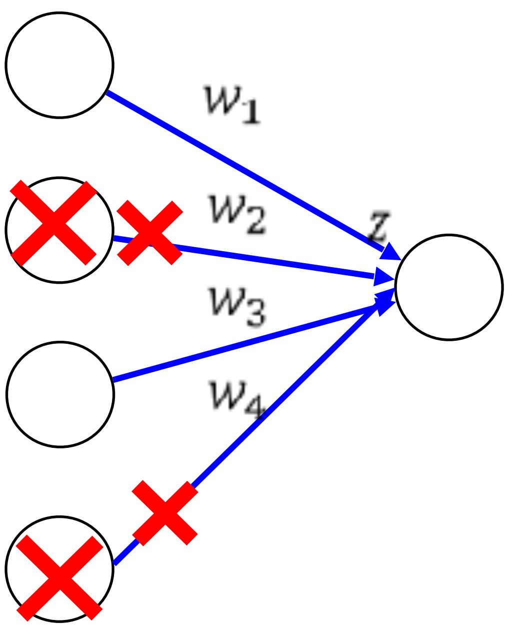


Dropout - Intuitive Reason

- Why the weights should multiply (1-p)% (dropout rate) when testing?

<u>Training of Dropout</u>

Assume dropout rate is 50%

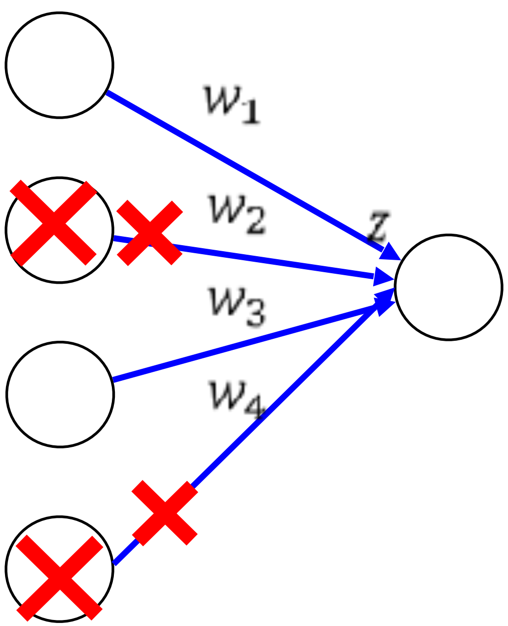


Dropout - Intuitive Reason

- Why the weights should multiply (1-p)% (dropout rate) when testing?

<u>Training of Dropout</u>

Assume dropout rate is 50%

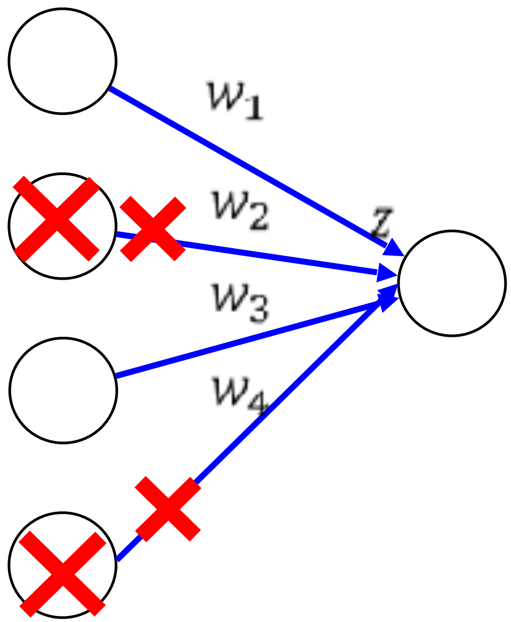


Dropout - Intuitive Reason

- Why the weights should multiply (1-p)% (dropout rate) when testing?

Training of Dropout

Assume dropout rate is 50%



Testing of Dropout

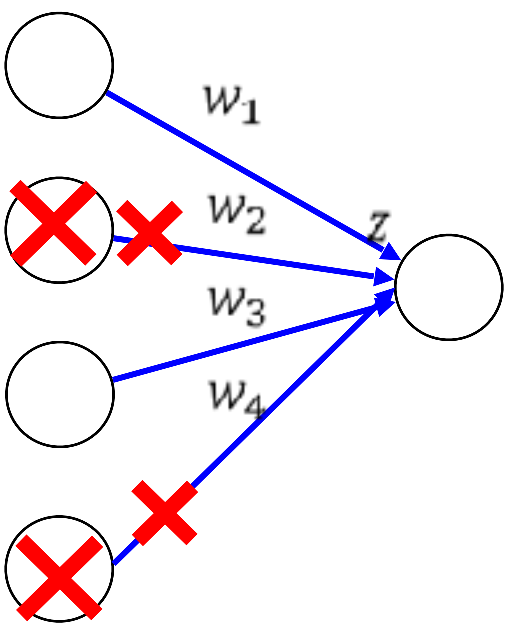
No dropout

Dropout - Intuitive Reason

- Why the weights should multiply (1-p)% (dropout rate) when testing?

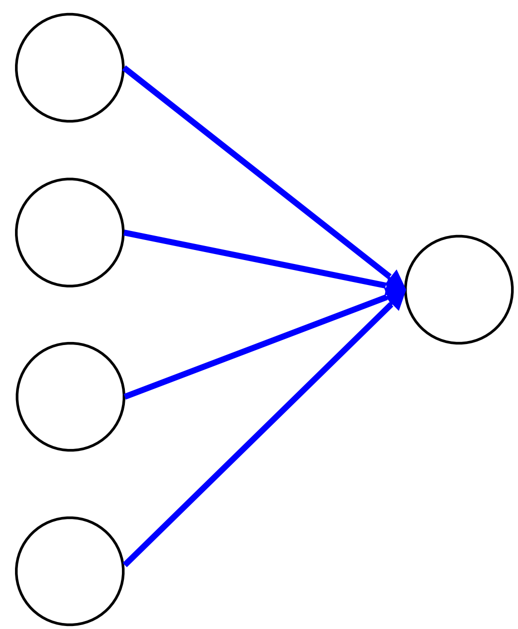
Training of Dropout

Assume dropout rate is 50%



Testing of Dropout

No dropout

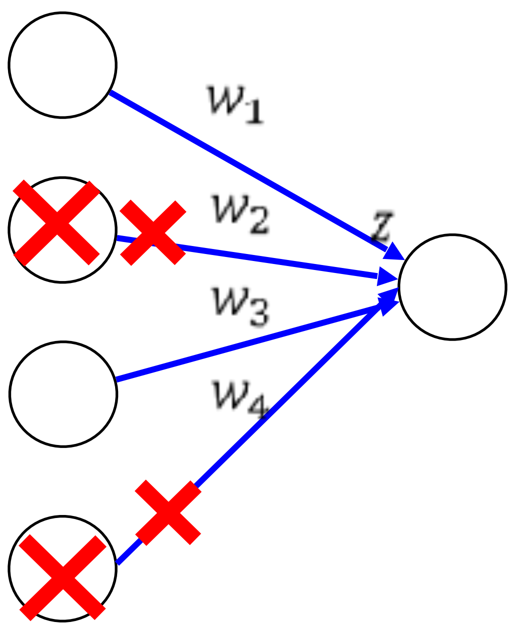


Dropout - Intuitive Reason

- Why the weights should multiply (1-p)% (dropout rate) when testing?

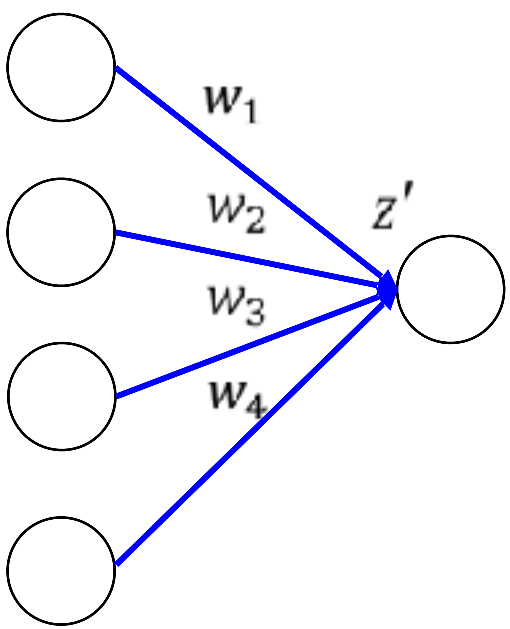
<u>Training of Dropout</u>

Assume dropout rate is 50%



<u>Testing of Dropout</u>

No dropout

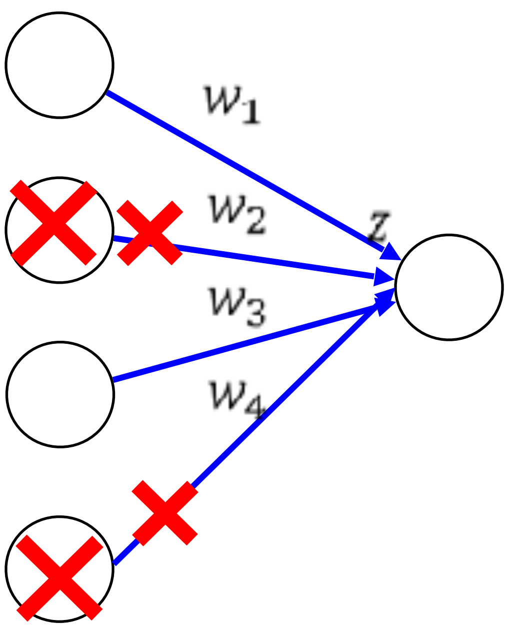


Dropout - Intuitive Reason

- Why the weights should multiply (1-p)% (dropout rate) when testing?

Training of Dropout

Assume dropout rate is 50%

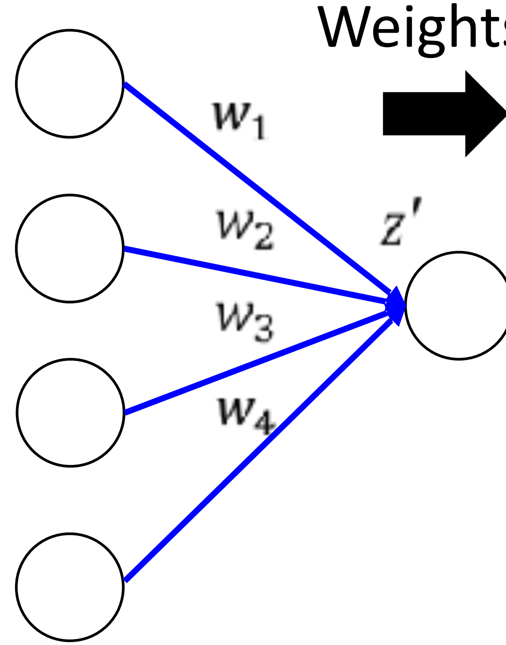


Testing of Dropout

No dropout

Weights from training

$\Rightarrow z' \approx 2z$

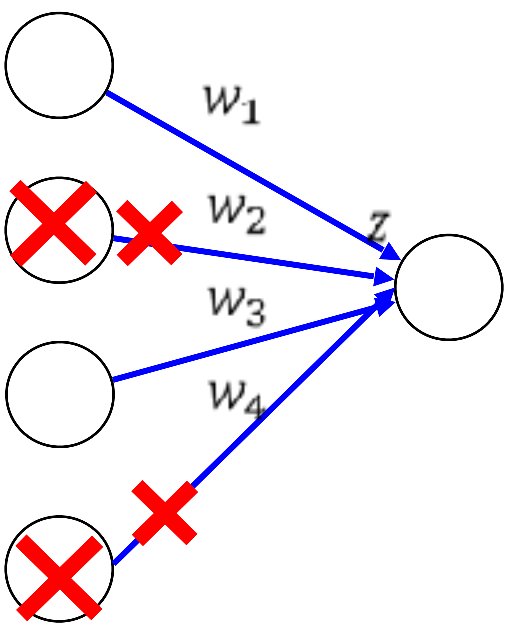


Dropout - Intuitive Reason

- Why the weights should multiply (1-p)% (dropout rate) when testing?

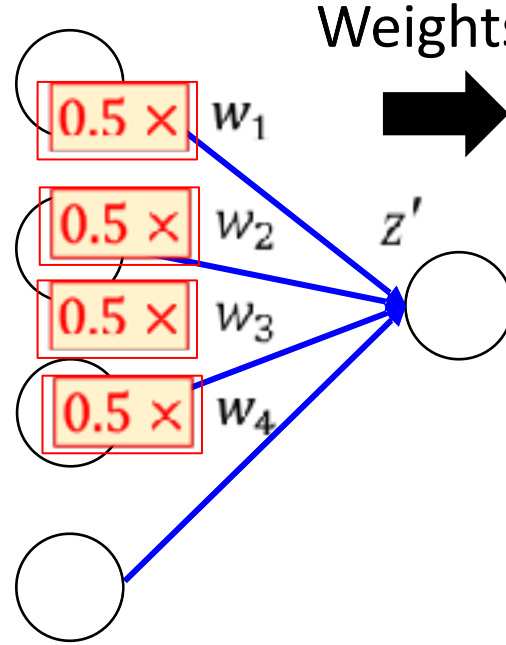
Training of Dropout

Assume dropout rate is 50%



Testing of Dropout

No dropout



Weights from training

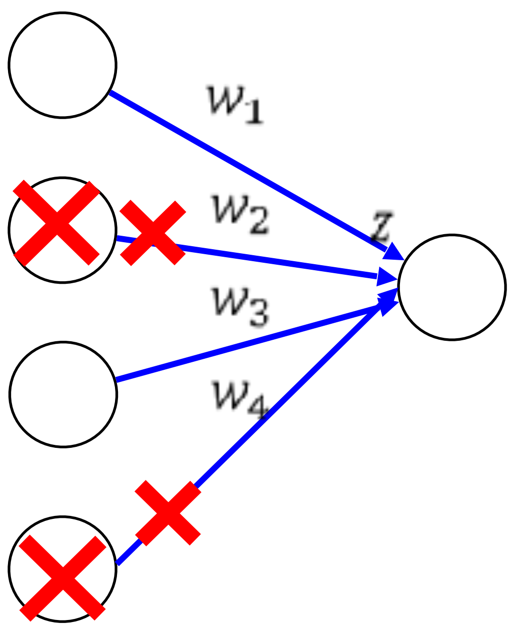
$z' \approx 2z$

Dropout - Intuitive Reason

- Why the weights should multiply (1-p)% (dropout rate) when testing?

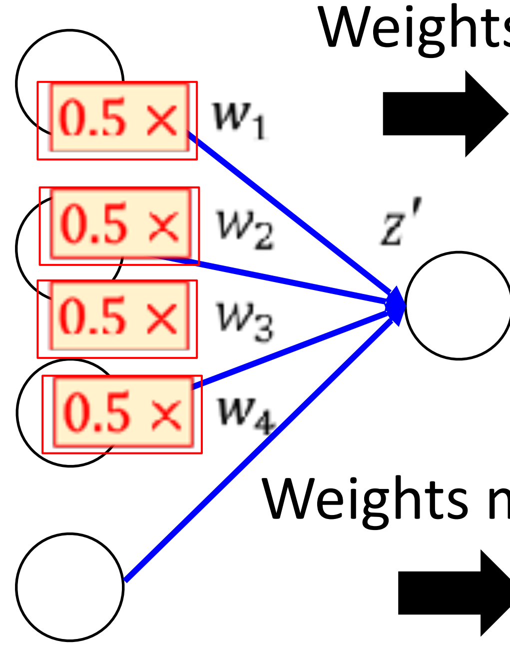
Training of Dropout

Assume dropout rate is 50%



Testing of Dropout

No dropout



Weights from training ➡ $z' \approx 2z$

Weights multiply (1-p)% ➡ $z' \approx z$

Concluding Remarks

Concluding Remarks

- Introduction of deep learning
- Discussing some reasons using deep learning
- New techniques for deep learning
    - ReLU, Maxout
    - Giving all the parameters different learning rates
    - Dropout

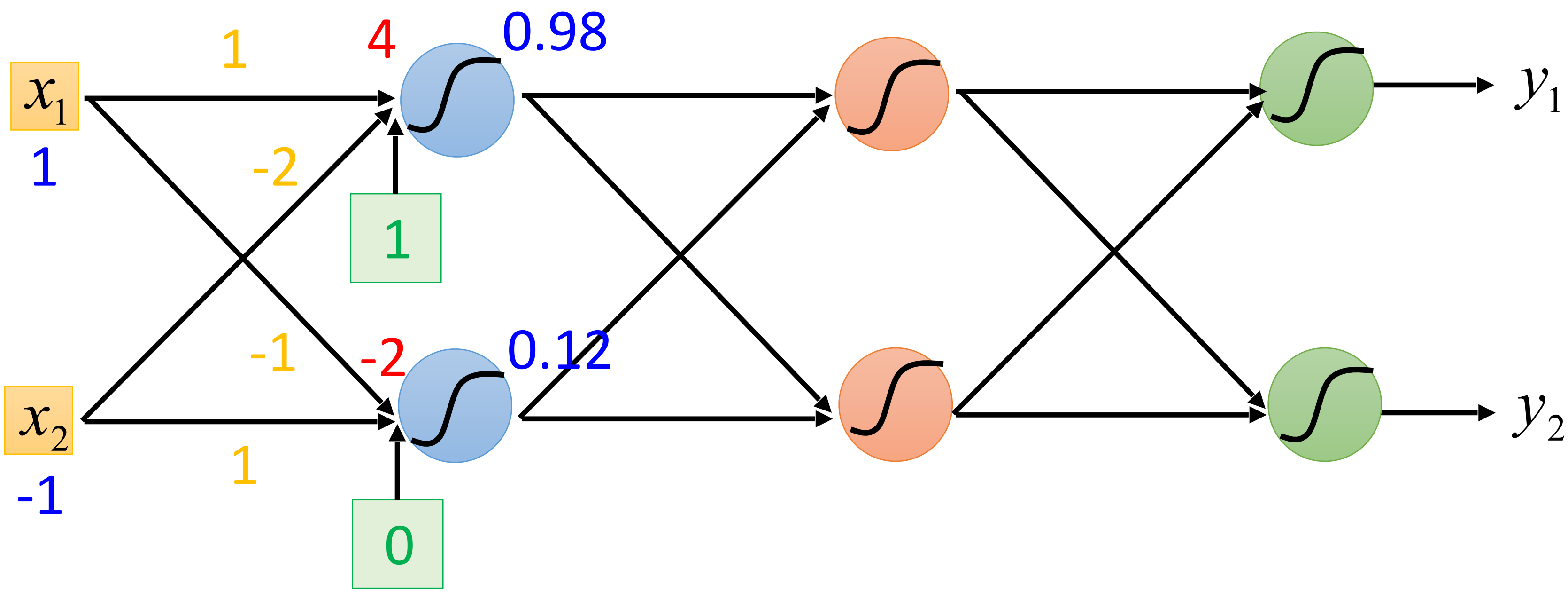
Reading Materials

- “Neural Networks and Deep Learning”
  - written by Michael Nielsen
  - http://neuralnetworksanddeeplearning.com/
- “Deep Learning” (not finished yet)
  - Written by Yoshua Bengio, Ian J. Goodfellow and Aaron Courville
  - http://www.iro.umontreal.ca/~bengioy/dlbook/

Thank you
for your attention!

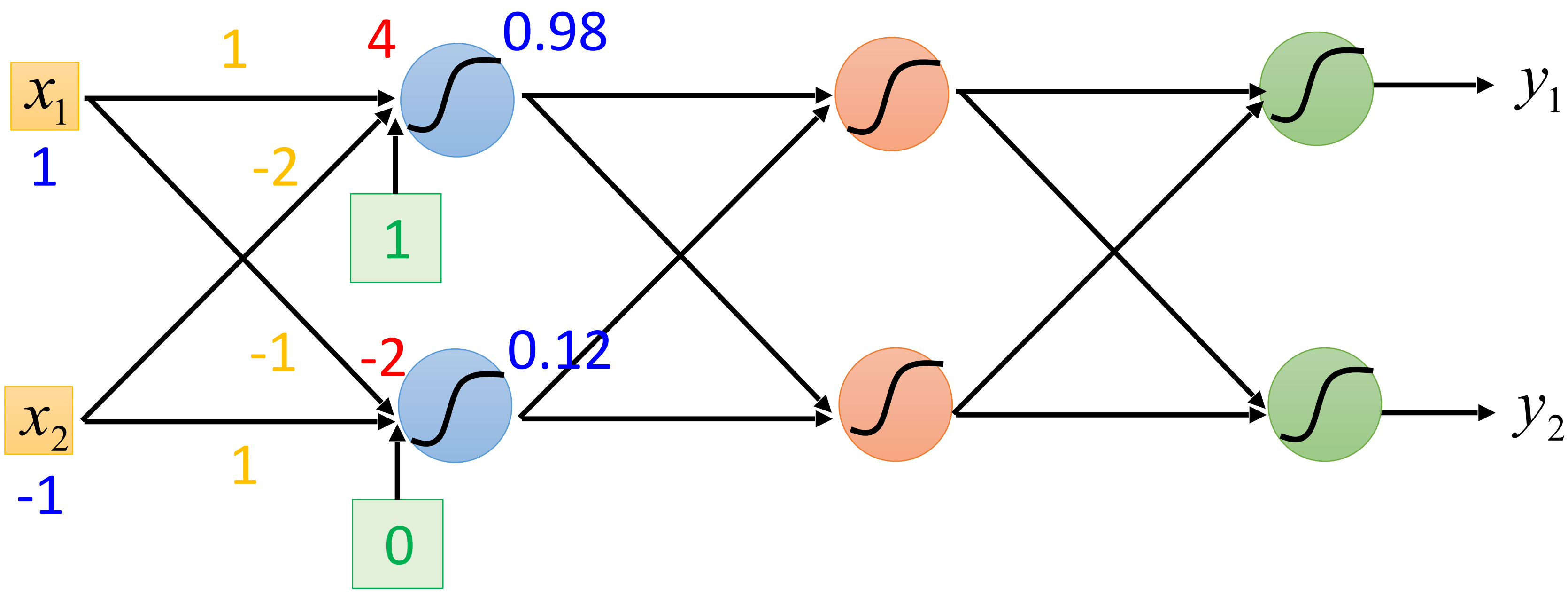
Appendix

Matrix Operation



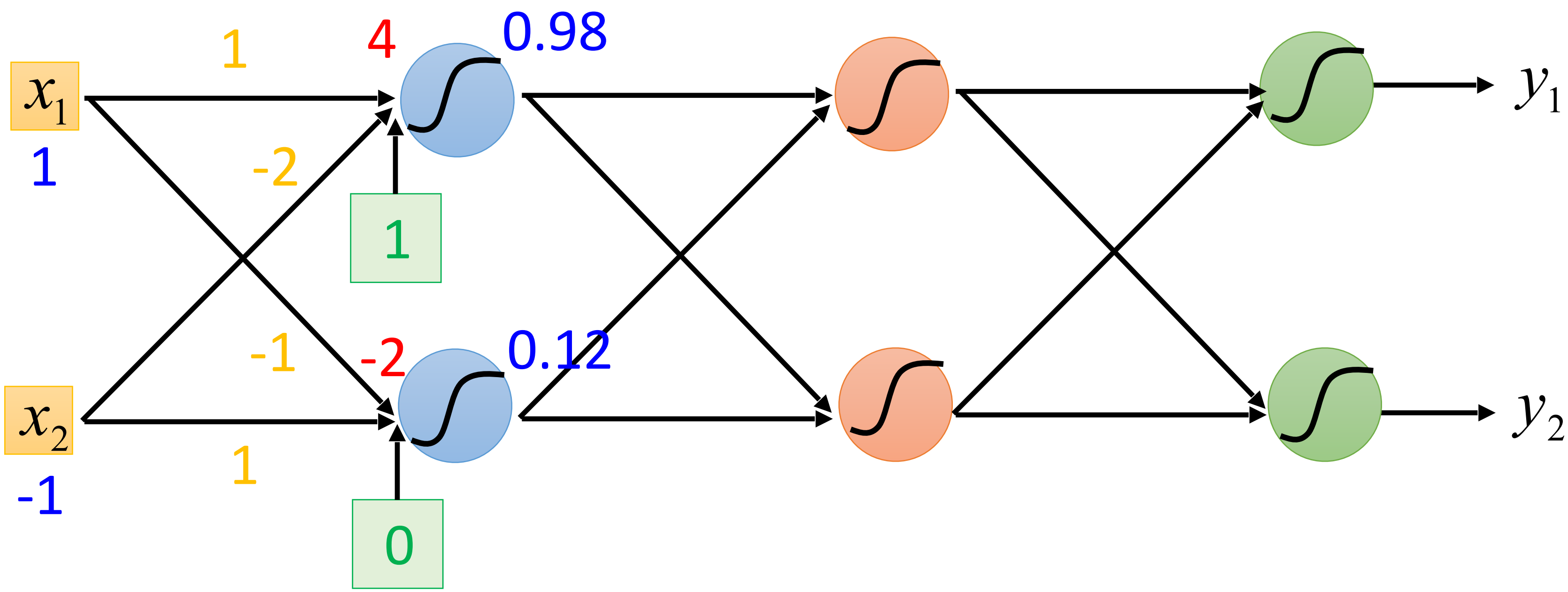
$$\sigma\left(\begin{bmatrix}1 & -2\\ -1 & 1\end{bmatrix}\begin{bmatrix}1\\ -1\end{bmatrix}+\begin{bmatrix}1\\ 0\end{bmatrix}\right)=\begin{bmatrix}0.98\\ 0.12\end{bmatrix}$$

Matrix Operation



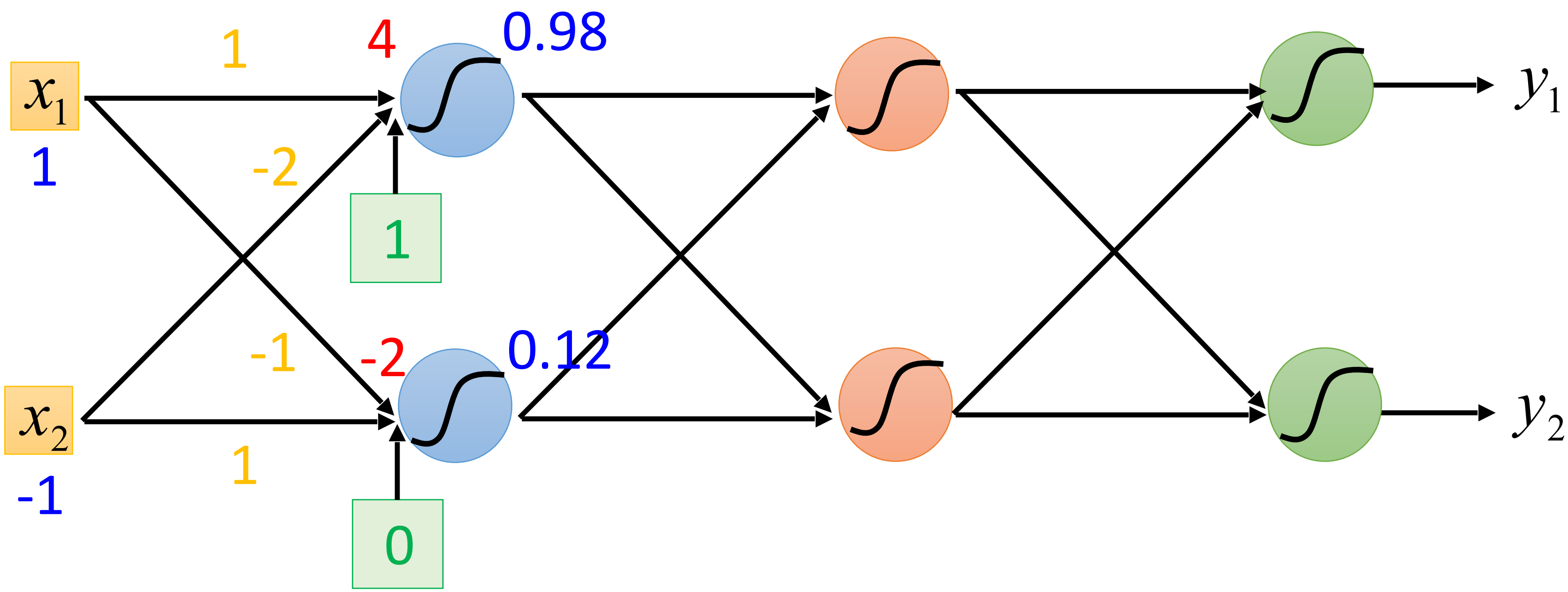
$$\sigma\left(\boxed{\quad W \quad} \begin{bmatrix} 1 \\ -1 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right) = \begin{bmatrix} 0.98 \\ 0.12 \end{bmatrix}$$

Matrix Operation



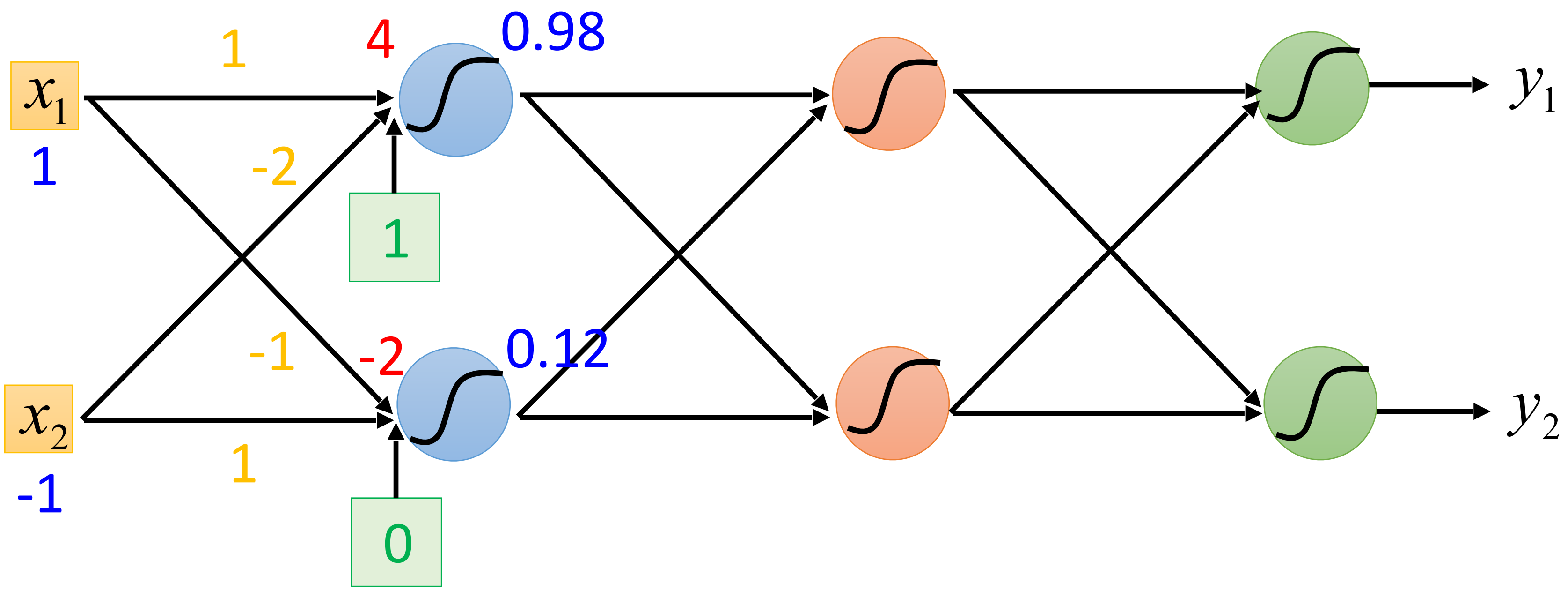
$$\sigma\left(W \; x + \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right) = \begin{bmatrix} 0.98 \\ 0.12 \end{bmatrix}$$

Matrix Operation



$$\sigma\left(W \; x + b \right) = \begin{bmatrix} 0.98 \\ 0.12 \end{bmatrix}$$

Matrix Operation



$$\sigma(\ W\ x + b\ ) = a$$

Why Deep? – Logic Circuits

Why Deep? – Logic Circuits

- A two levels of basic logic gates can represent any Boolean function.

Why Deep? – Logic Circuits

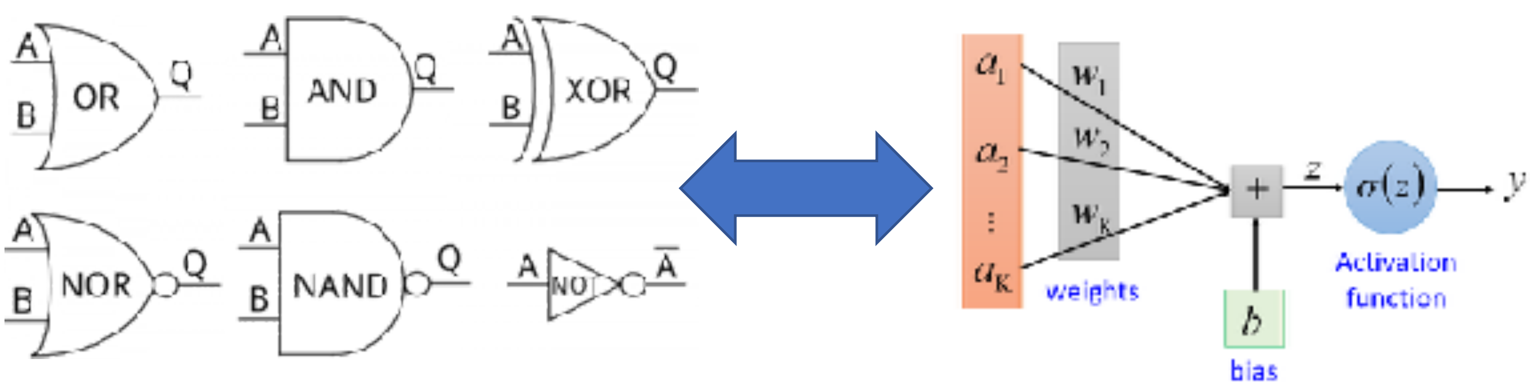
- A two levels of basic logic gates can represent any Boolean function.
- However, no one uses two levels of logic gates to build computers

Why Deep? – Logic Circuits

- A two levels of basic logic gates can represent any Boolean function.
- However, no one uses two levels of logic gates to build computers
- Using multiple layers of logic gates to build some functions are much simpler (less gates needed).

Why Deep? – Logic Circuits

- A two levels of basic logic gates can represent any Boolean function.
- However, no one uses two levels of logic gates to build computers
- Using multiple layers of logic gates to build some functions are much simpler (less gates needed).



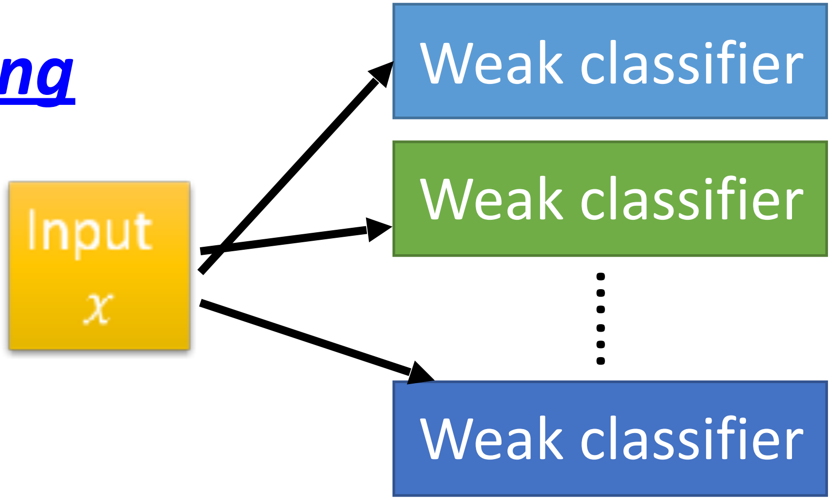
Boosting

Deep Learning

Boosting

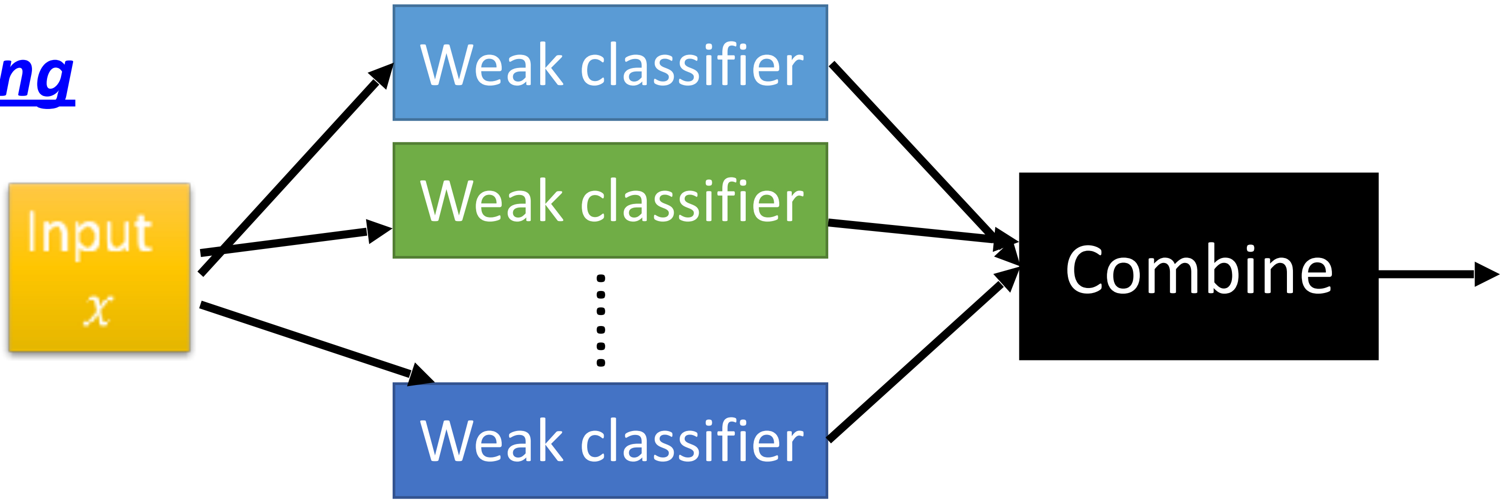
Deep Learning

Boosting



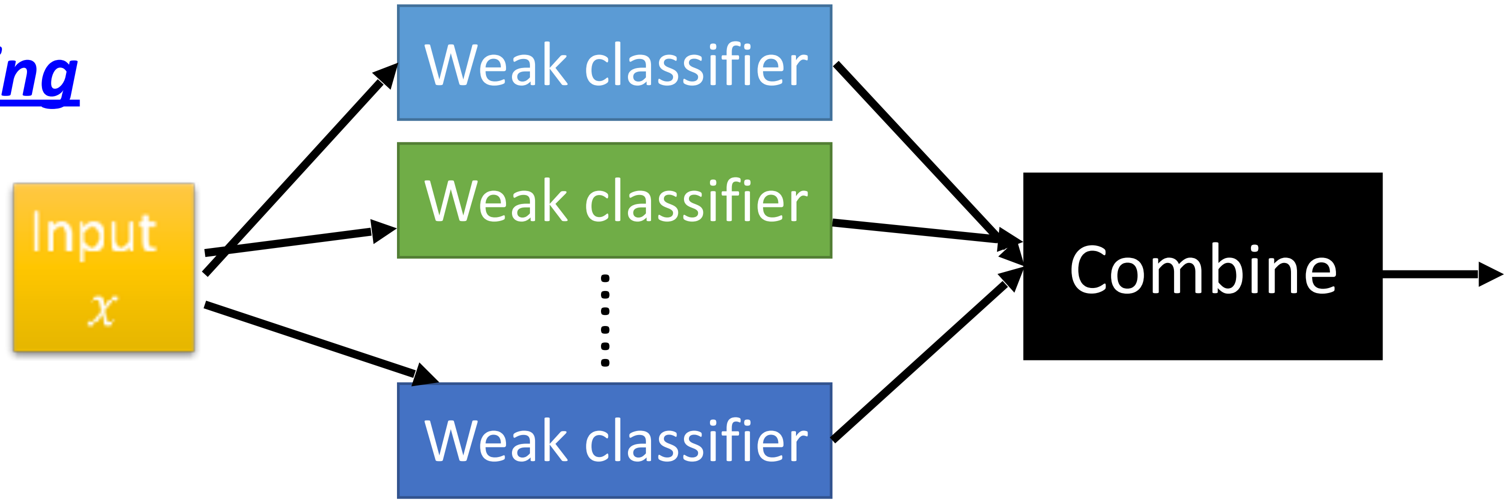
Deep Learning

Boosting

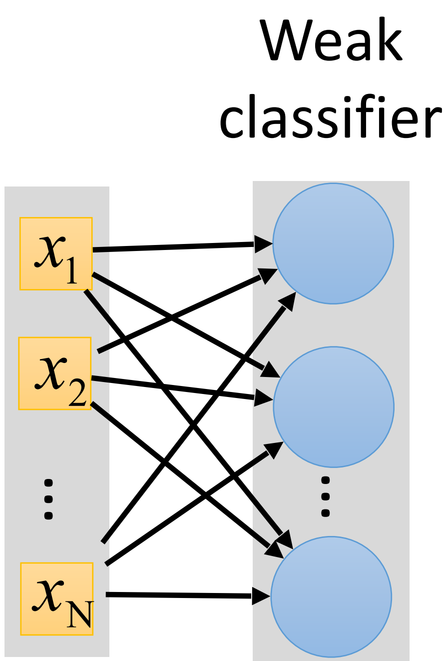


Deep Learning

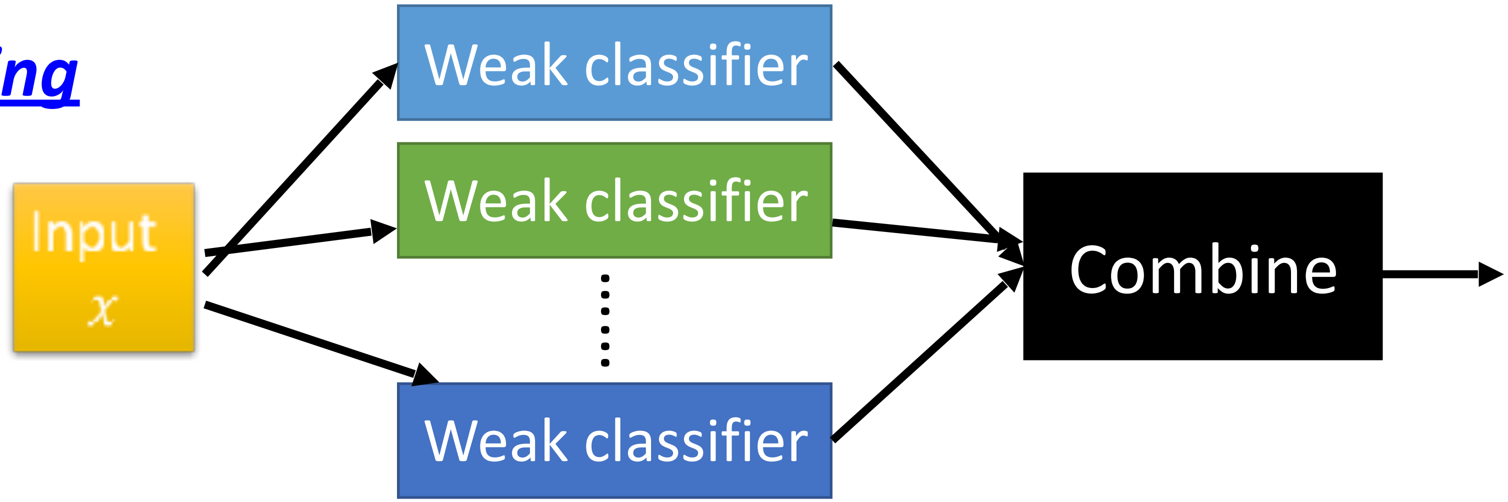
Boosting



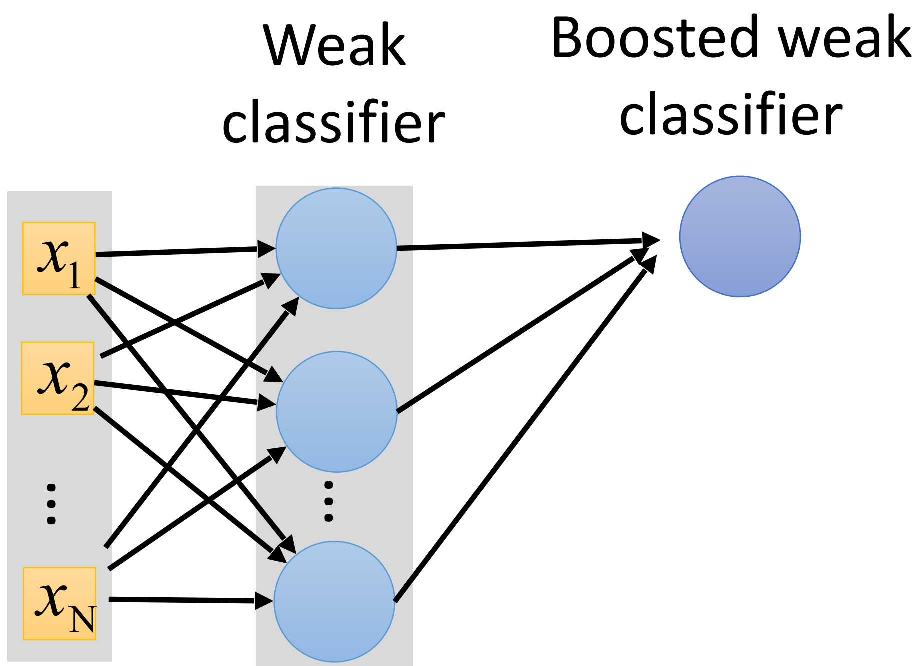
Deep Learning



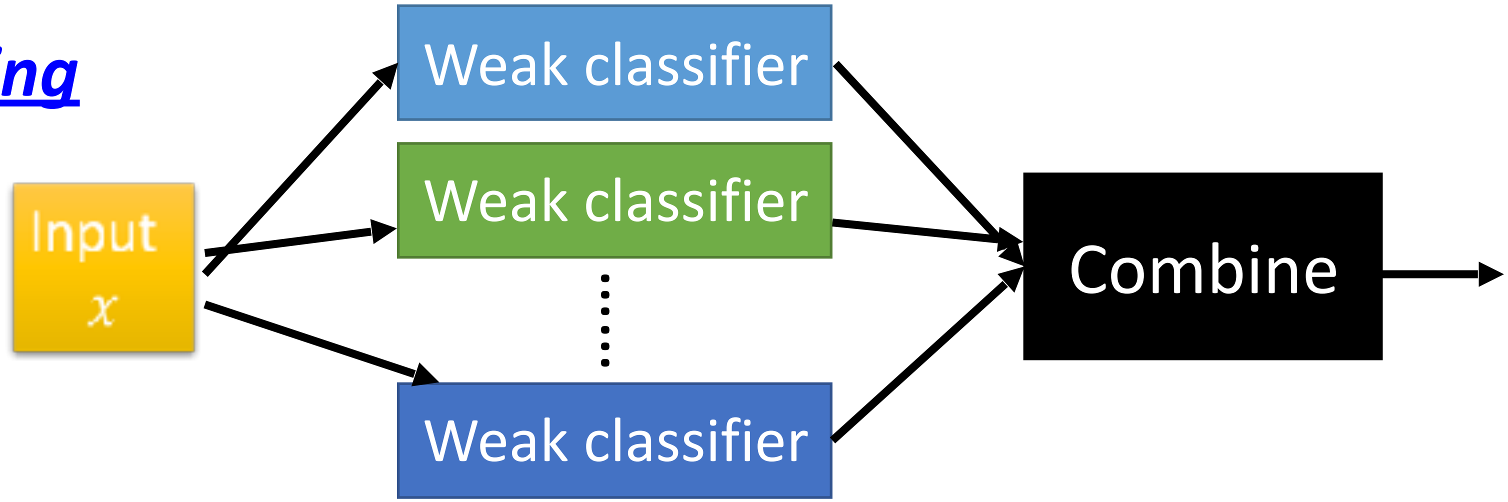
Boosting



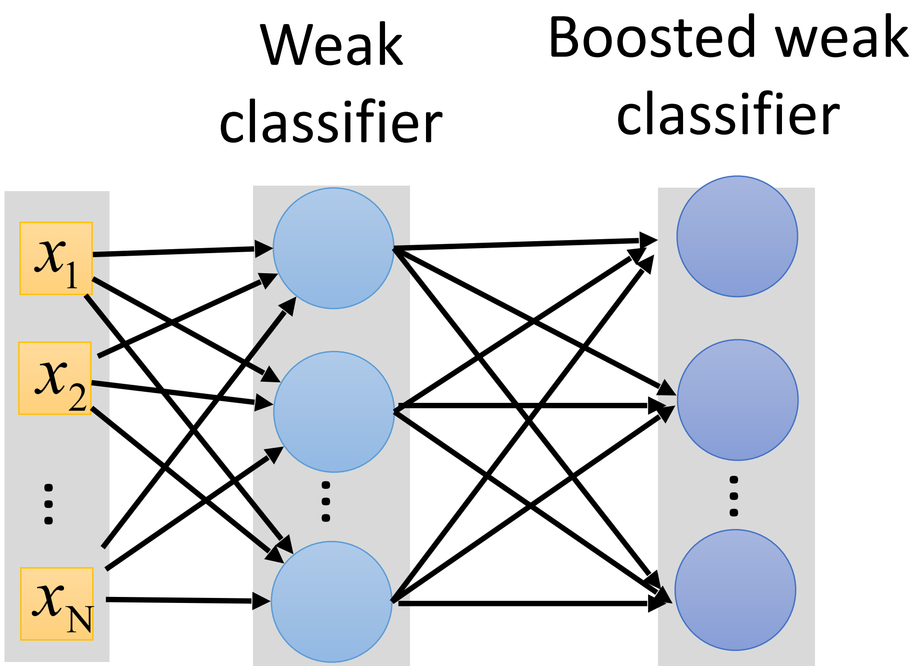
Deep Learning



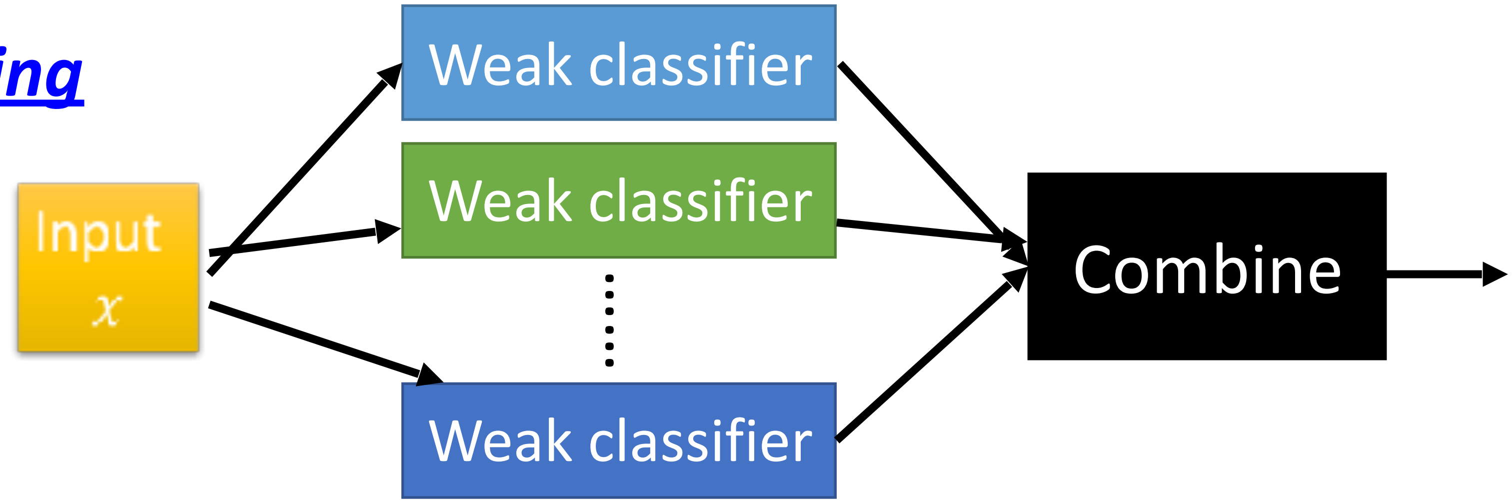
Boosting



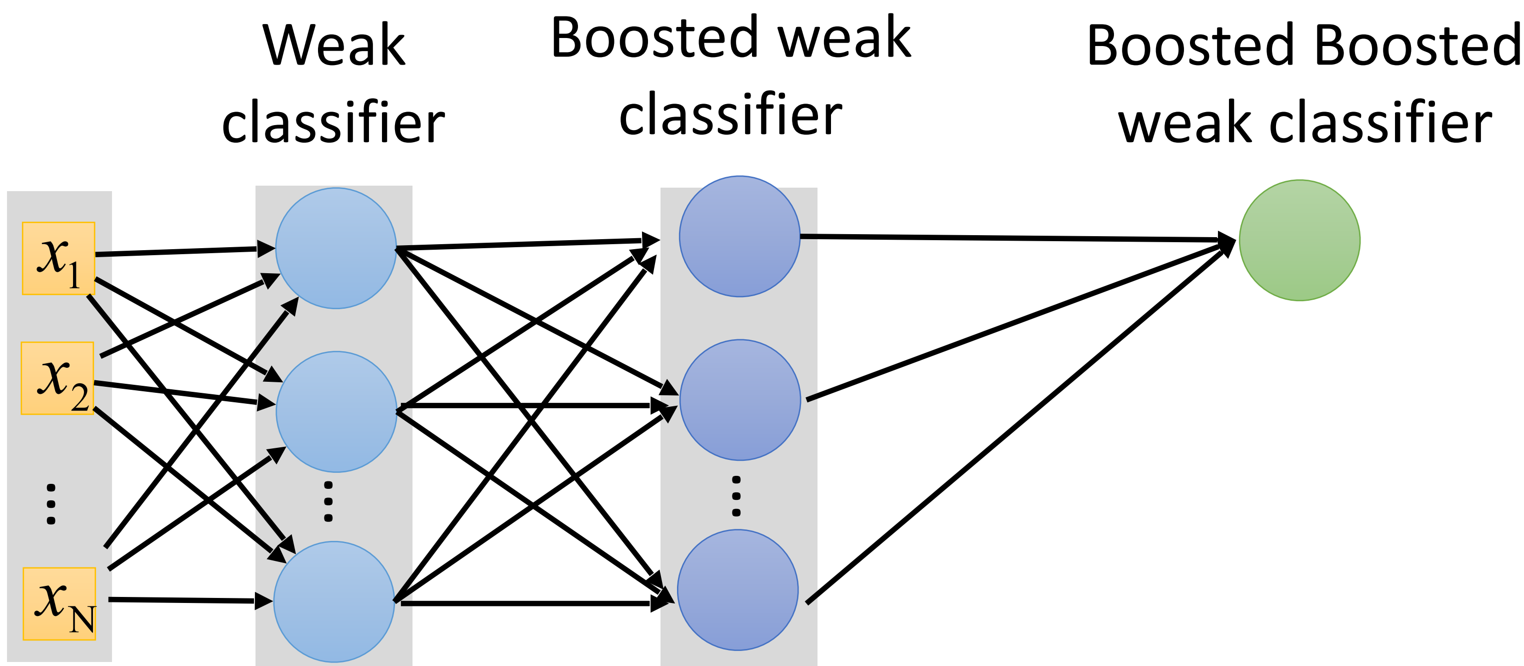
Deep Learning



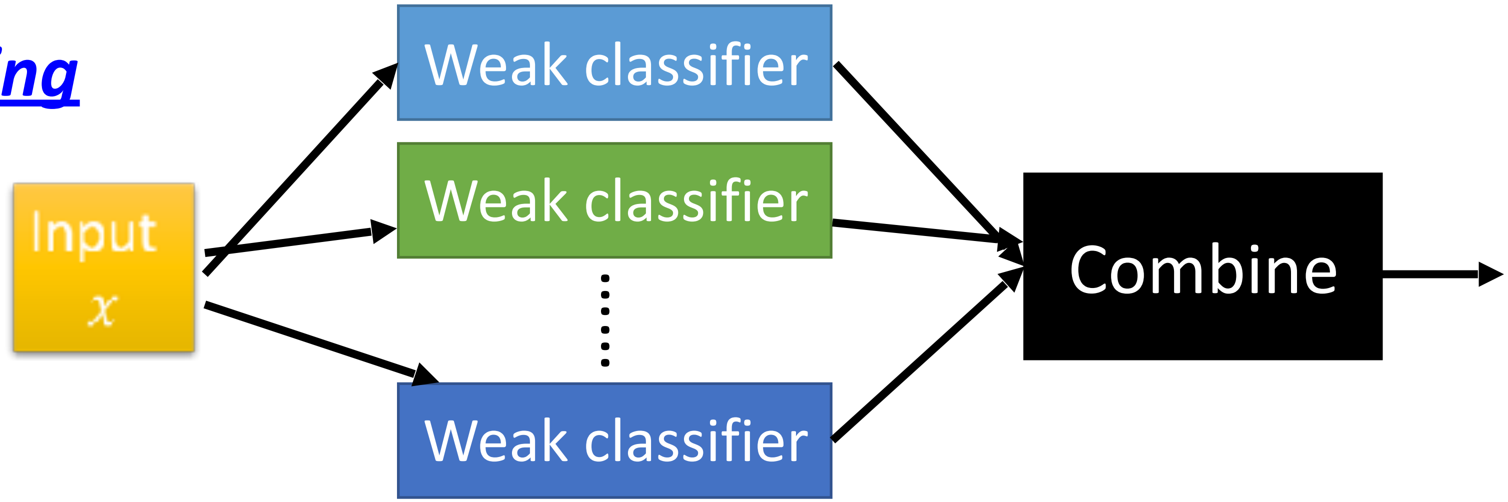
Boosting



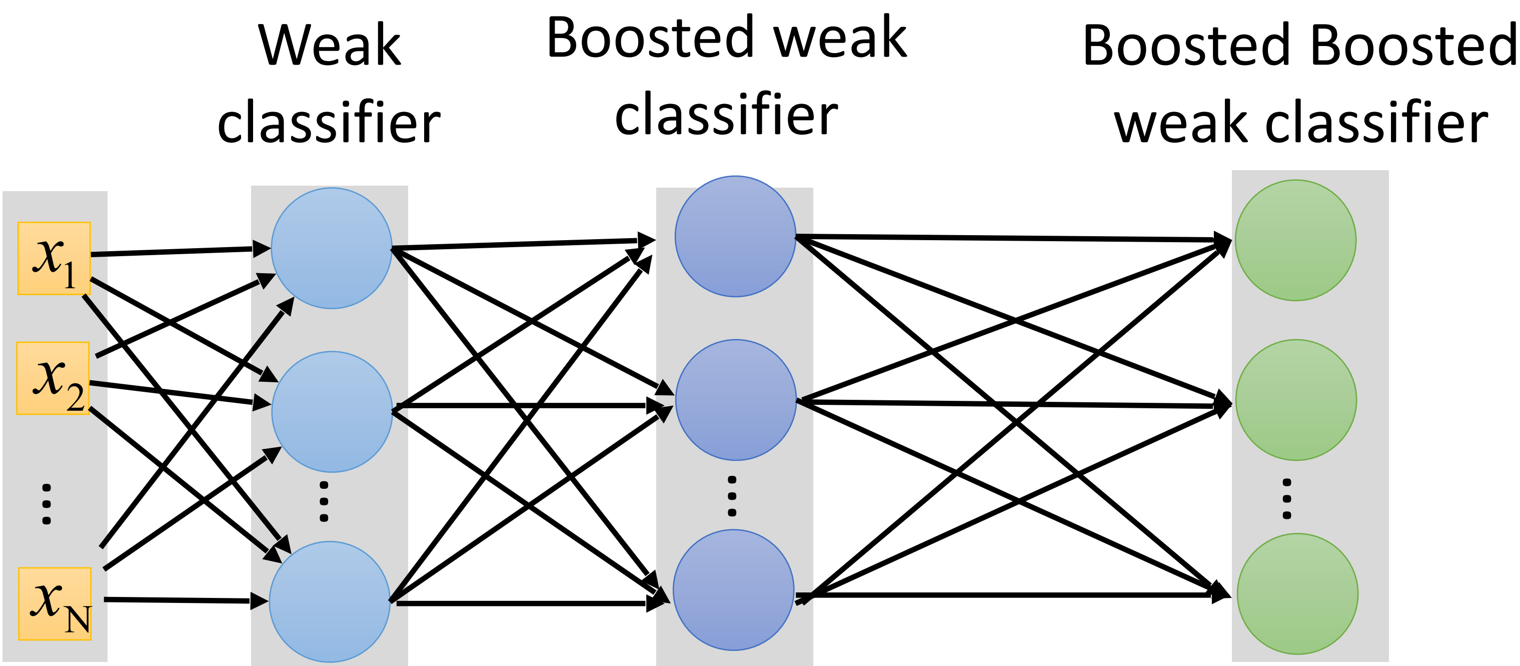
Deep Learning



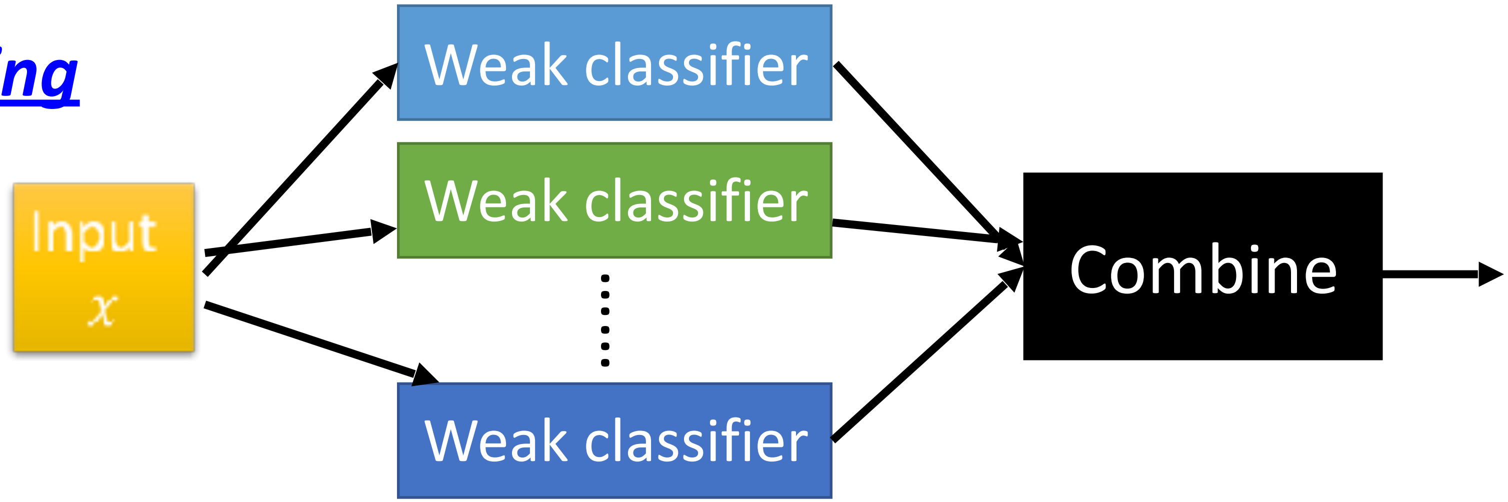
Boosting



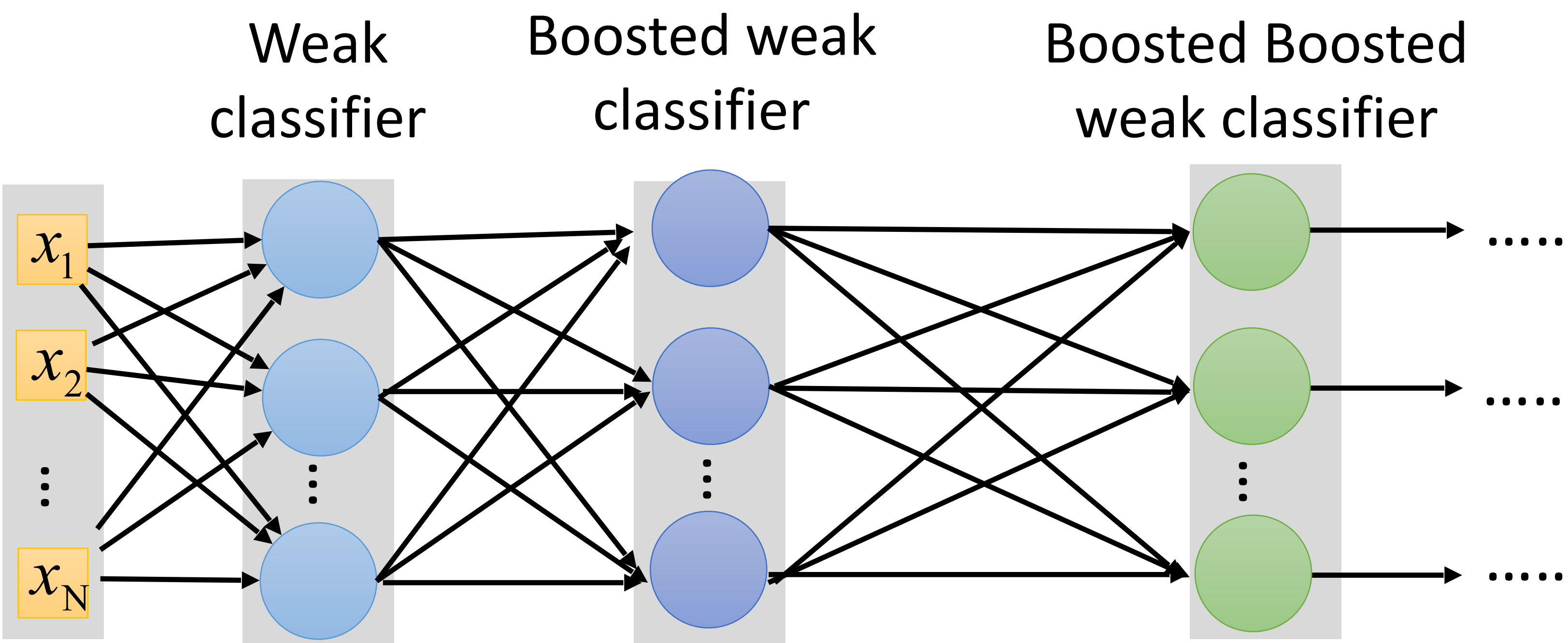
Deep Learning



Boosting



Deep Learning

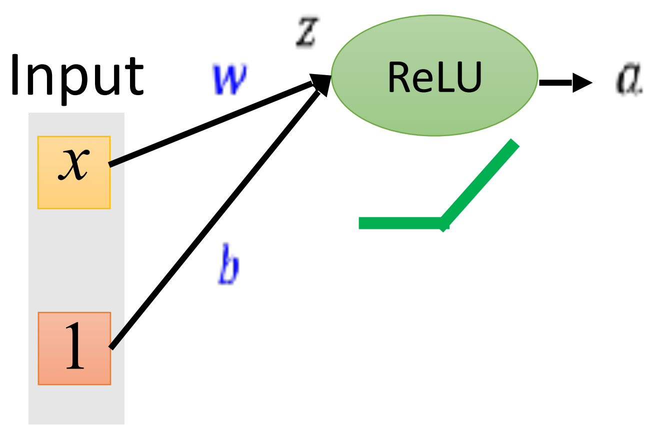


Maxout

ReLU is a special cases of Maxout

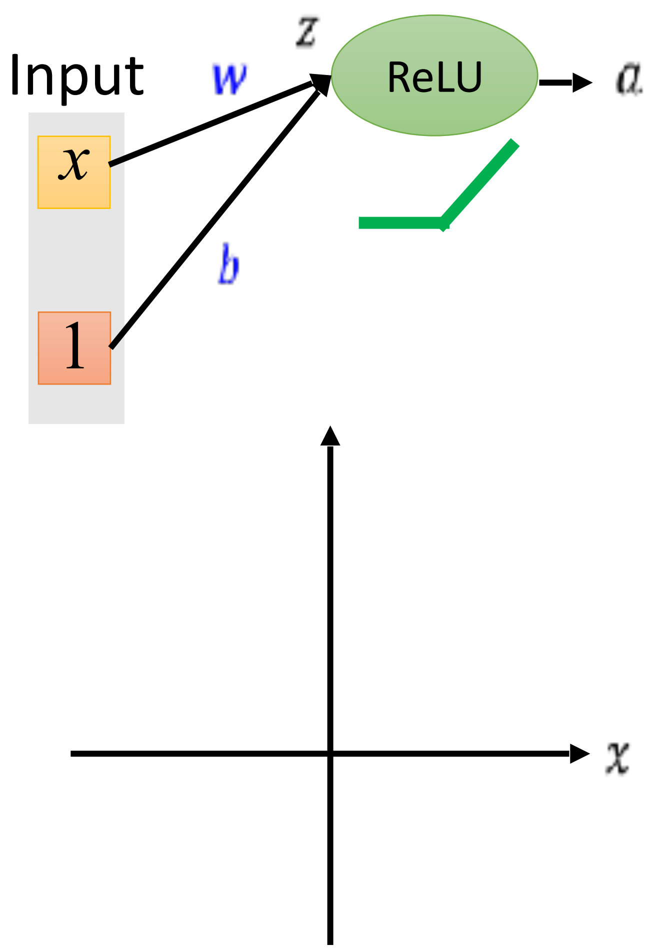
Maxout

ReLU is a special cases of Maxout



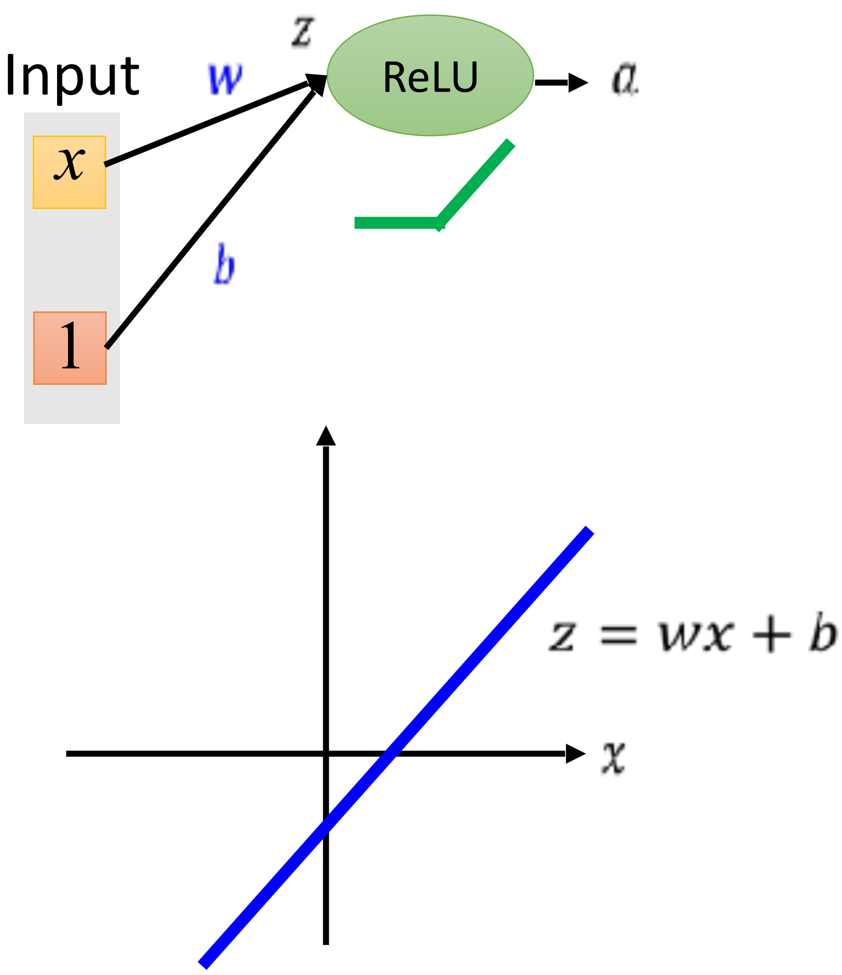
Maxout

ReLU is a special cases of Maxout



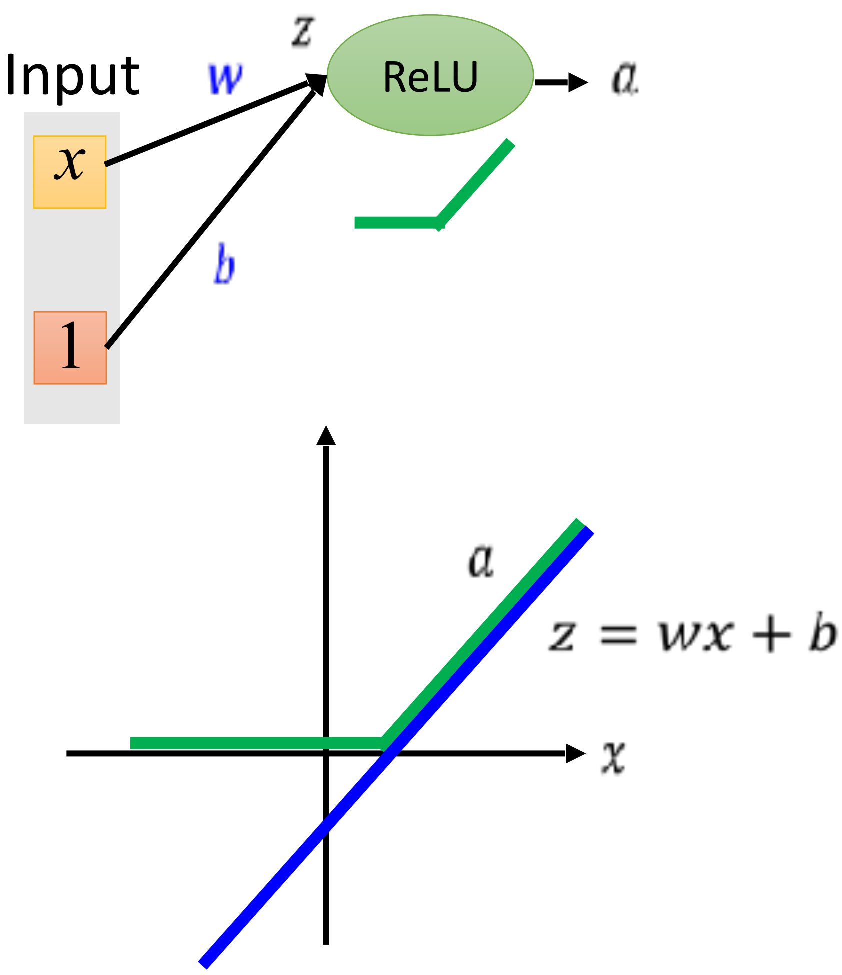
Maxout

ReLU is a special cases of Maxout



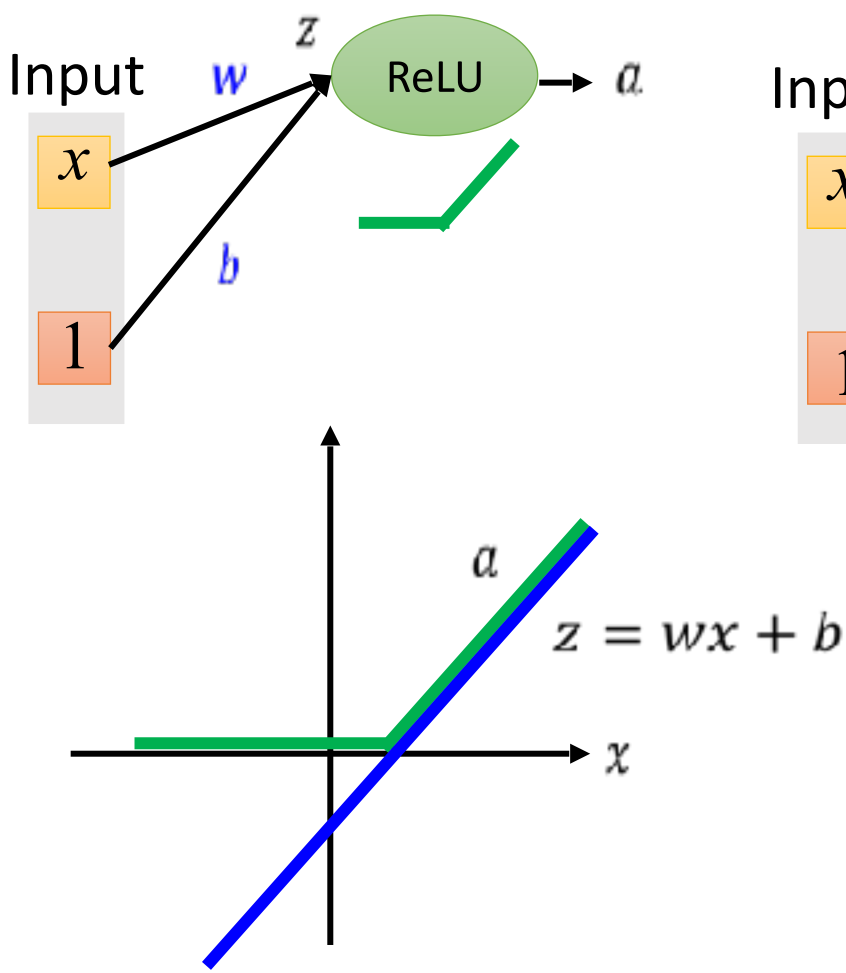
Maxout

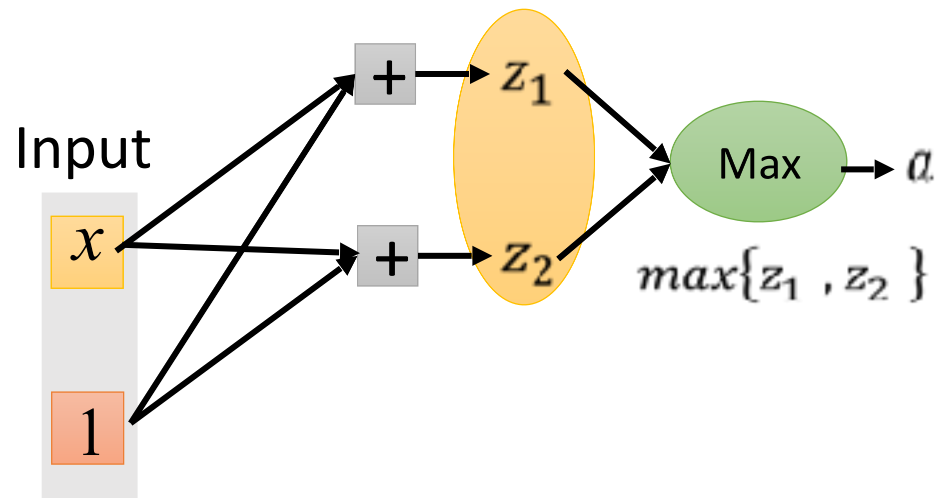
ReLU is a special cases of Maxout



Maxout

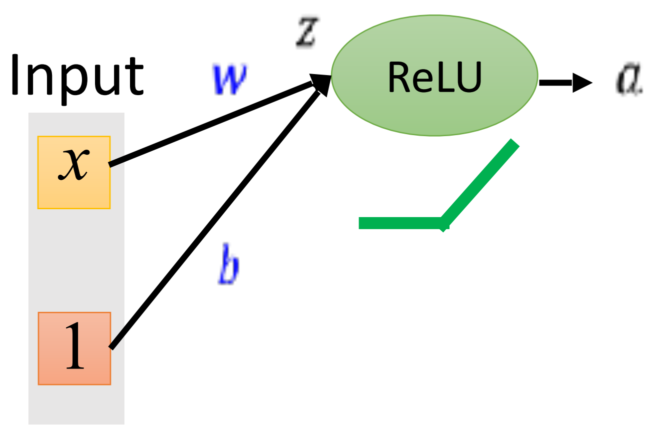
ReLU is a special cases of Maxout

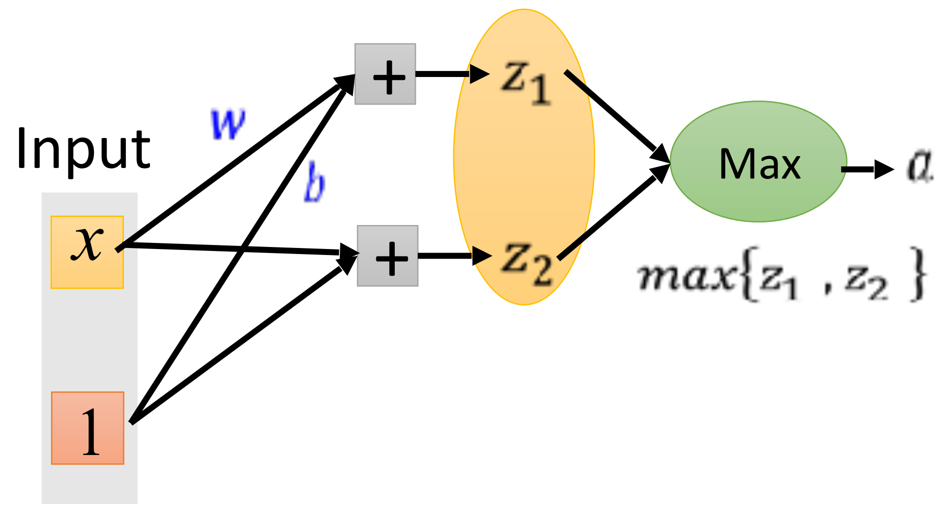


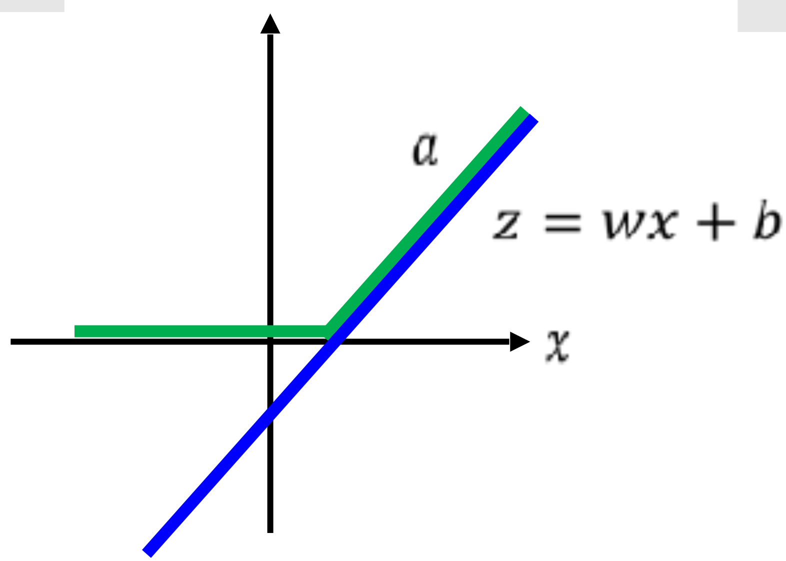


Maxout

ReLU is a special cases of Maxout

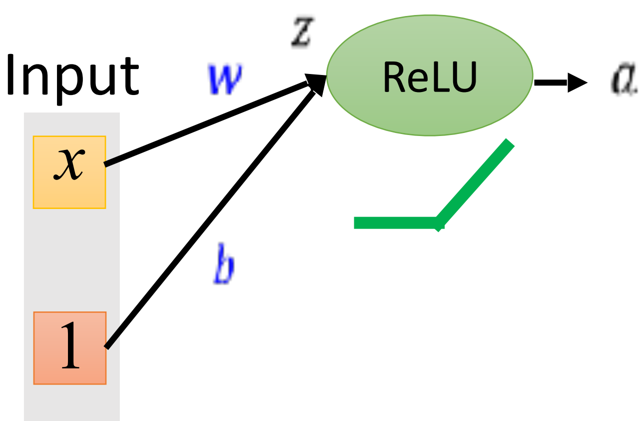


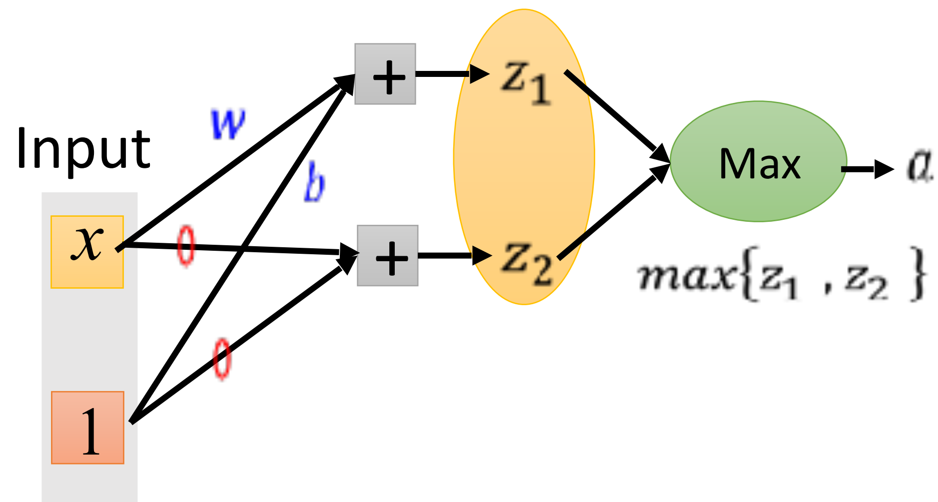


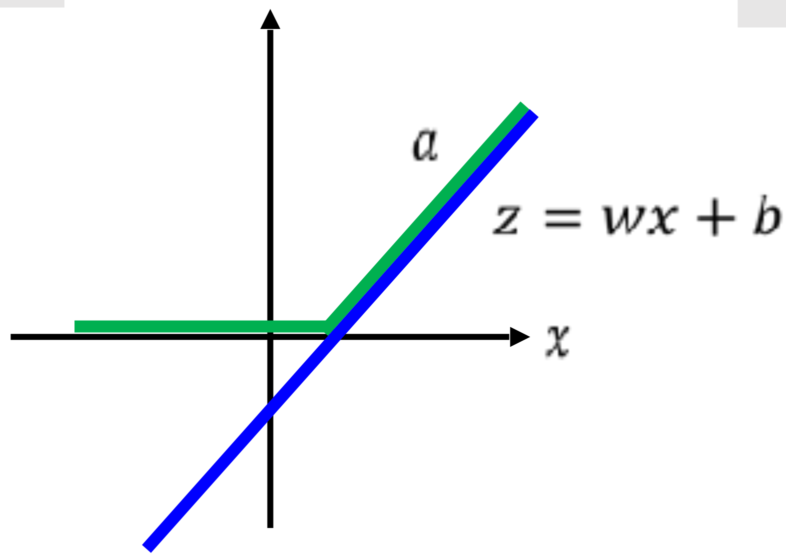


Maxout

ReLU is a special cases of Maxout

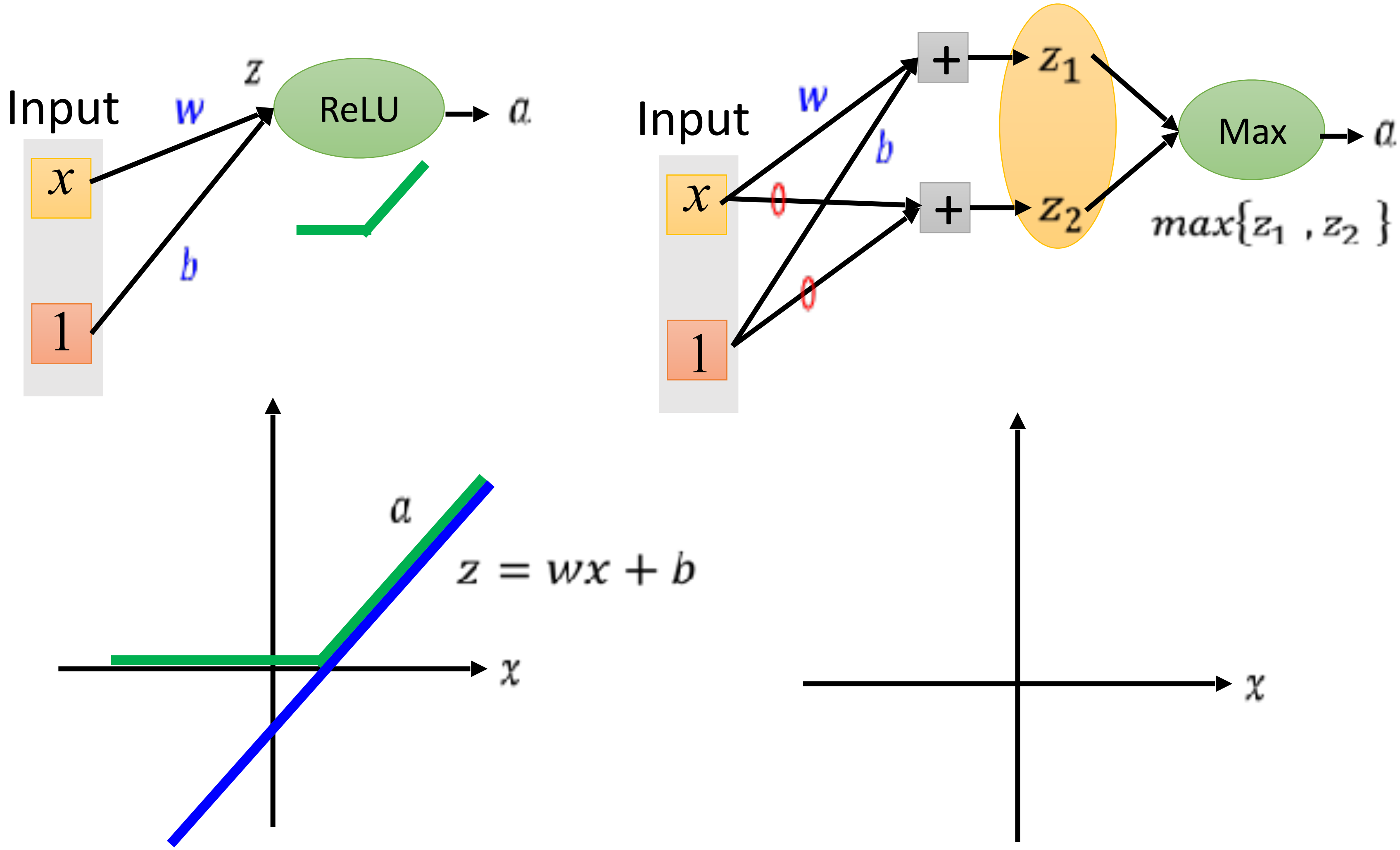






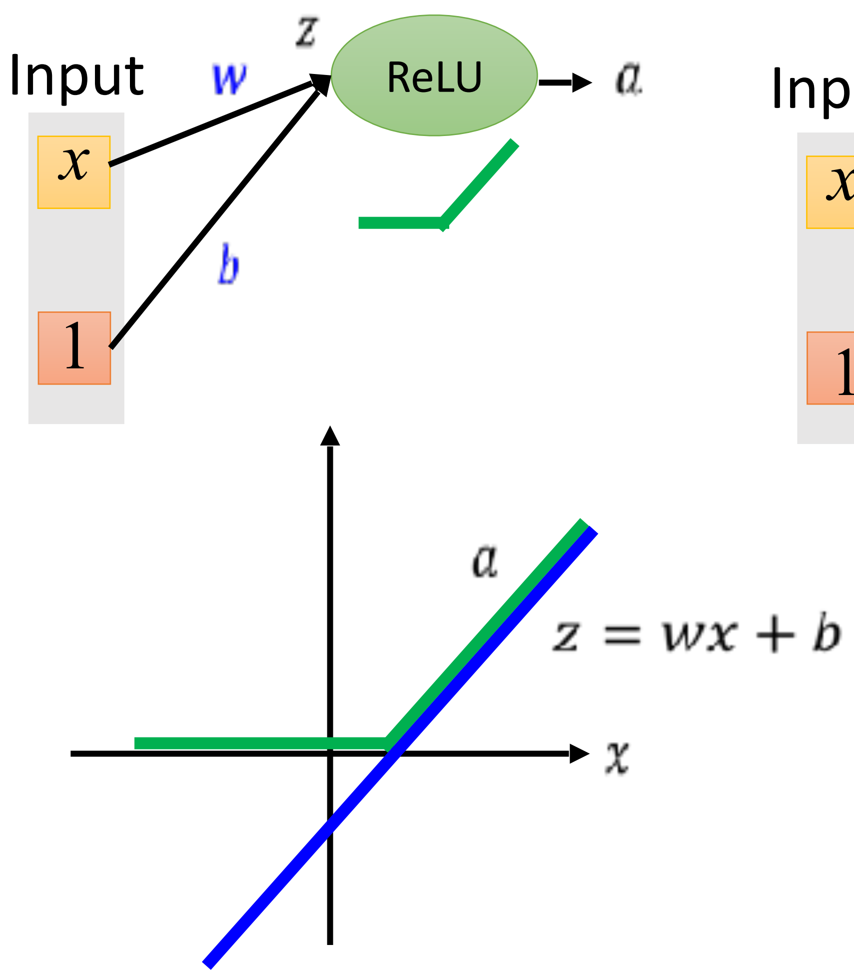
Maxout

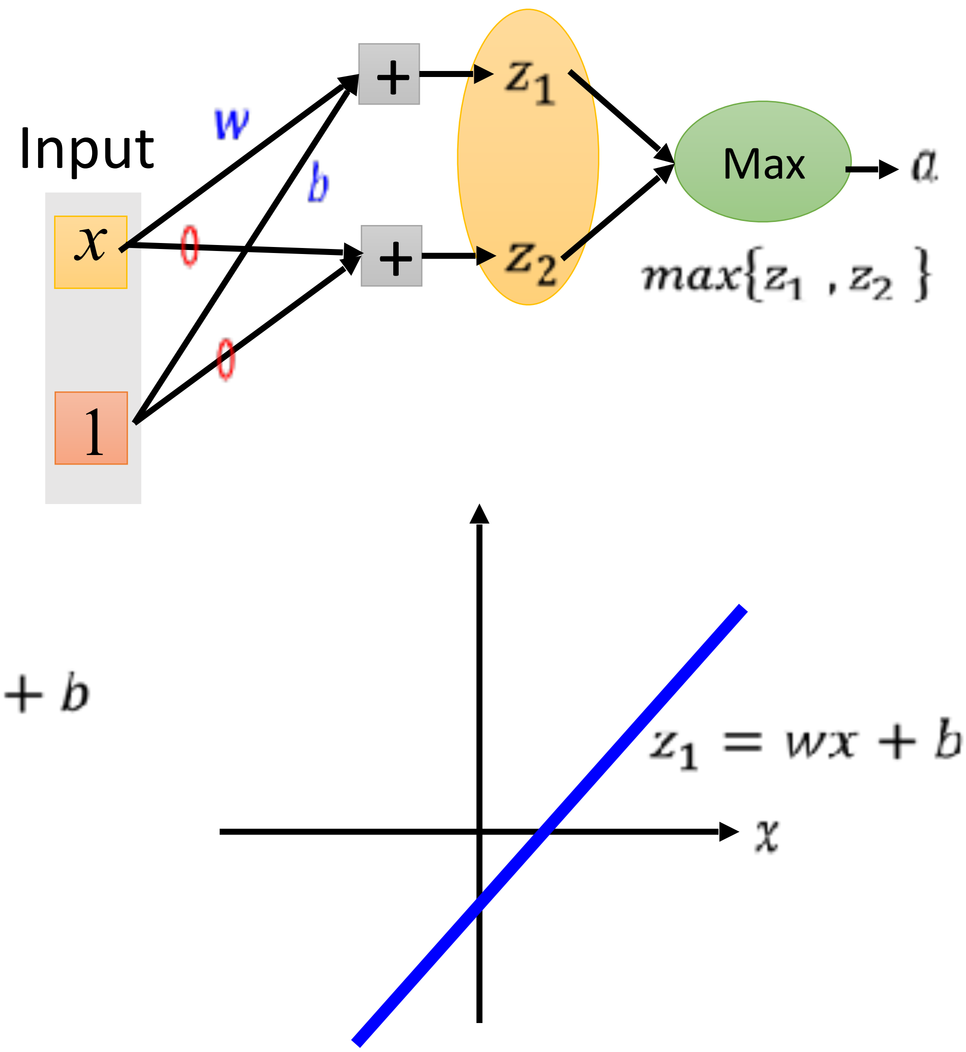
ReLU is a special cases of Maxout



Maxout

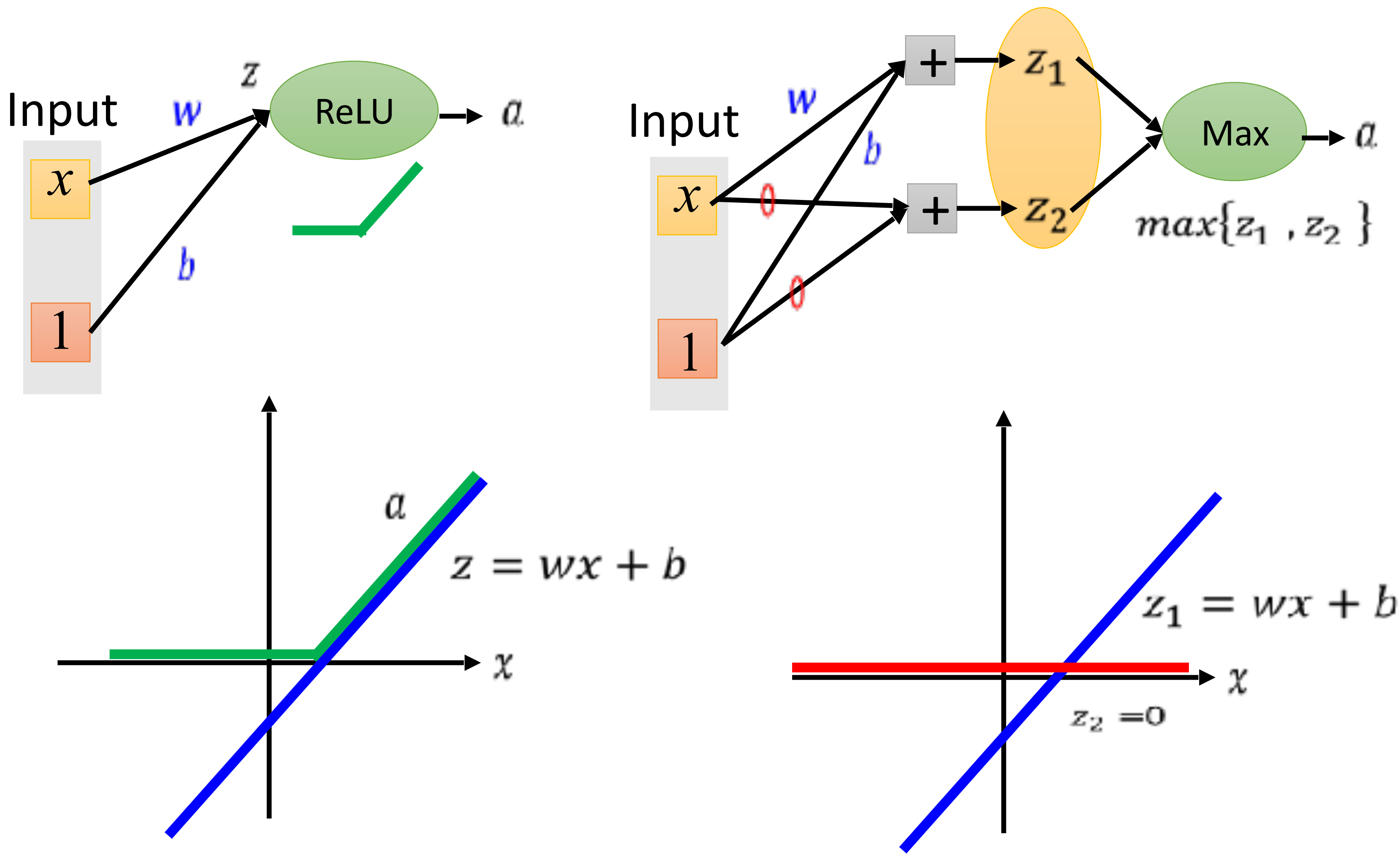
ReLU is a special cases of Maxout





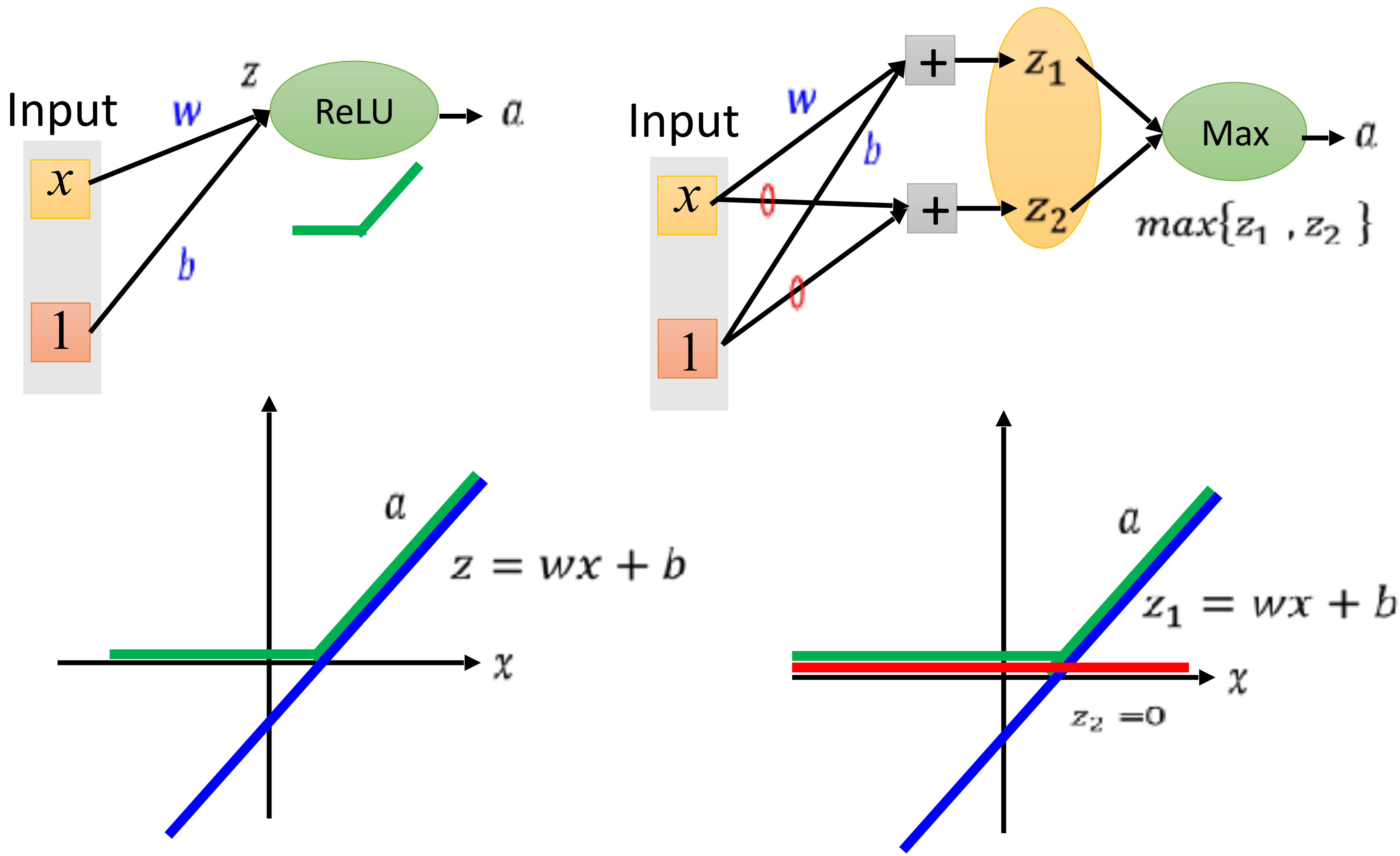
Maxout

ReLU is a special cases of Maxout



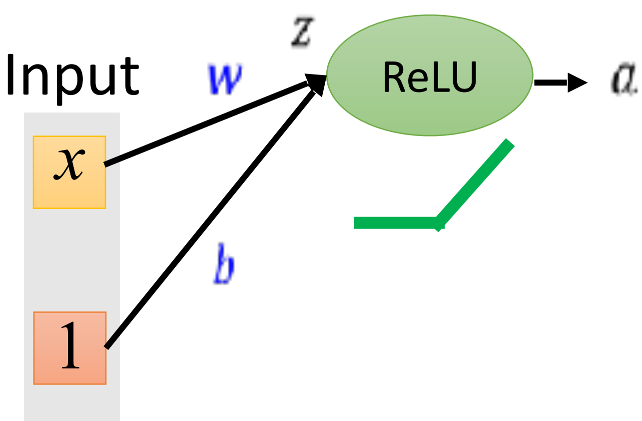
Maxout

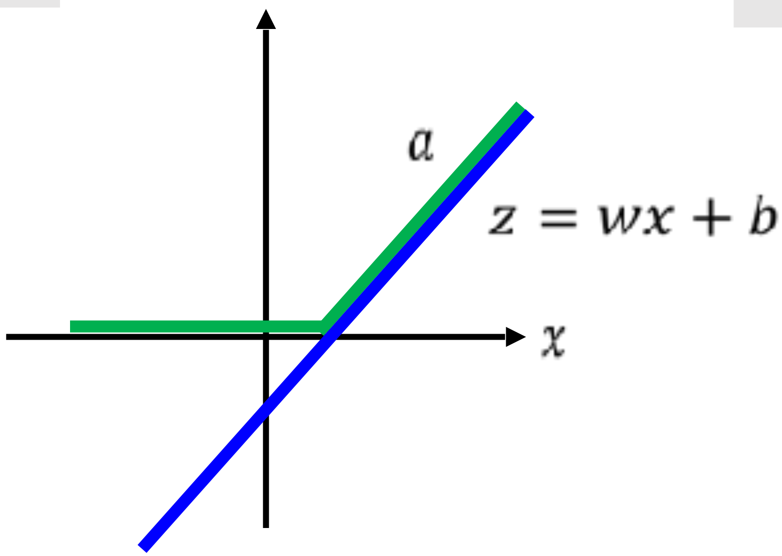
ReLU is a special cases of Maxout

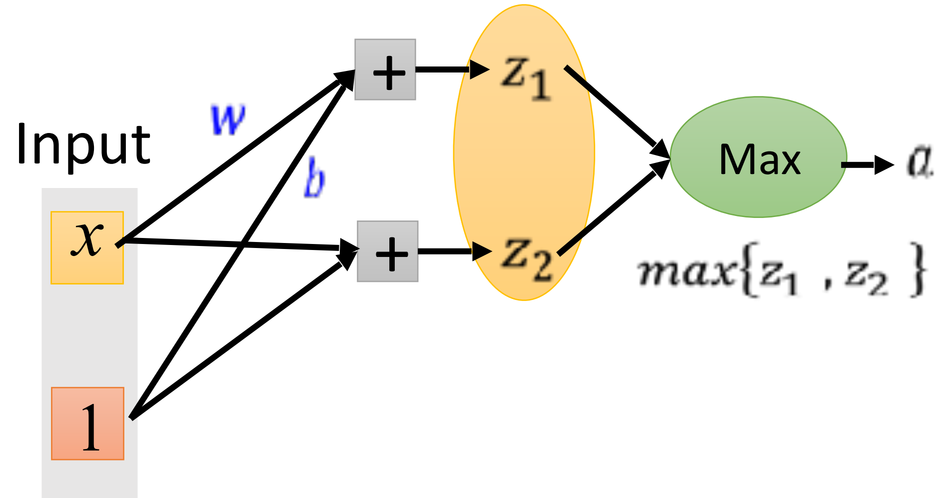


Maxout

ReLU is a special cases of Maxout

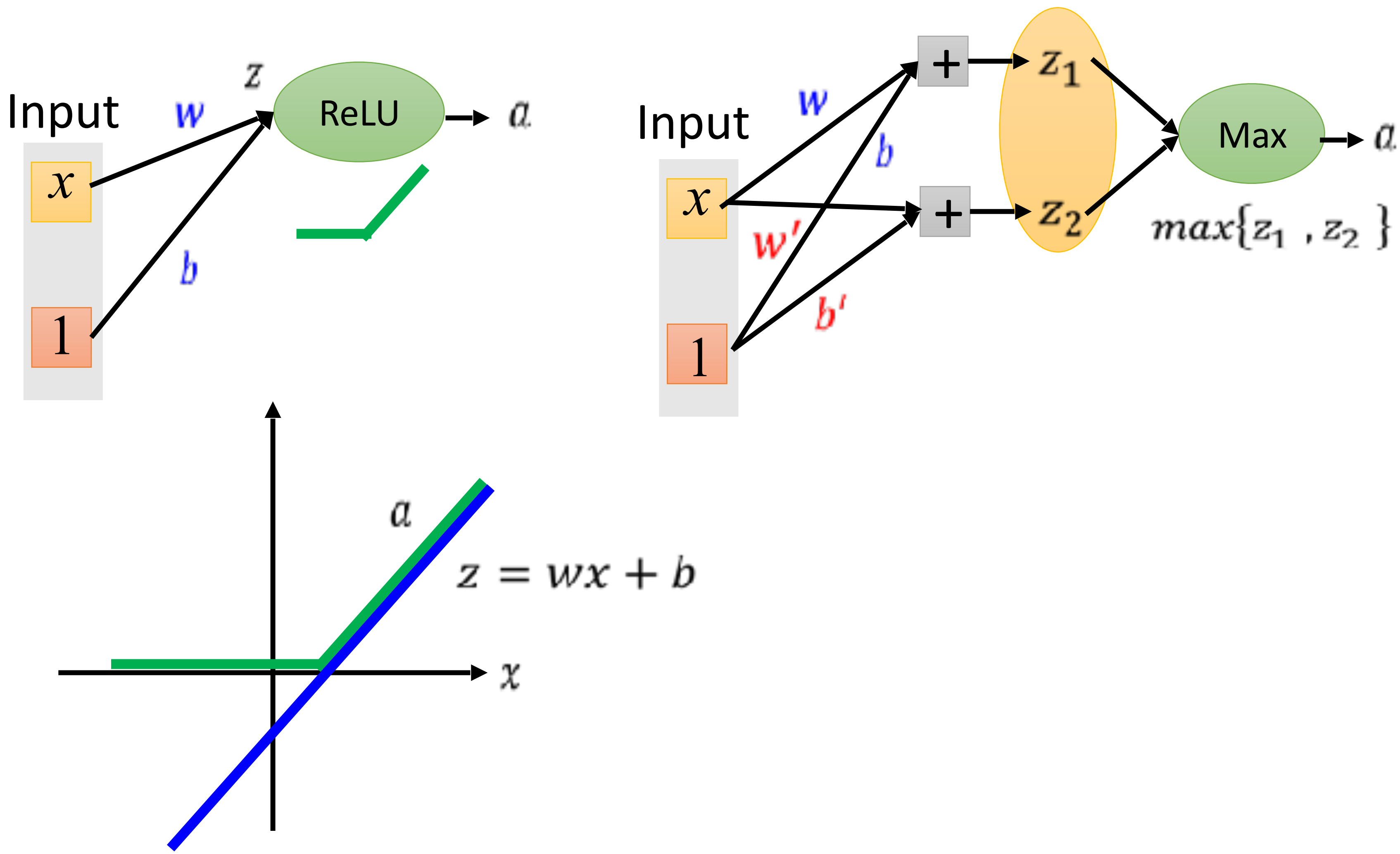






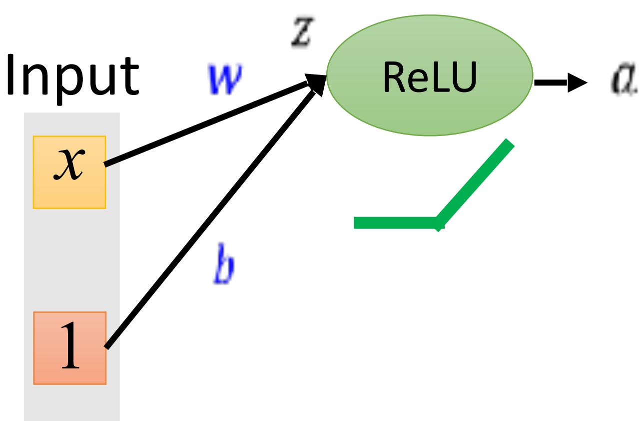
Maxout

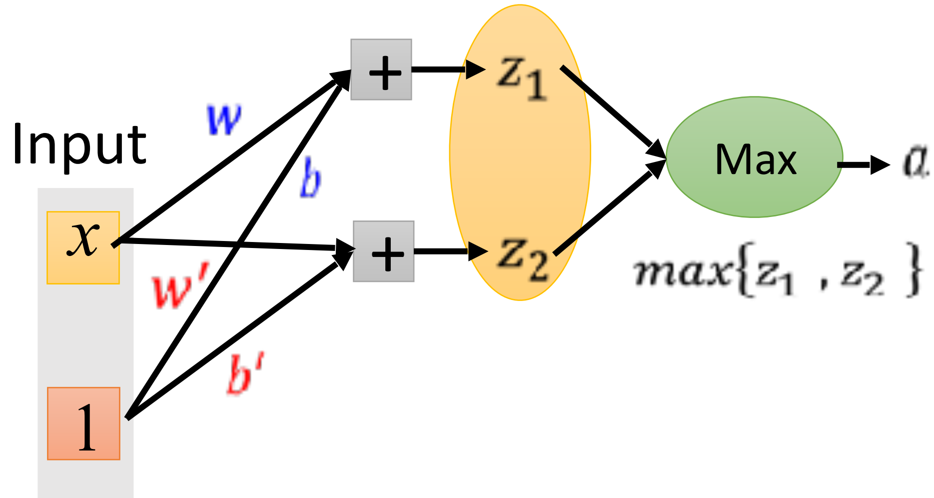
ReLU is a special cases of Maxout

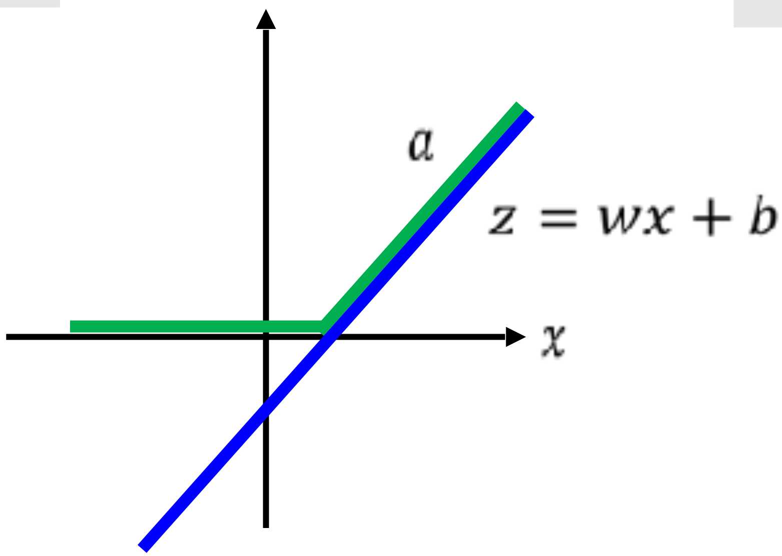


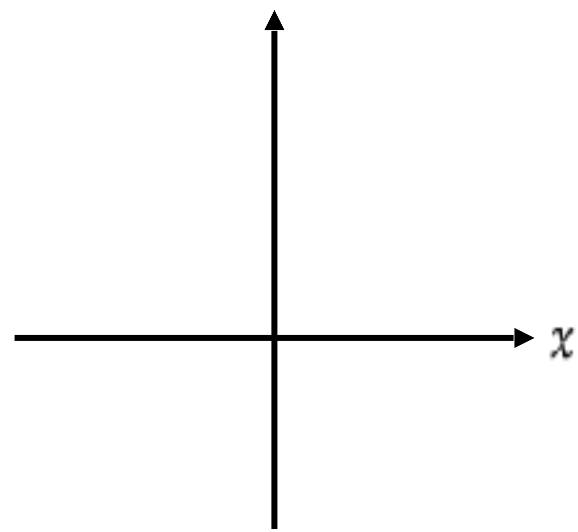
Maxout

ReLU is a special cases of Maxout



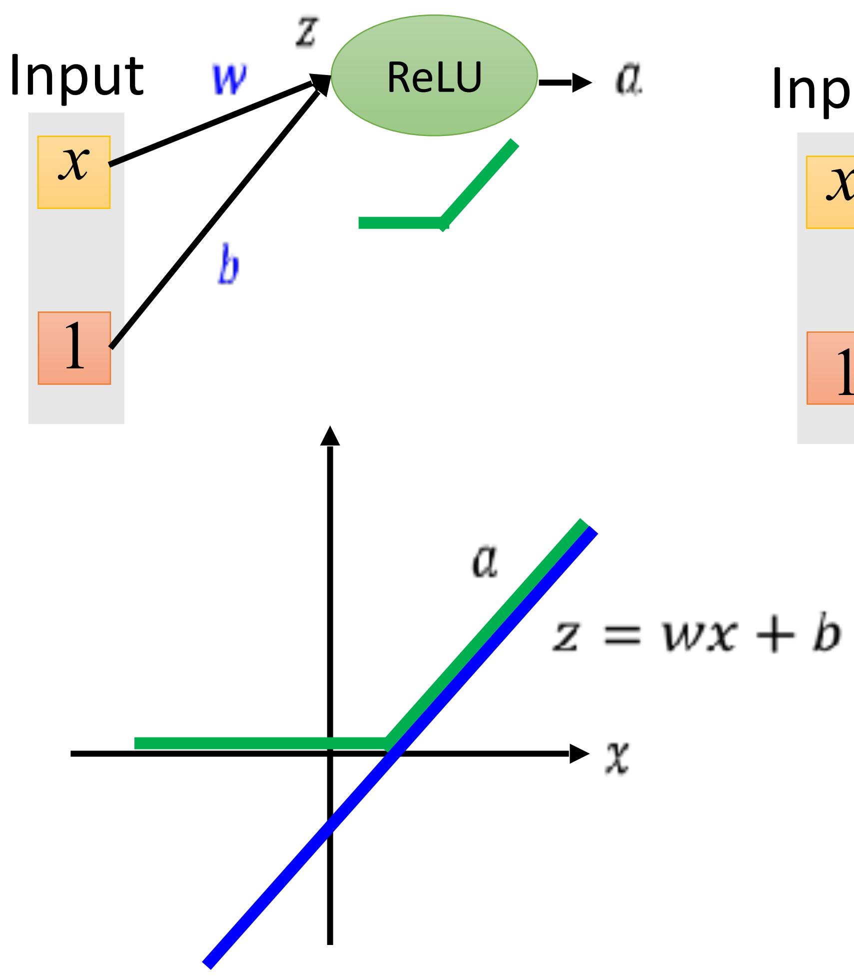


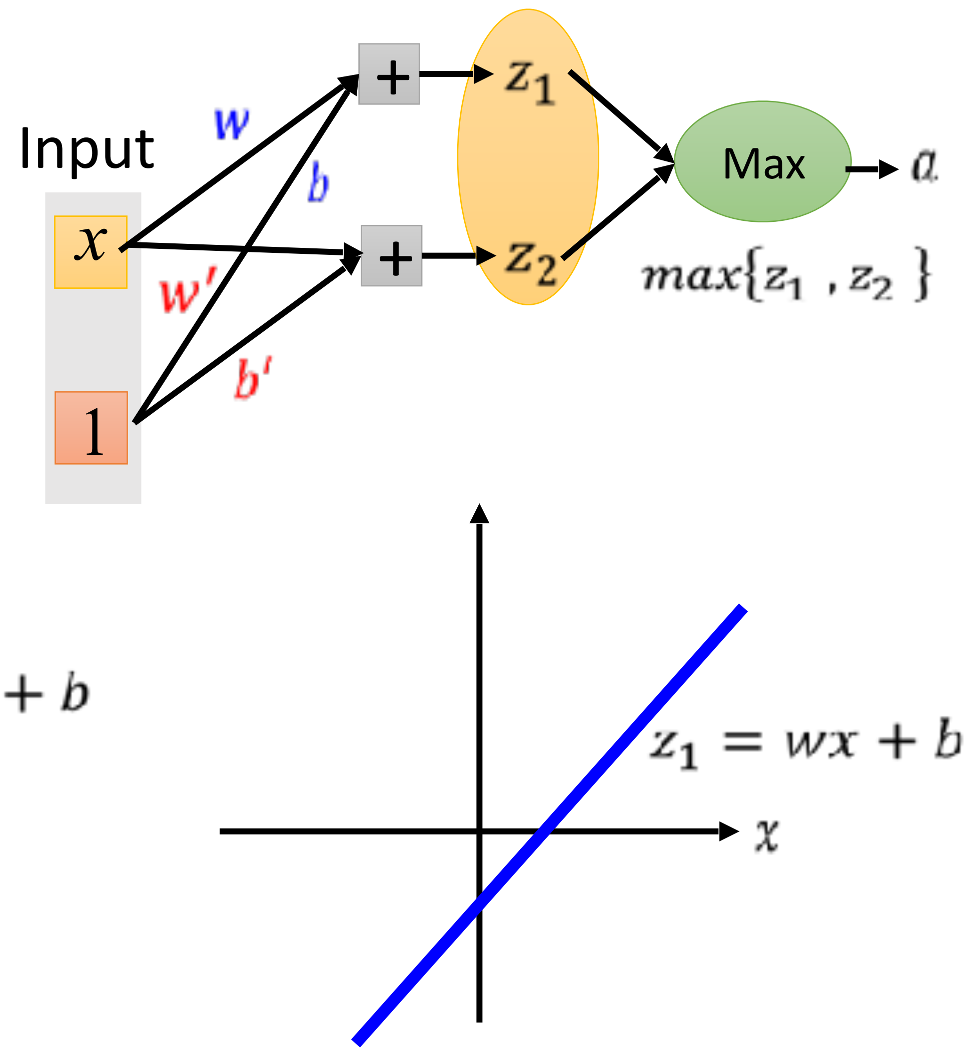




Maxout

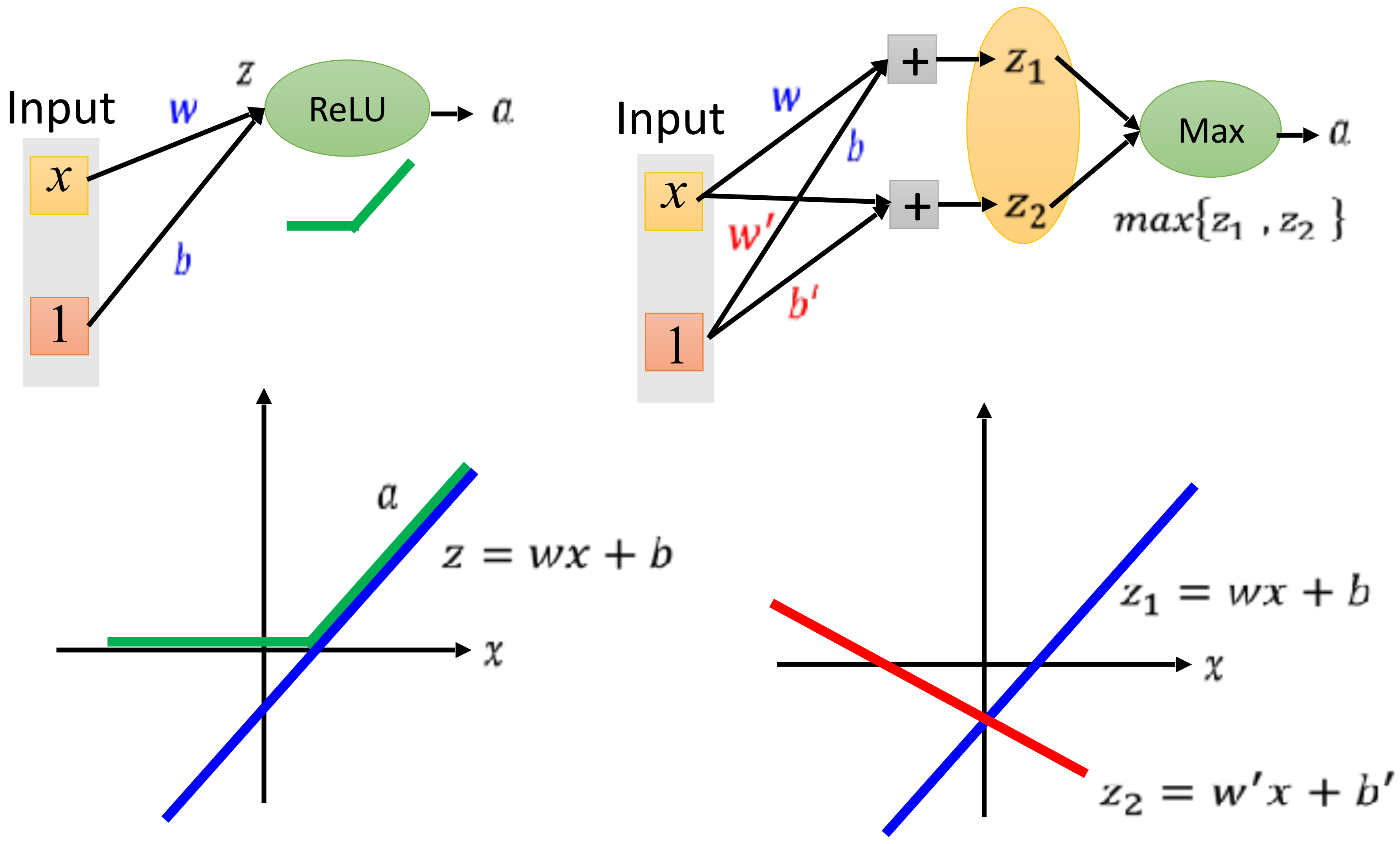
ReLU is a special cases of Maxout





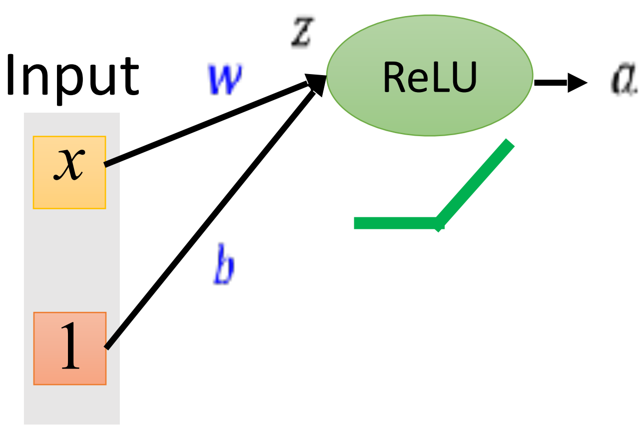
Maxout

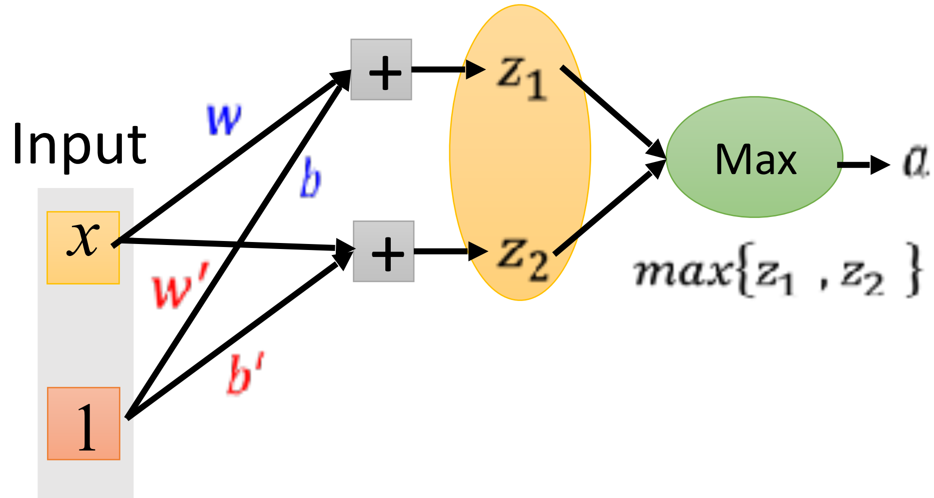
ReLU is a special cases of Maxout

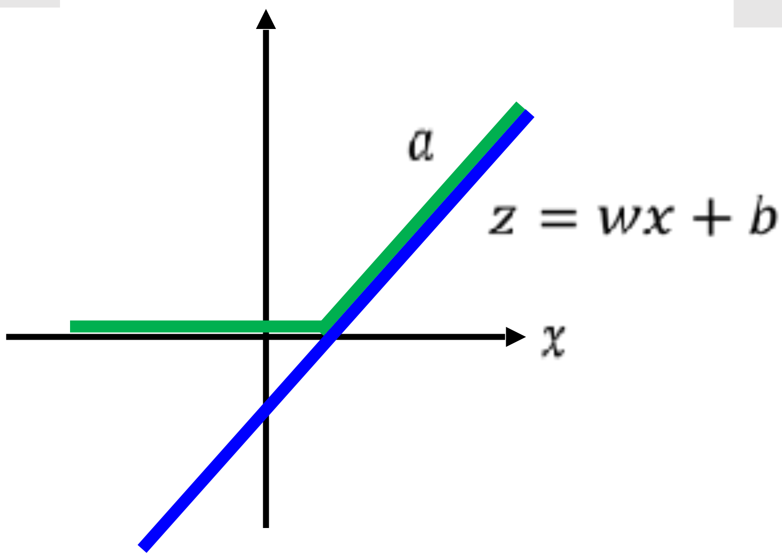


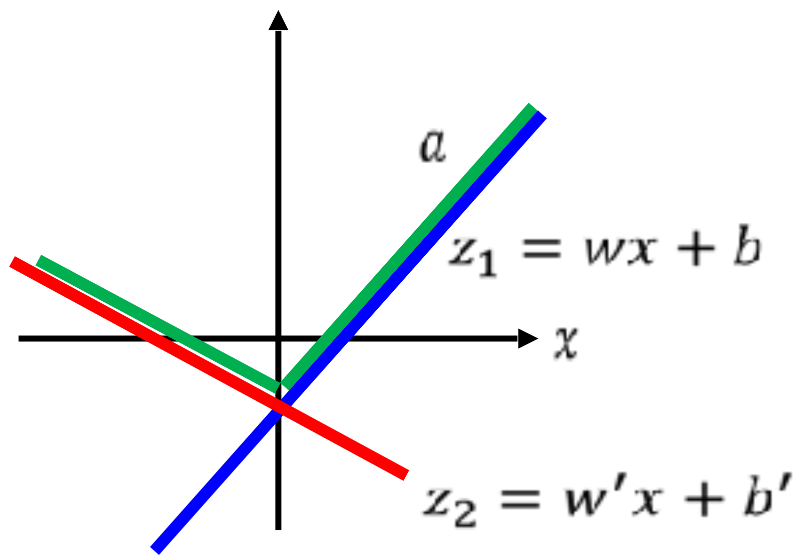
Maxout

ReLU is a special cases of Maxout



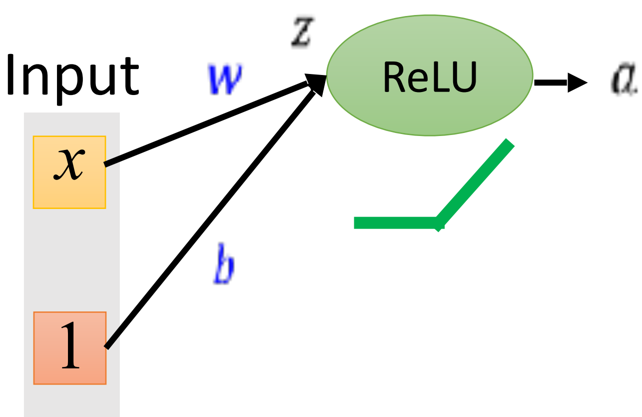


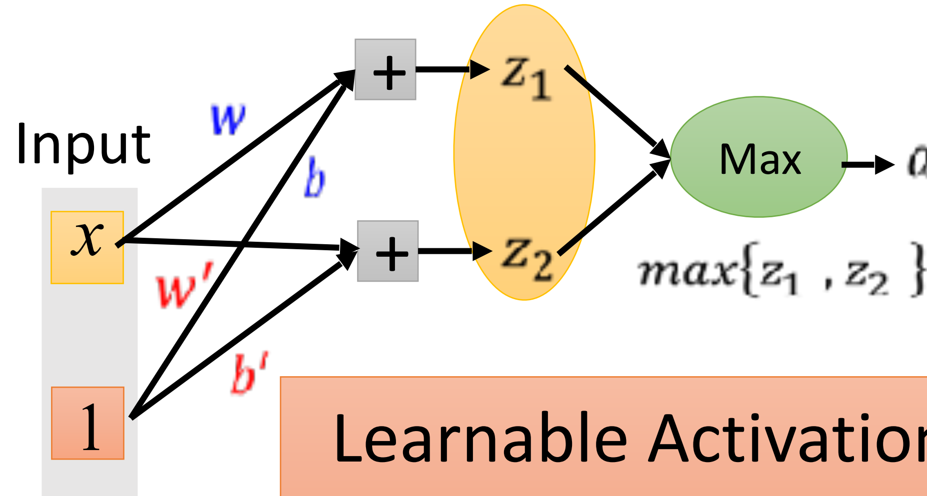




Maxout

ReLU is a special cases of Maxout





Learnable Activation Function