

Introduction to DNN: Modeling 2D and Sequential Data using CNN and RNN

國立清華大學
孫民教授



VSLab

Deep Learning Tutorial

李宏毅

Hung-yi Lee

http://www.slideshare.net/tw_dsconf/ss-62245351

Outline

Lecture I: Introduction of Deep Learning



Lecture II: ~~Tips for Training Deep Neural Network~~

Intro to Convolutional Neural Network



Lecture III: Deep Learning with Memory

Lecture I: Introduction of Deep Learning

Outline of Lecture 1

Three Steps for Deep Learning

- The three steps can also apply on other machine learning methods

Why Deep Learning?

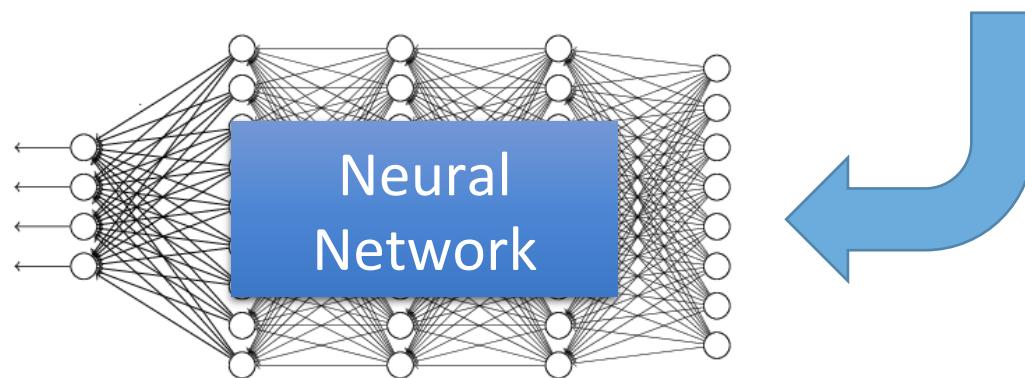
Three Steps for Deep Learning



天生的腦

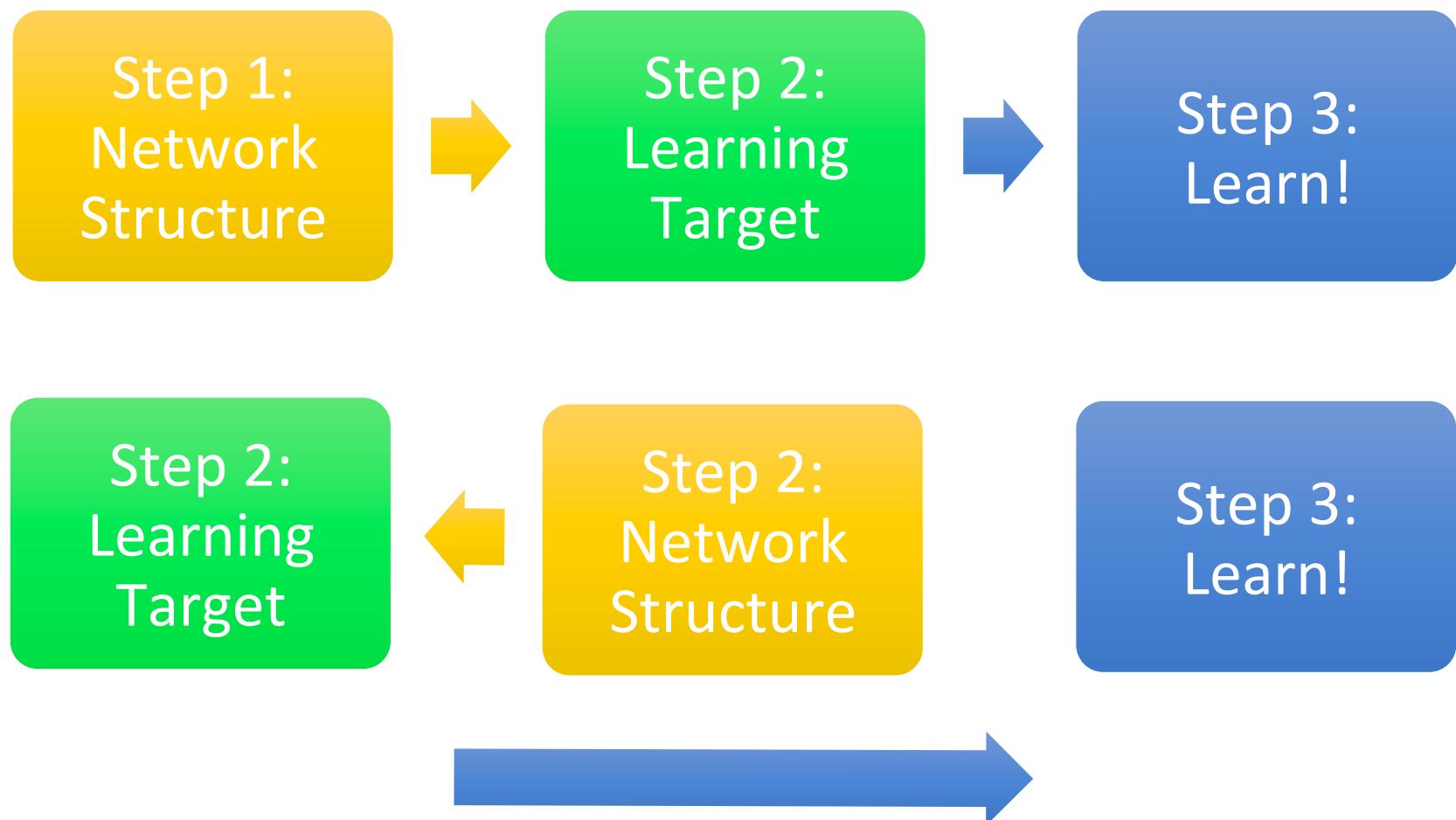


Three Steps for Deep Learning



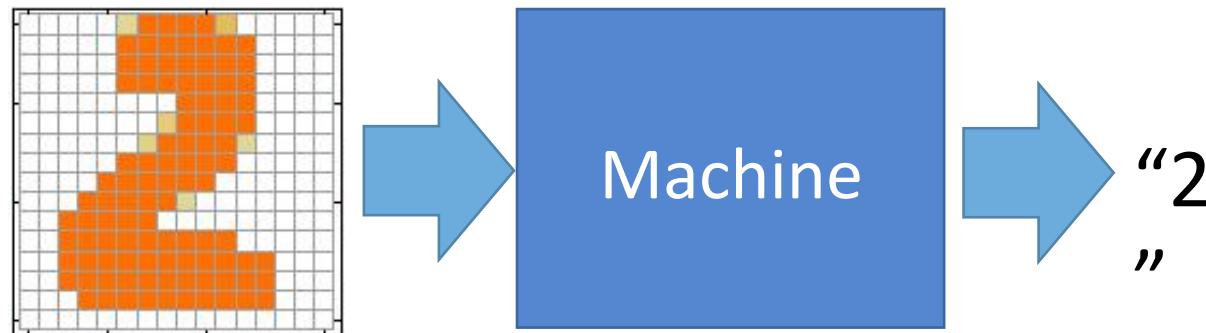
It can do image recognition, speech recognition, playing Go, etc. according to the learning target.

Three Steps for Deep Learning



Example Application

- Handwriting Digit Recognition



“Hello world” for deep learning

Data: <http://yann.lecun.com/exdb/mnist/>

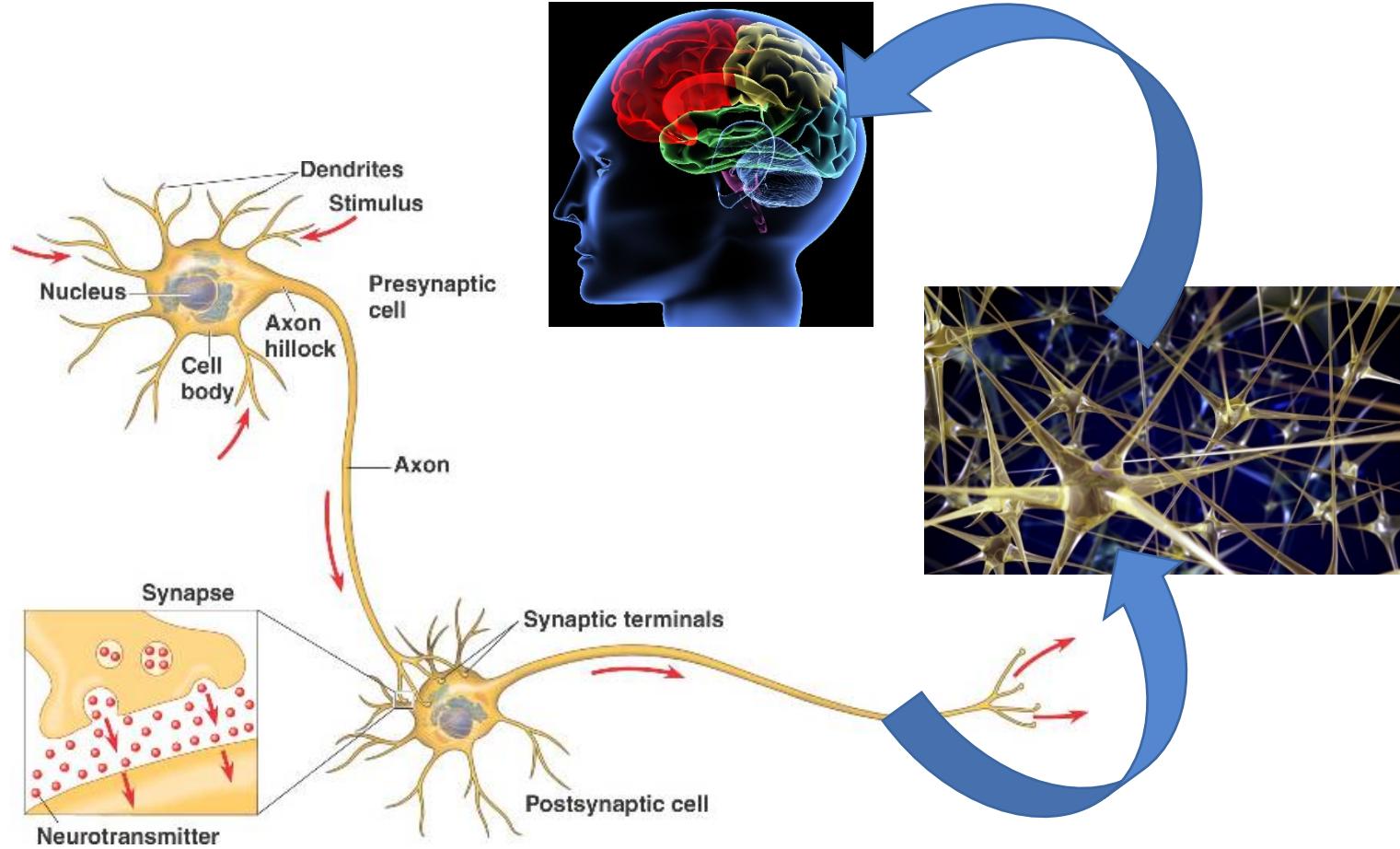
Three Steps for Deep Learning



天生的腦



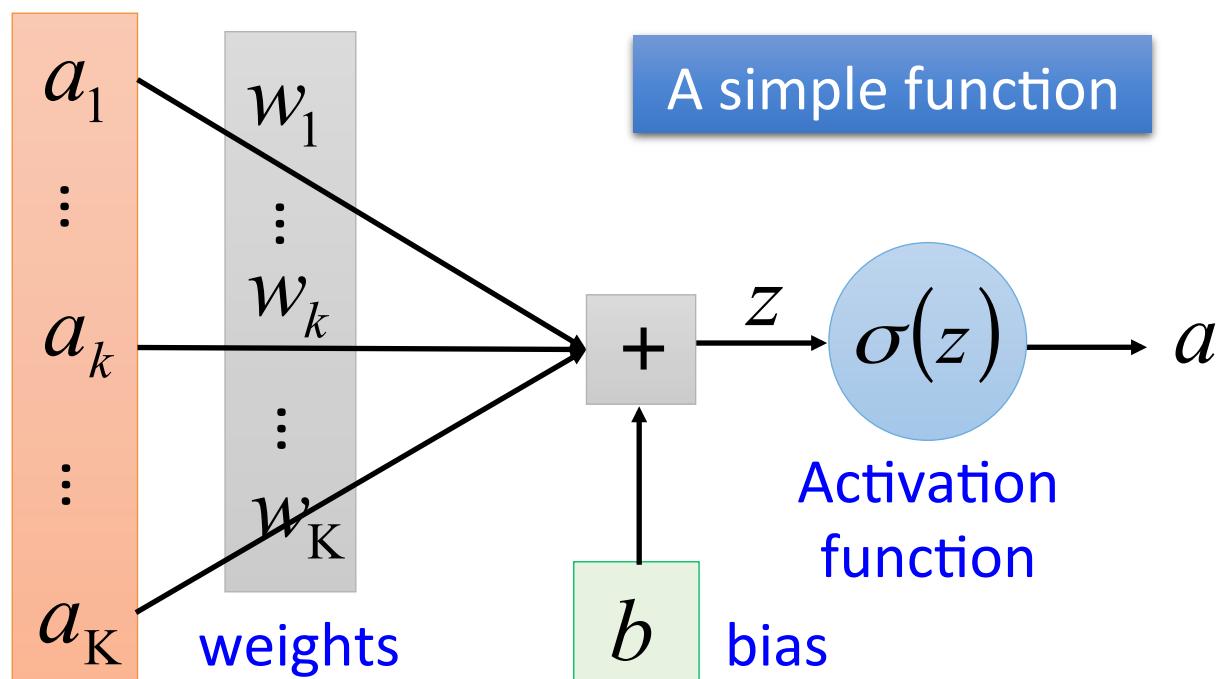
Human Brains



Neural Network

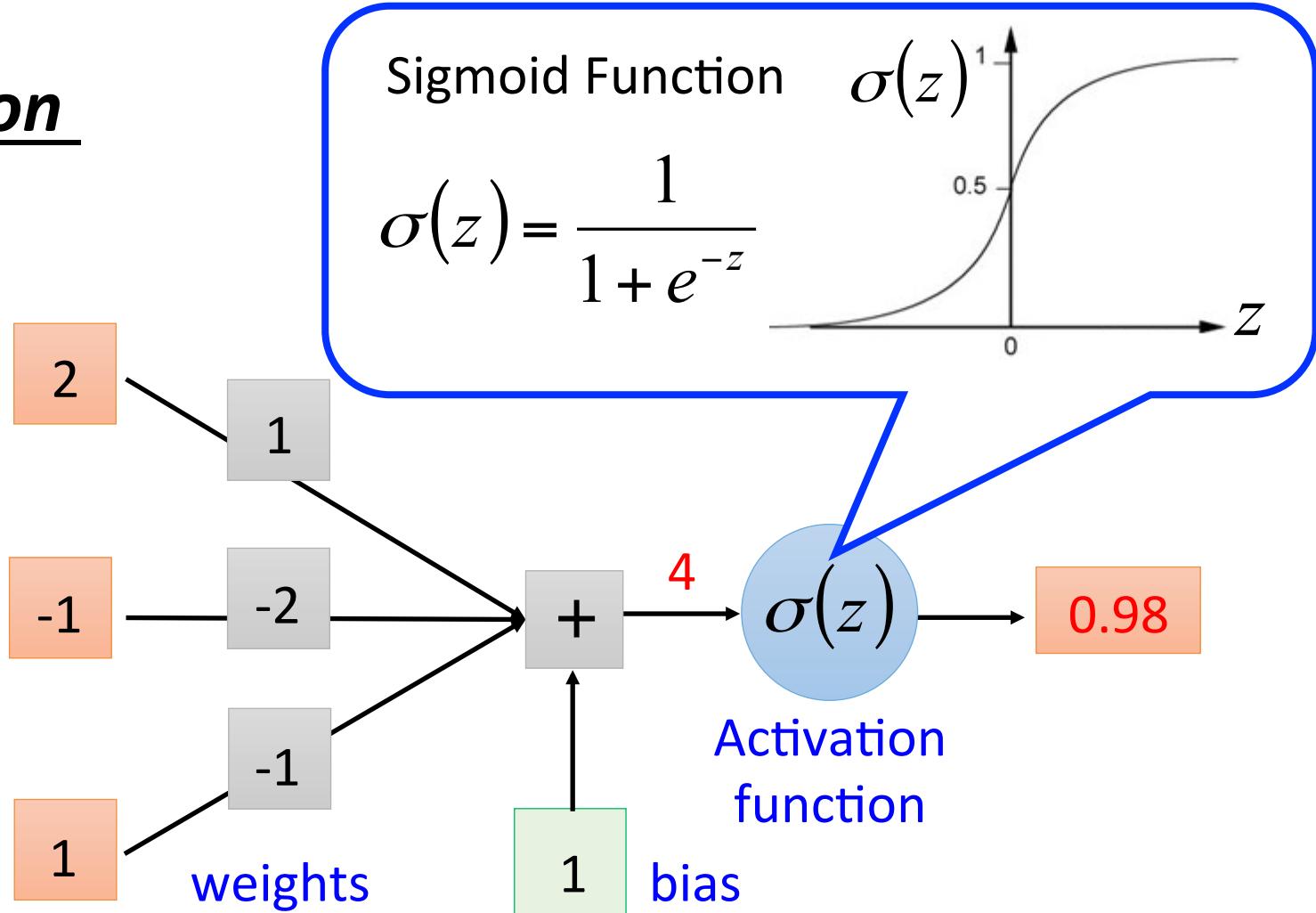
Neuron (a building block)

$$z = a_1 w_1 + \cdots + a_k w_k + \cdots + a_K w_K + b$$

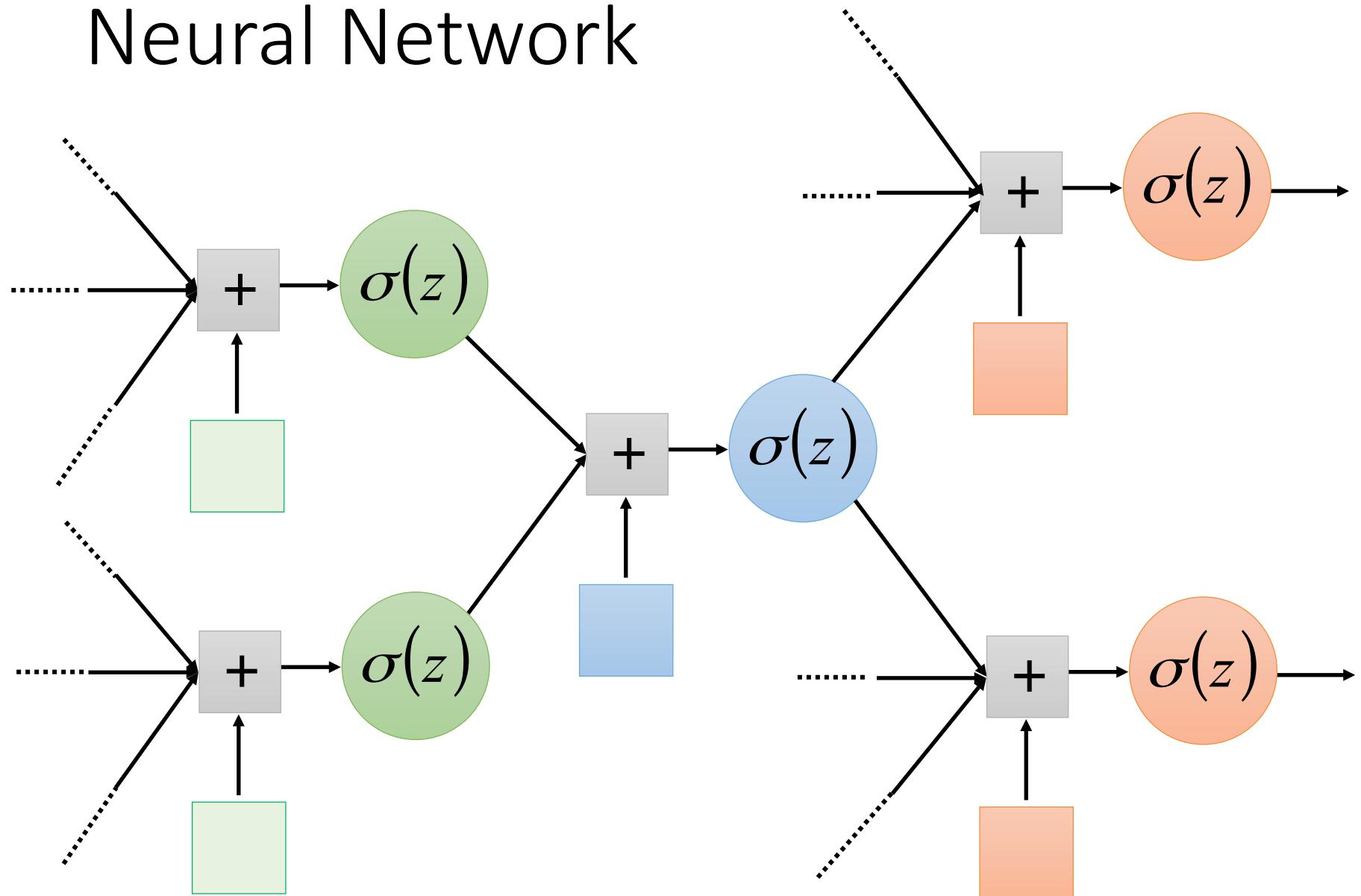


Neural Network

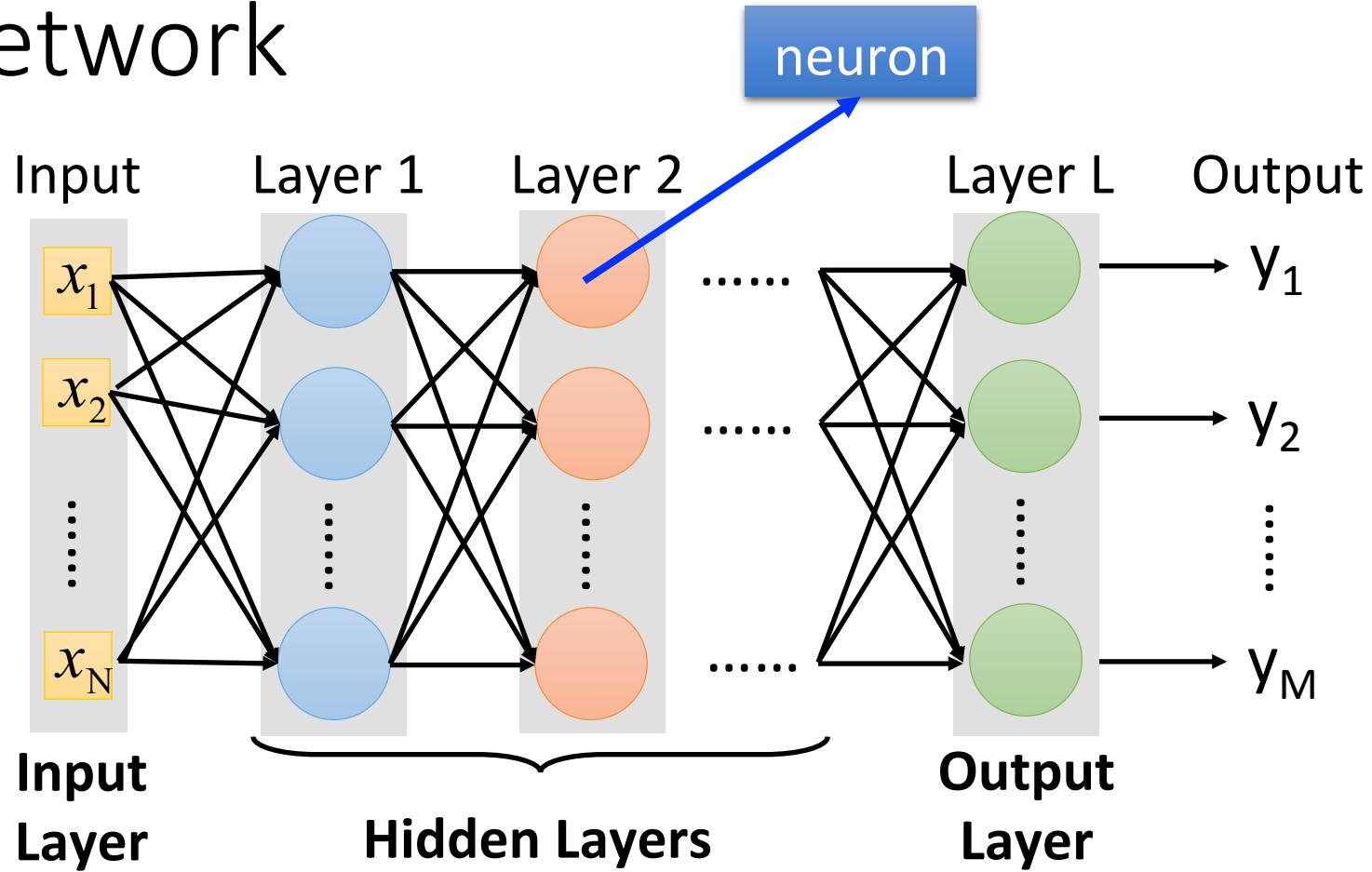
Neuron



Neural Network



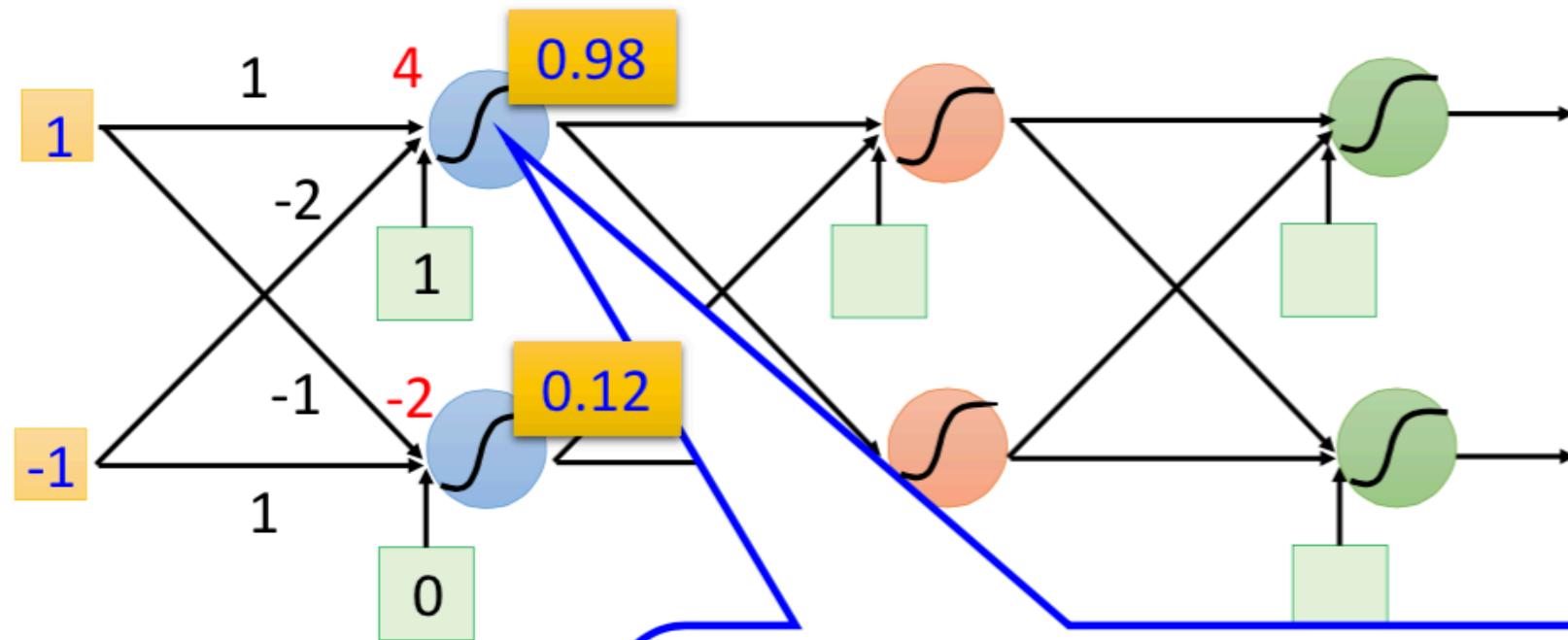
Fully Connect Feedforward Network



Deep means many hidden layers

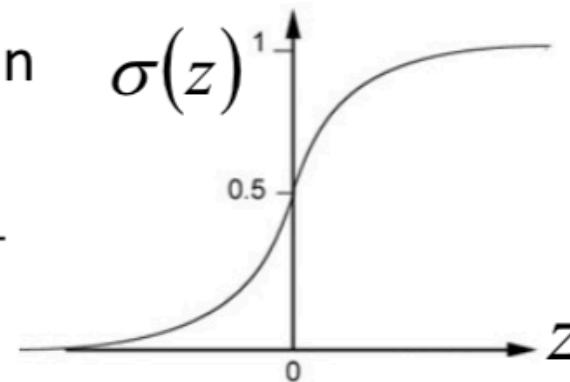
You can connect the neurons by other ways you like 😊

Fully Connect Feedforward Network

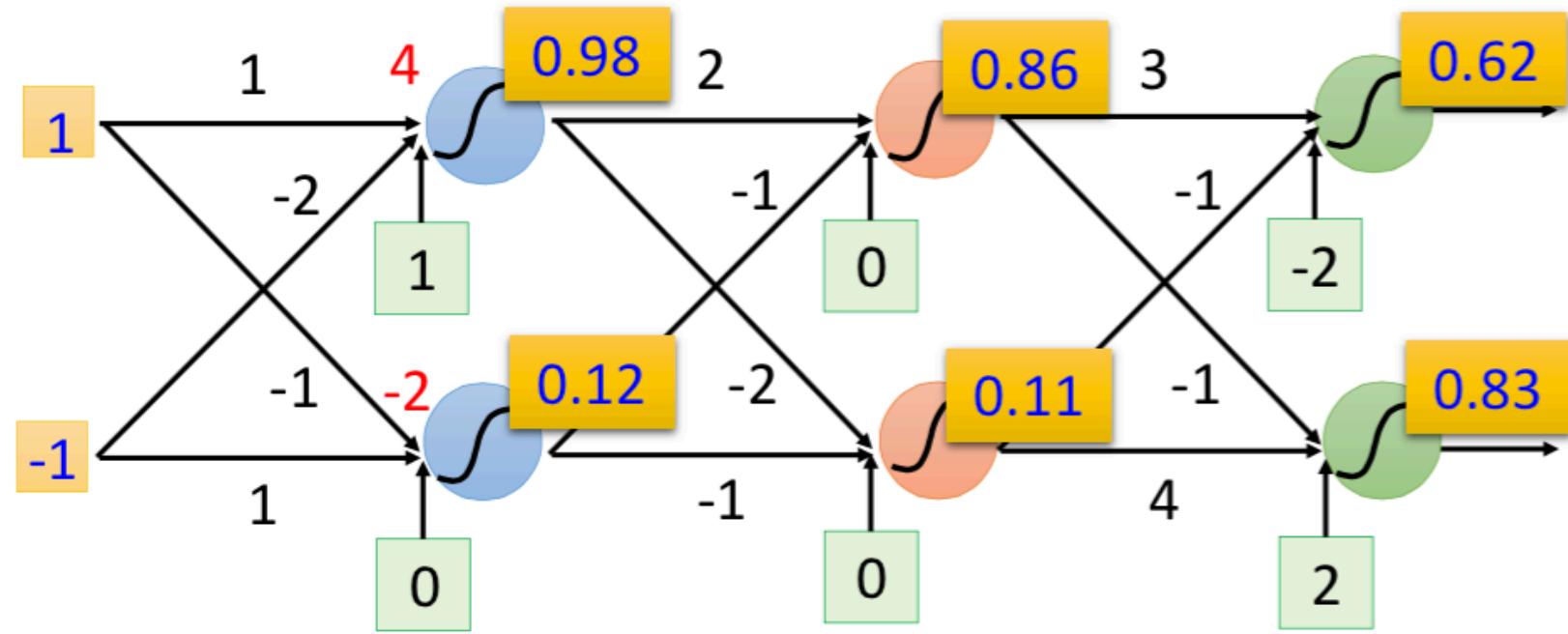


Sigmoid Function

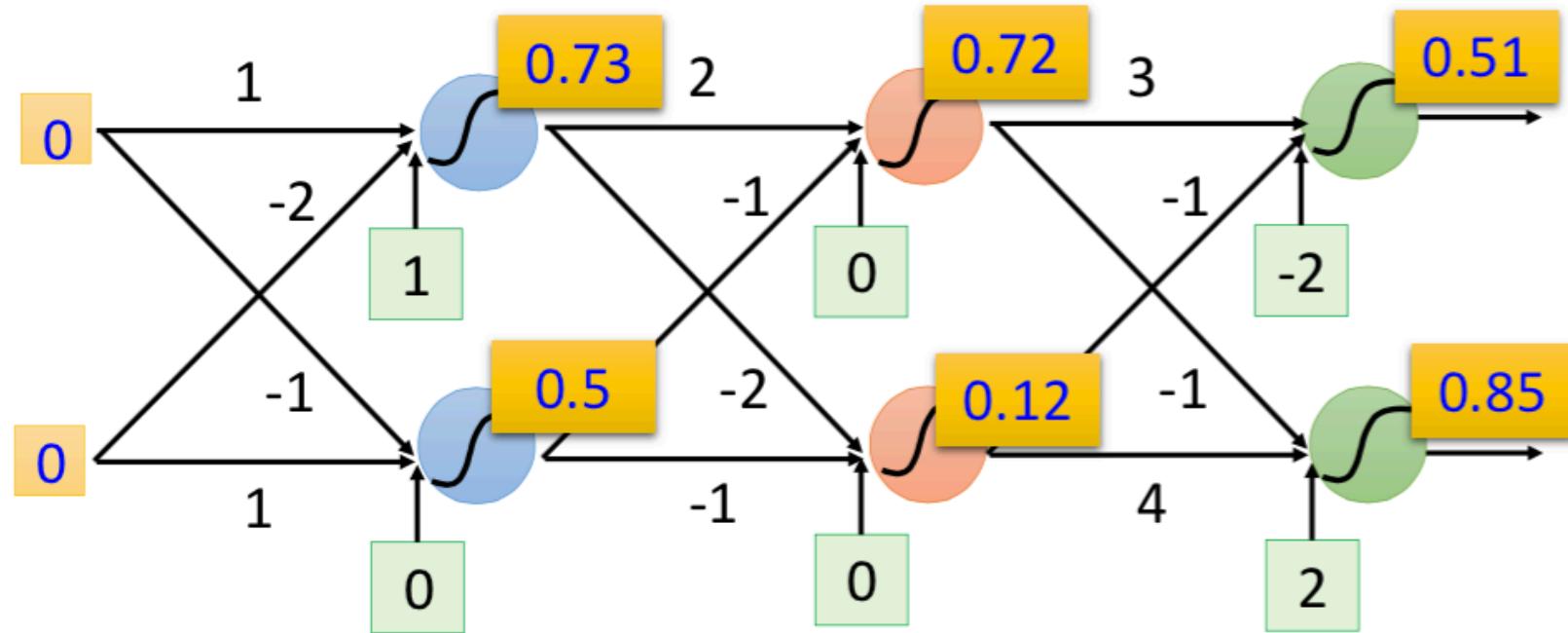
$$\sigma(z) = \frac{1}{1 + e^{-z}}$$



Network



Fully Connect Feedforward Network



Network is a function.

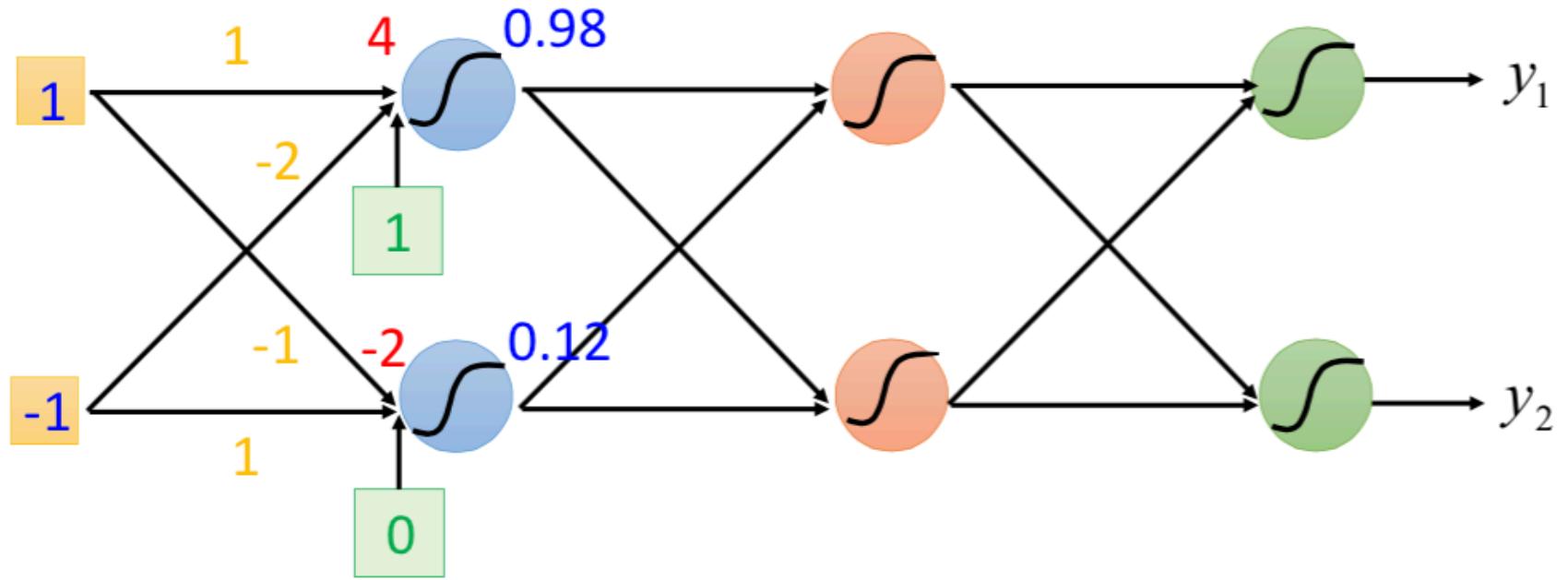
Input vector, output vector

$$f \begin{pmatrix} 1 \\ -1 \end{pmatrix} = \begin{bmatrix} 0.62 \\ 0.83 \end{bmatrix} \quad f \begin{pmatrix} 0 \\ 0 \end{pmatrix} = \begin{bmatrix} 0.51 \\ 0.85 \end{bmatrix}$$

Weights and biases are network parameters θ

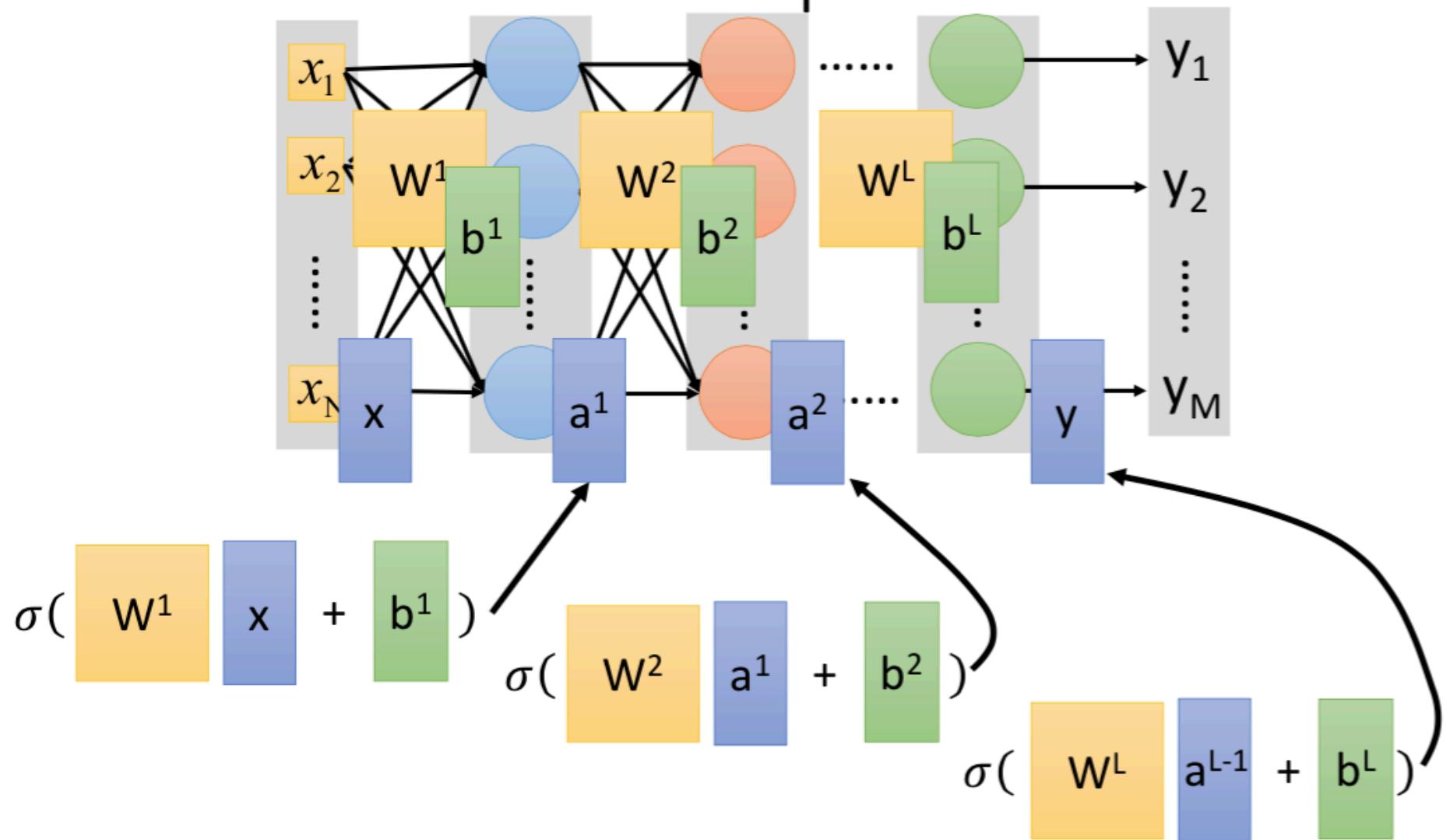
Different parameters θ define different functions

Network - Matrix Operation

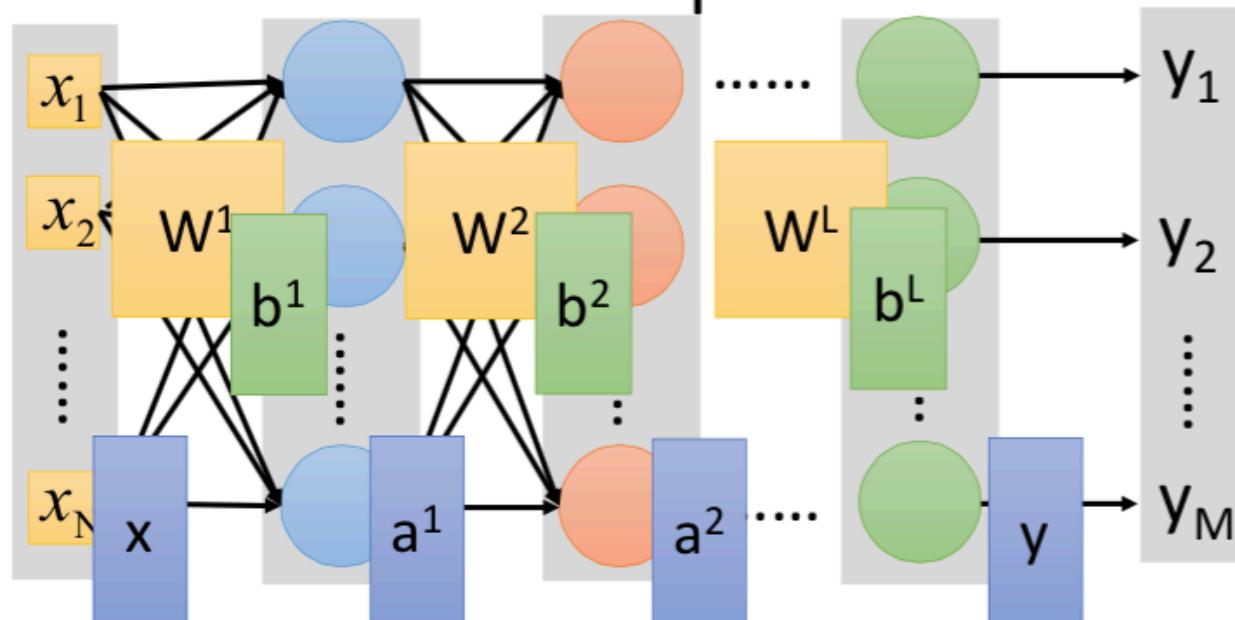


$$\sigma(\underbrace{\begin{bmatrix} 1 & -2 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix}}_{\text{Red bracket}}) = \begin{bmatrix} 0.98 \\ 0.12 \end{bmatrix}$$

Fully Connect Feedforward Network - Matrix Operation



Fully Connect Feedforward Network - Matrix Operation



$$y = f(x)$$

Using parallel computing techniques to speed up matrix operation

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

Output Layer

- Softmax layer as the output layer

Ordinary Layer

$$z_1 \rightarrow \sigma \rightarrow y_1 = \sigma(z_1)$$

$$z_2 \rightarrow \sigma \rightarrow y_2 = \sigma(z_2)$$

$$z_3 \rightarrow \sigma \rightarrow y_3 = \sigma(z_3)$$

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

May not be easy to interpret

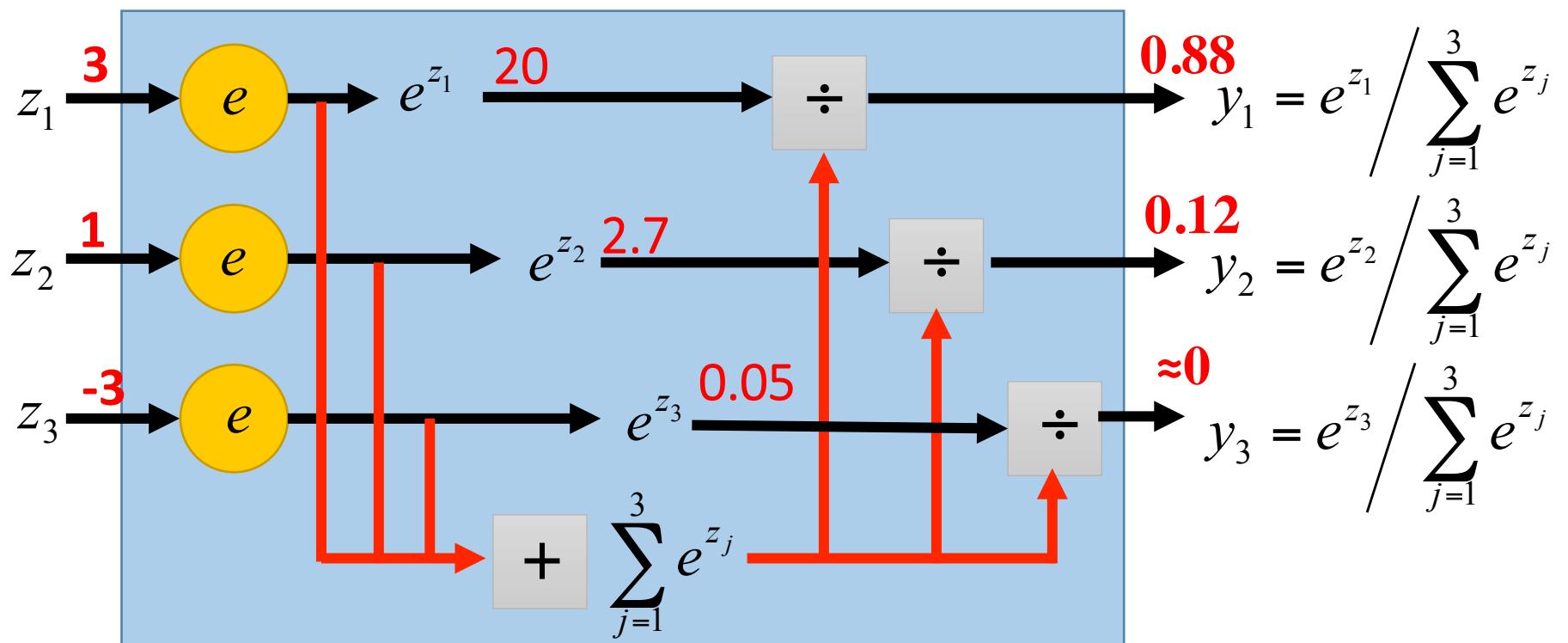
Output Layer

- Softmax layer as the output layer

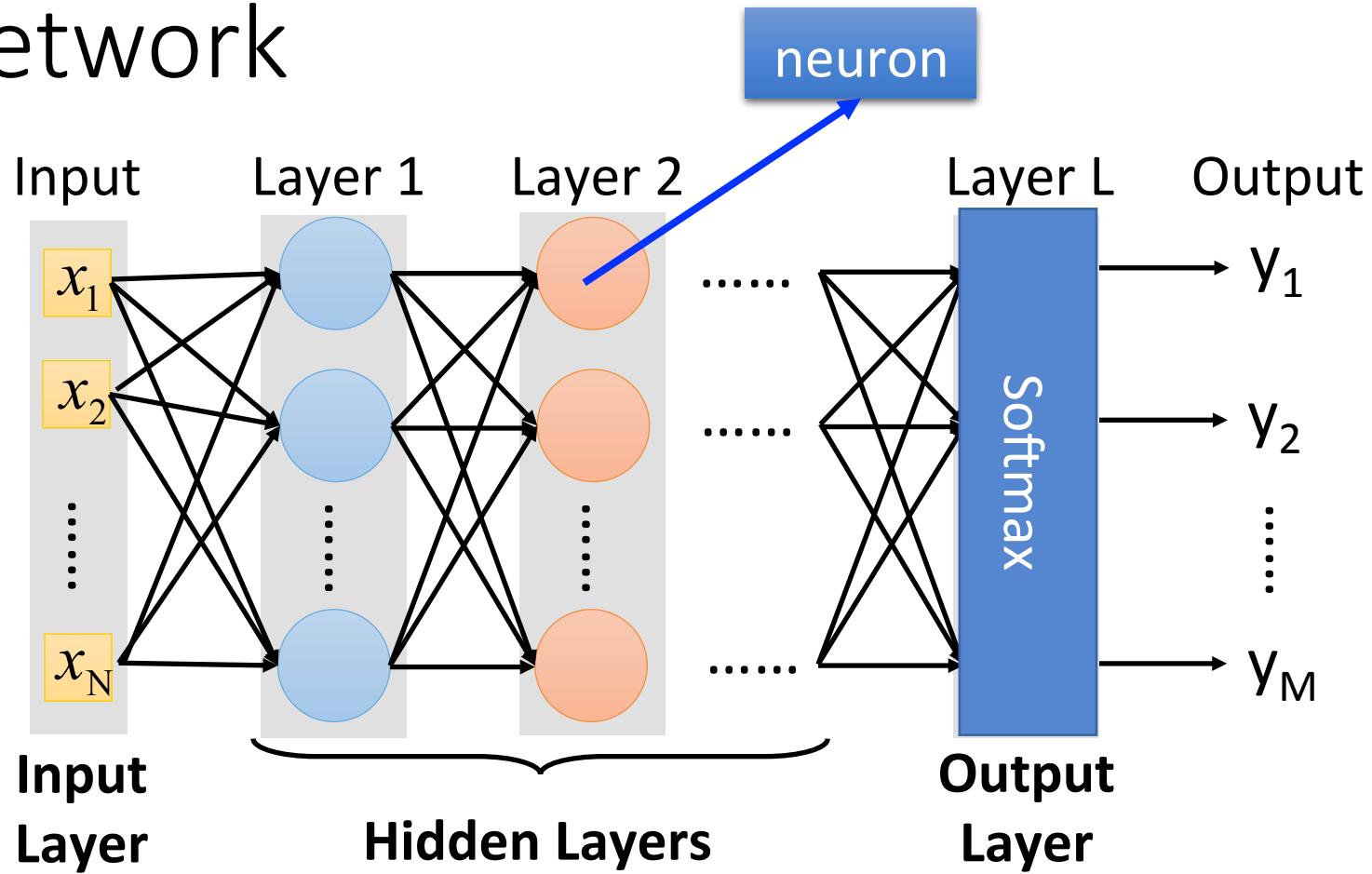
Probability:

- $0 < y_i < 1$
- $\sum_i y_i = 1$

Softmax Layer



Fully Connect Feedforward Network



Q: How many layers? How many neurons for each layer?

Ans: I don't know.

Three Steps for Deep Learning

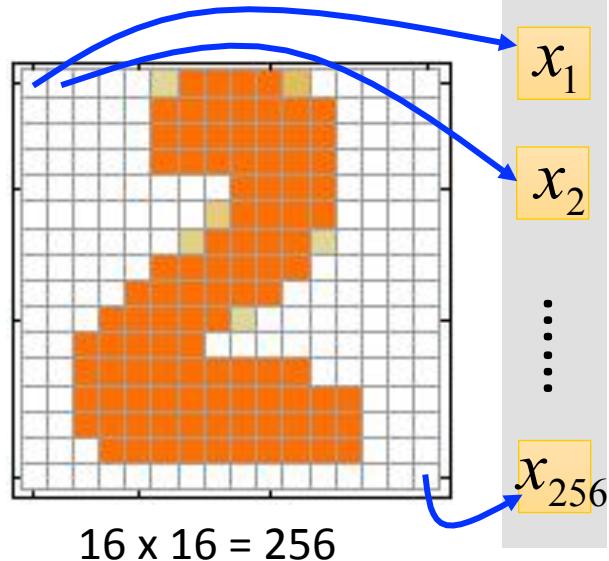


天生的腦

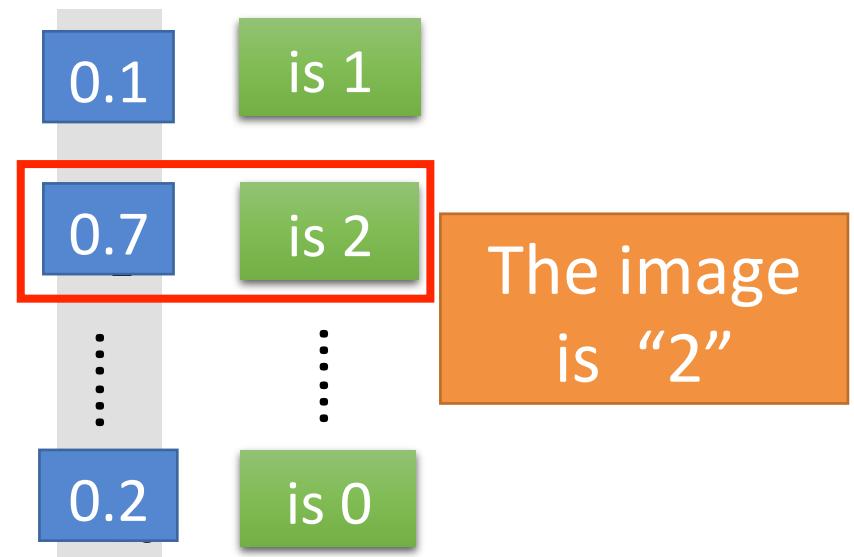


Back to Example Application

Input



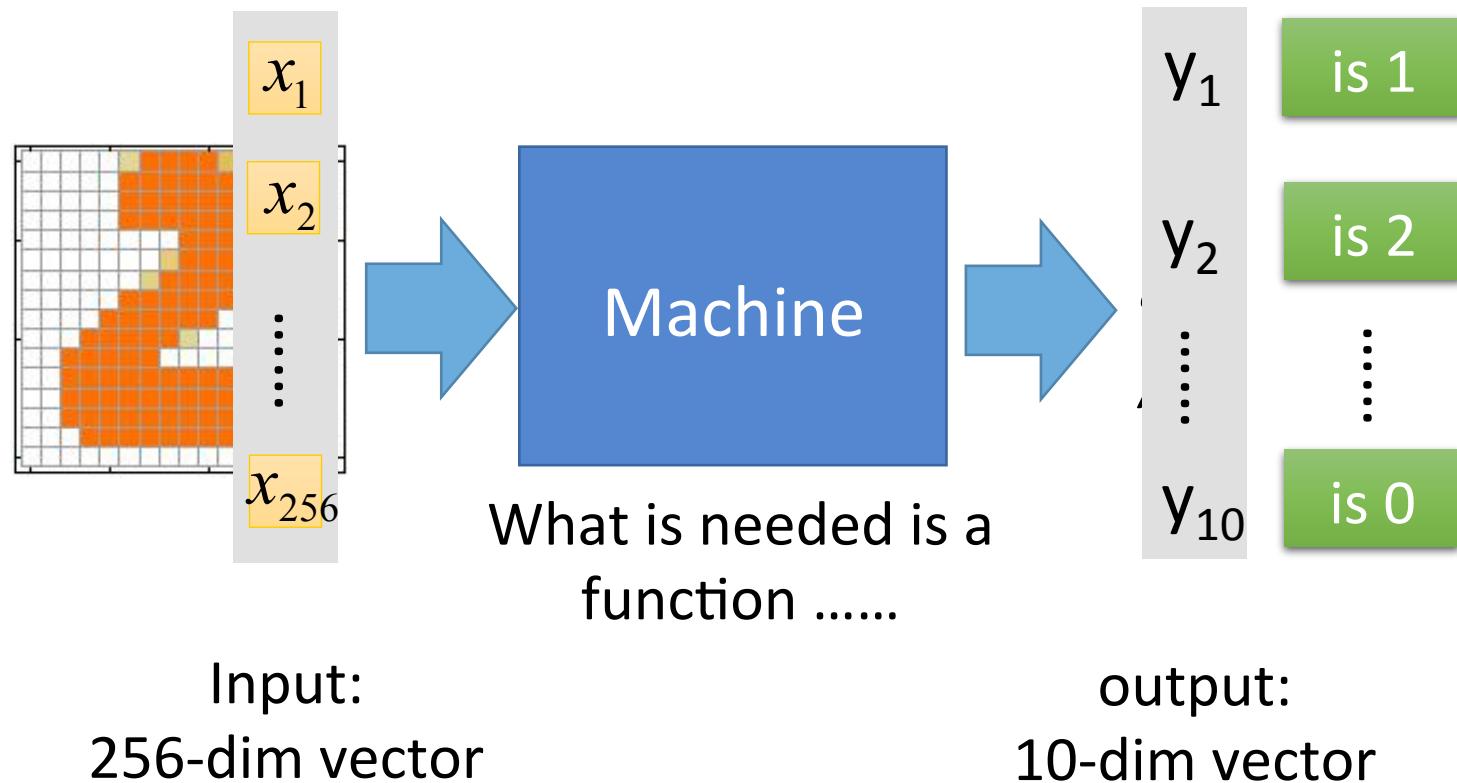
Output



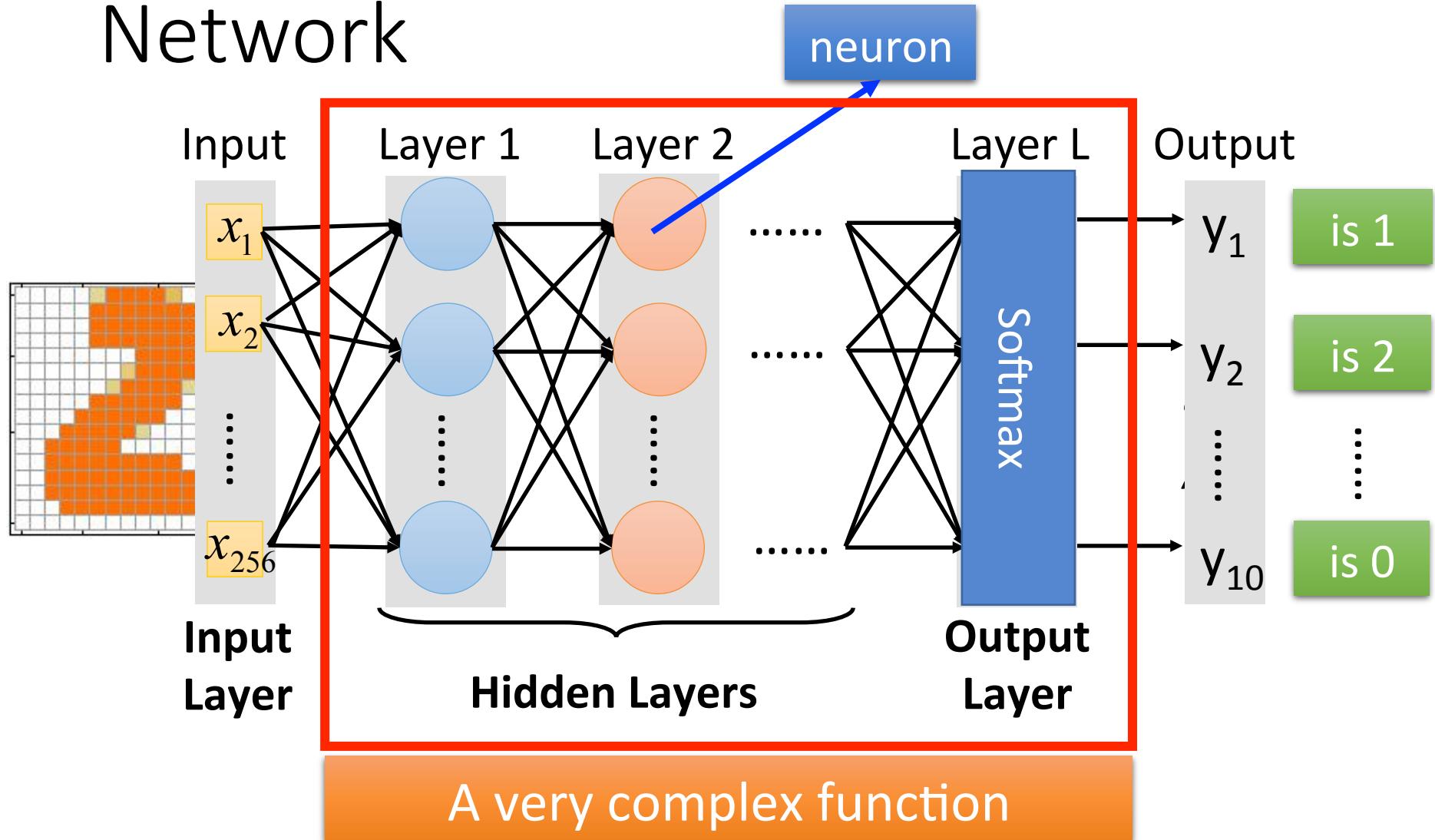
Each dimension represents the confidence of a digit.

Back to Example Application

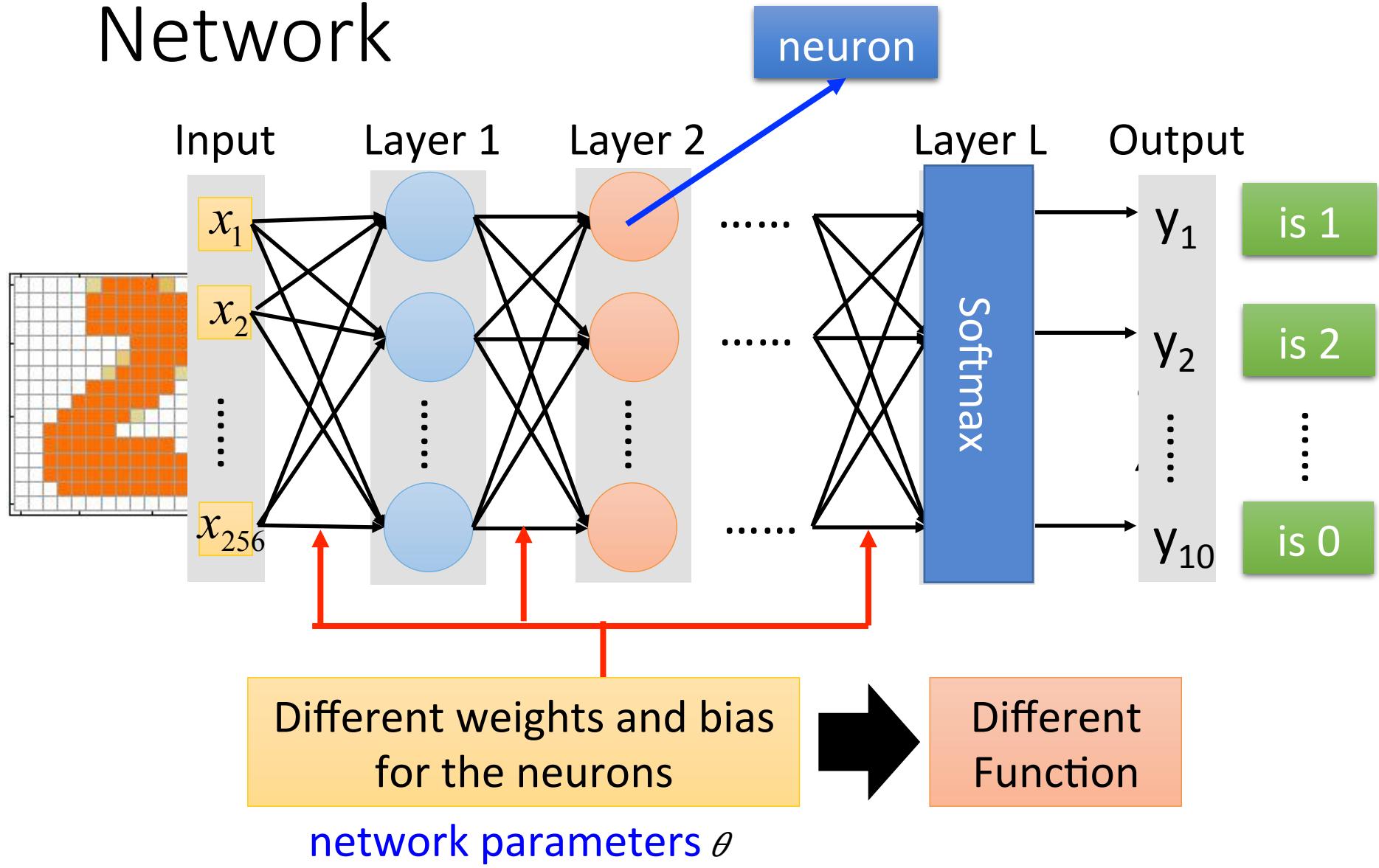
- Handwriting Digit Recognition



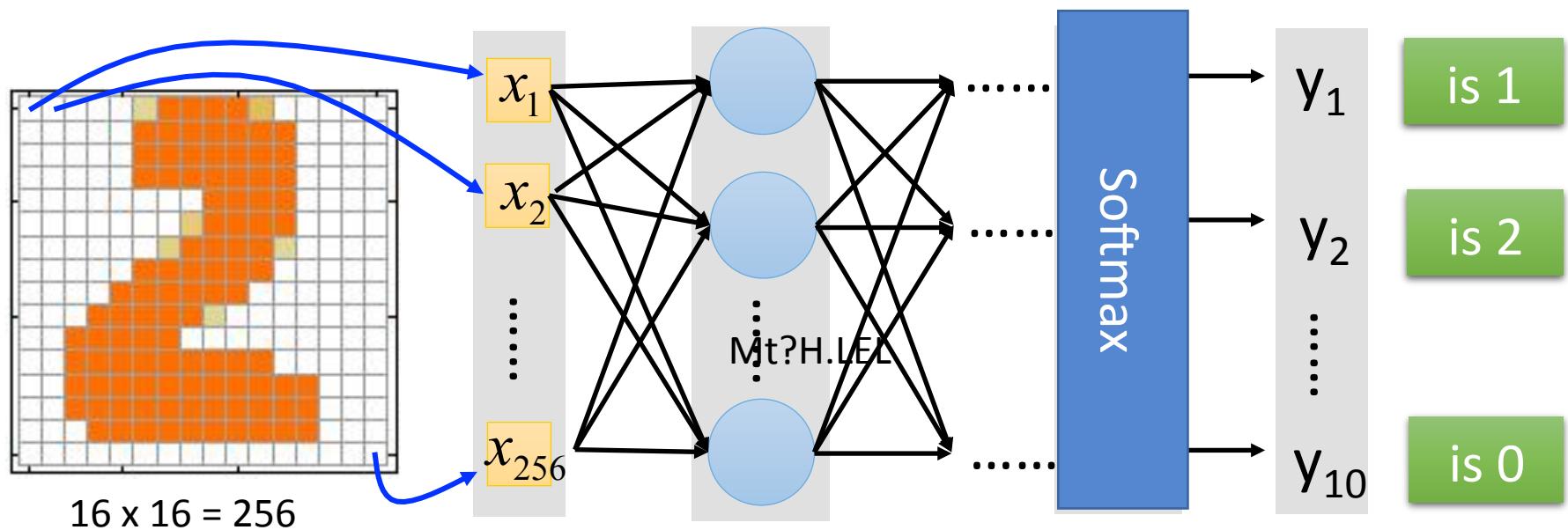
Fully Connect Feedforward Network



Fully Connect Feedforward Network



Learning Target



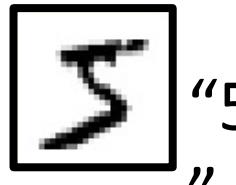
The learning target is

Input: $\rightarrow y_1$ has the maximum value

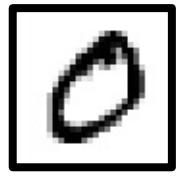
Input: $\rightarrow y_2$ has the maximum value

Training Data (Supervised Training)

- Preparing training data: images and their labels



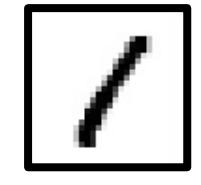
“5”



“0”



“4”



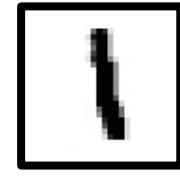
“1”



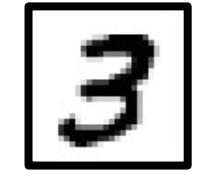
“9”



“2”



“1”

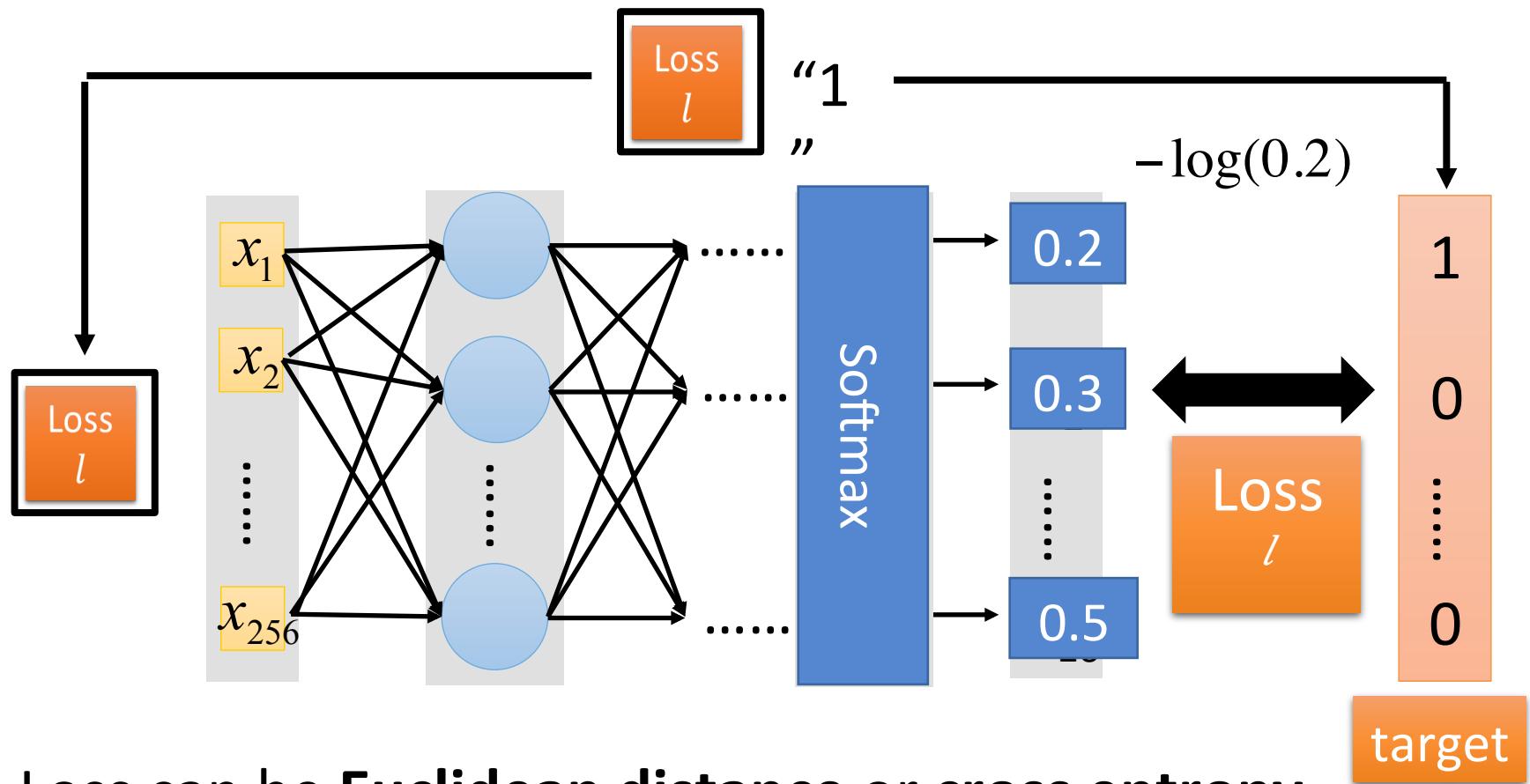


“3”

The learning target is defined on
the training data.

LOSS

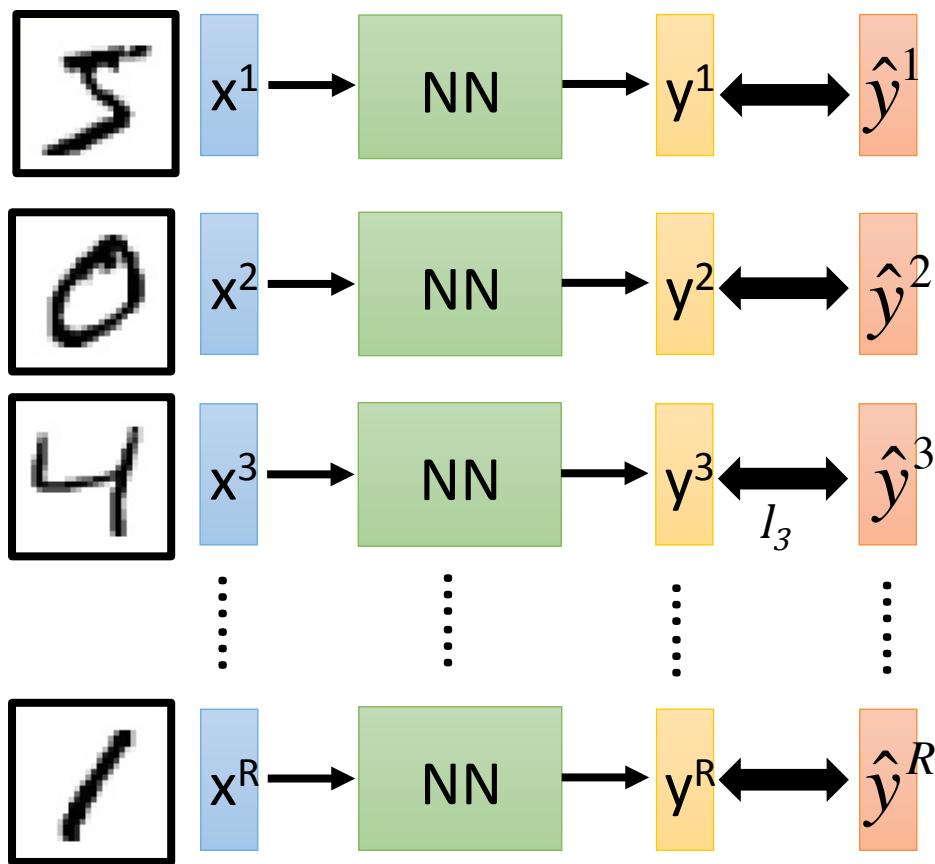
Given a set of network parameters θ , each example has a loss value.



Loss can be **Euclidean distance** or **cross entropy** between the network output and target

Total Loss

For all training data ...



Total Loss:

$$L = \sum_{r=1}^R l_r$$

Find the network parameters θ^* that minimize total loss

Next Step

Three Steps for Deep Learning



天生的腦



Find the network parameters θ^* that minimize total loss



How to Learn?

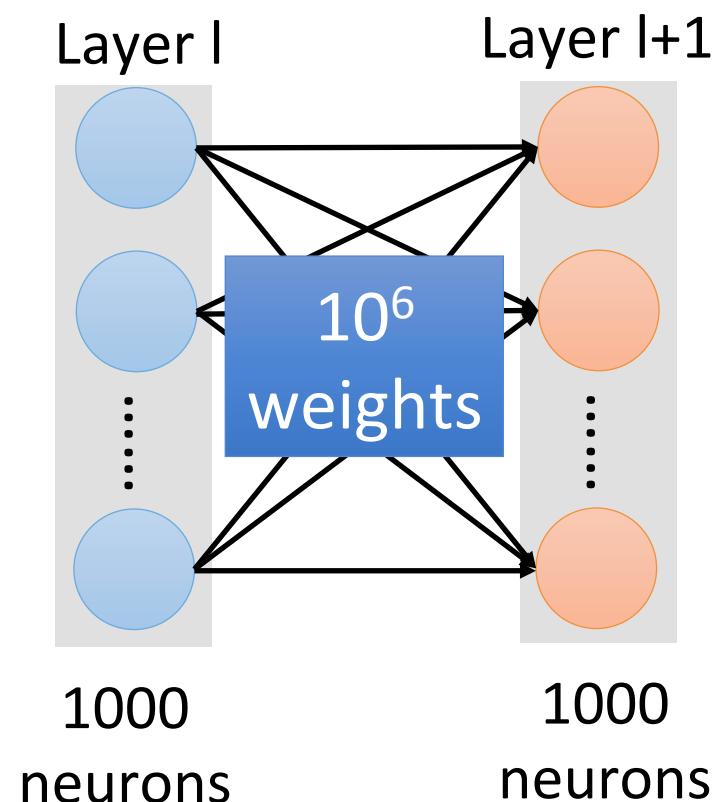
Find the network parameters θ^* that minimize total loss

Enumerate all possible ~~θ values~~

Network parameters

Millions of parameters

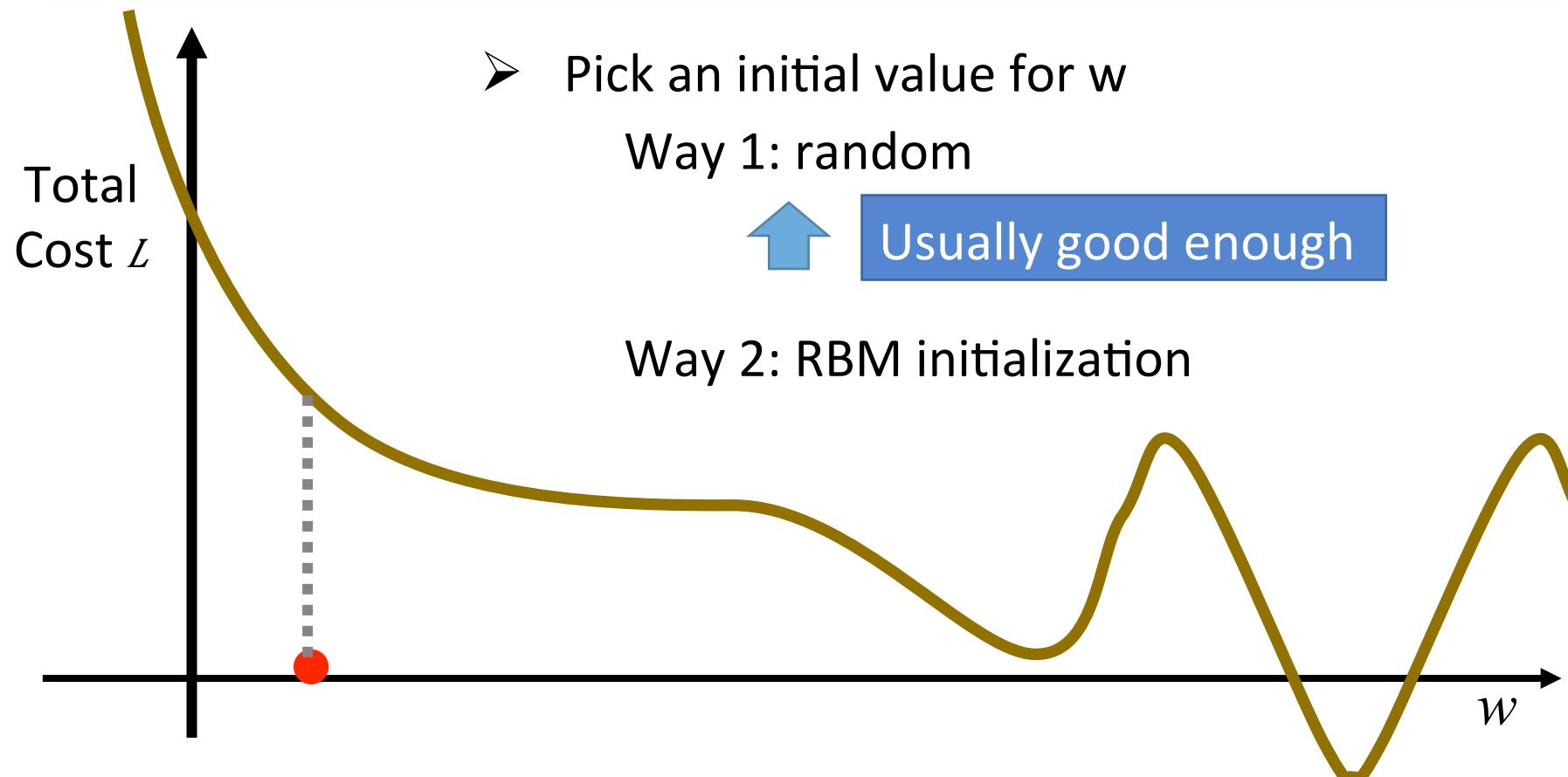
E.g. speech recognition: 8 layers
and 1000 neurons each layer



Gradient Descent

Network parameters

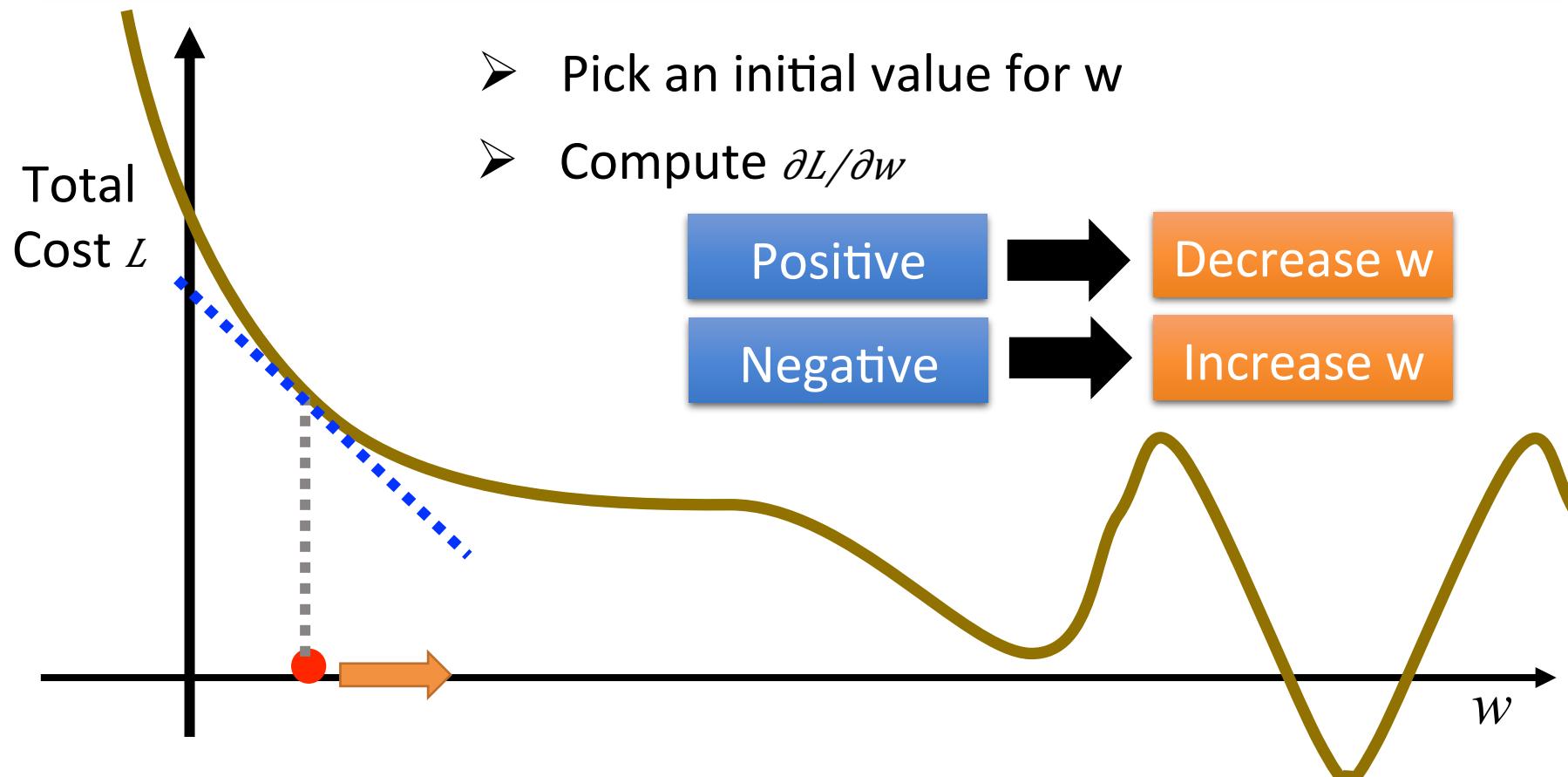
Find the network parameters θ^* that minimize total loss



Gradient Descent

Network parameters

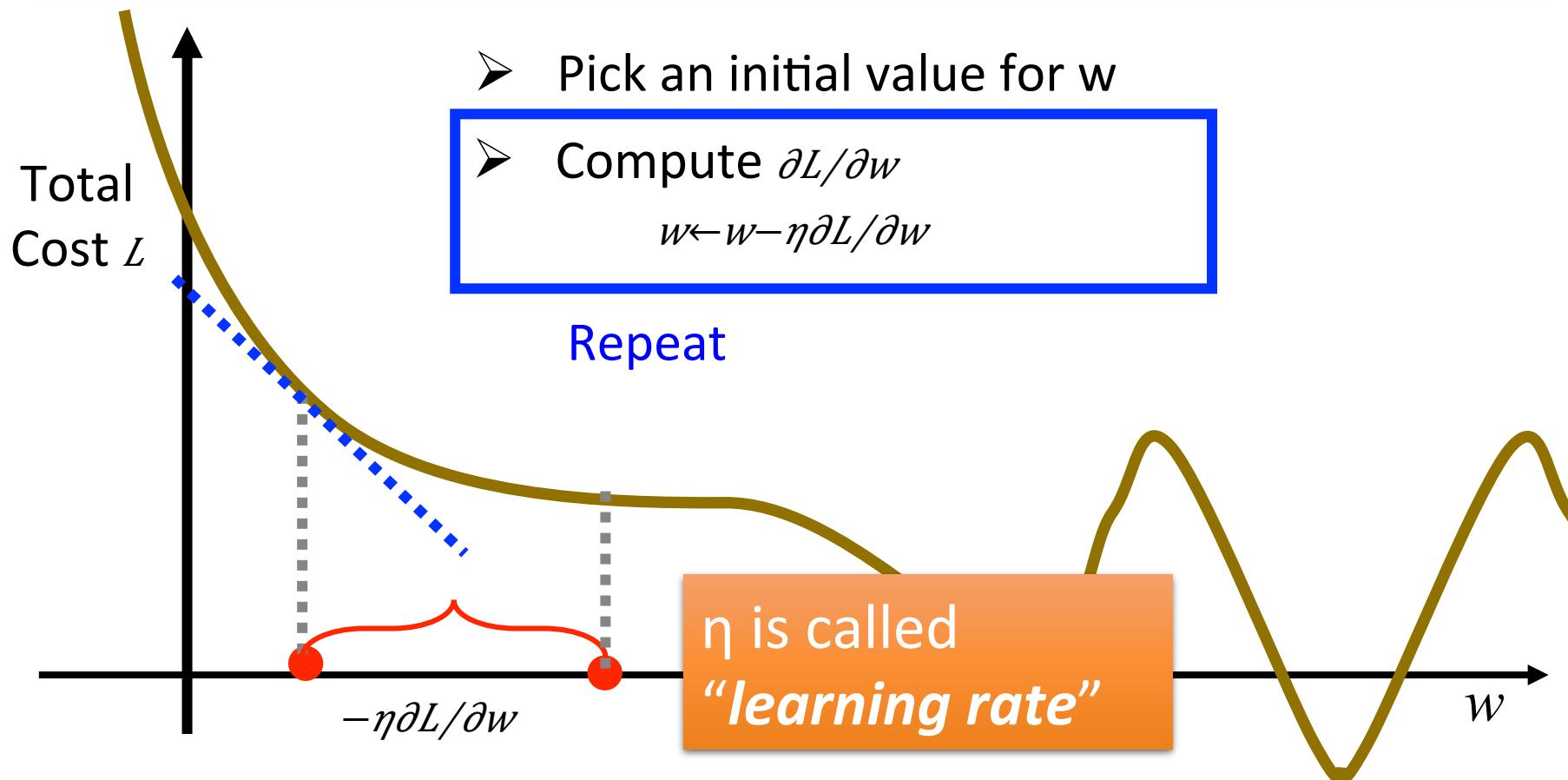
Find the network parameters θ^* that minimize total loss



Gradient Descent

Network parameters

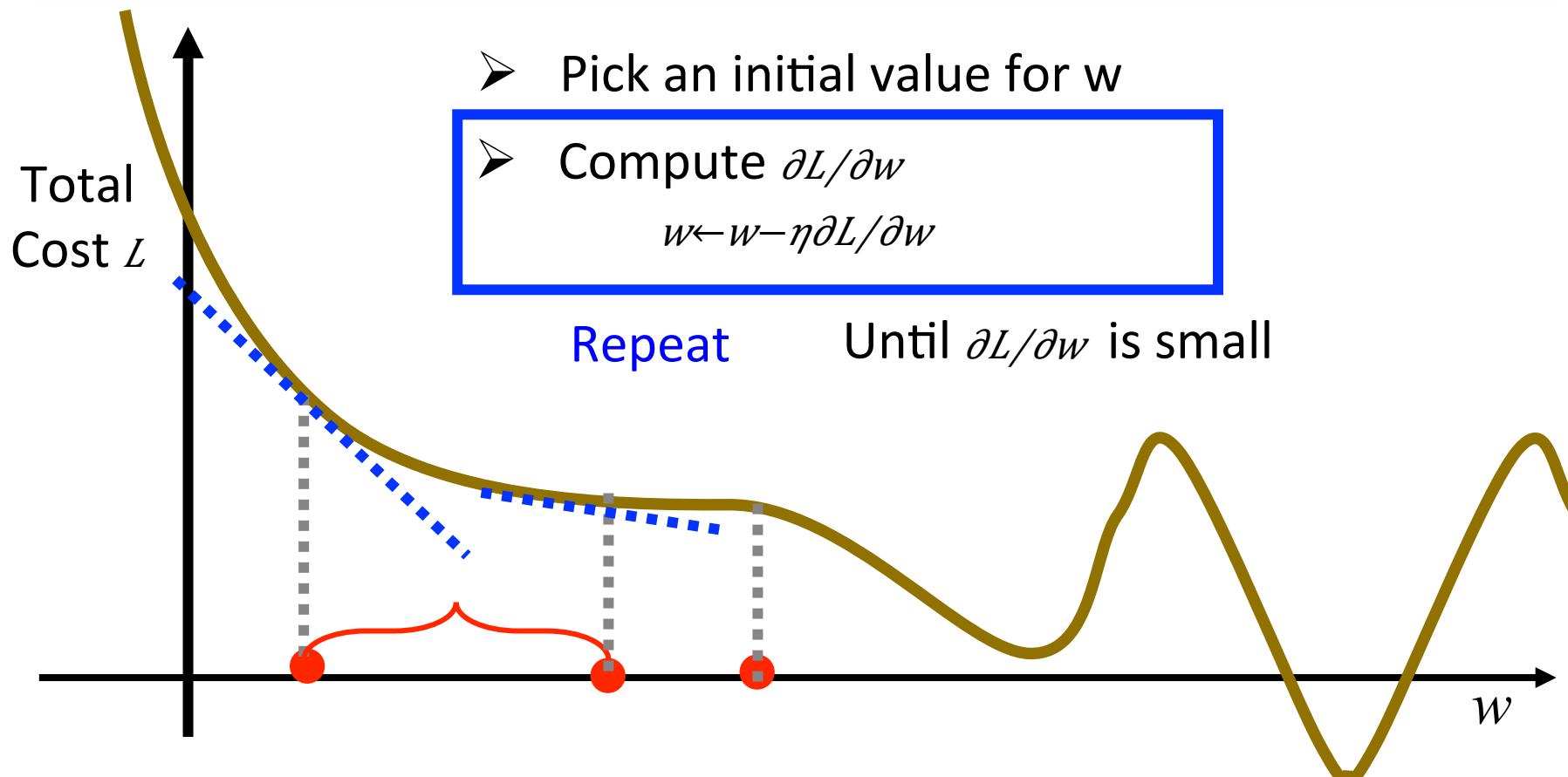
Find the network parameters θ^* that minimize total loss



Gradient Descent

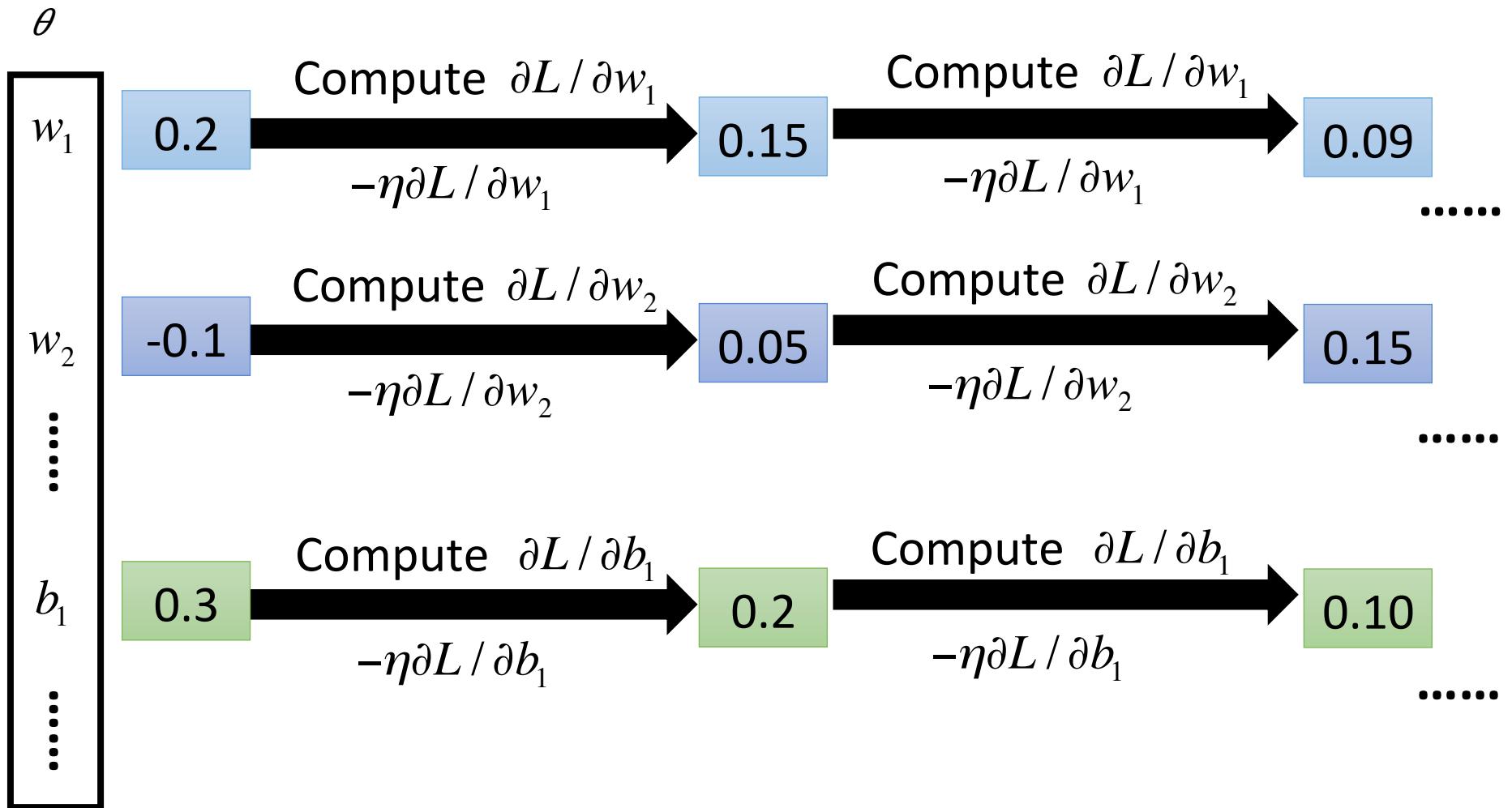
Network parameters

Find the network parameters θ^* that minimize total loss



Gradient Descent

➤ Compute





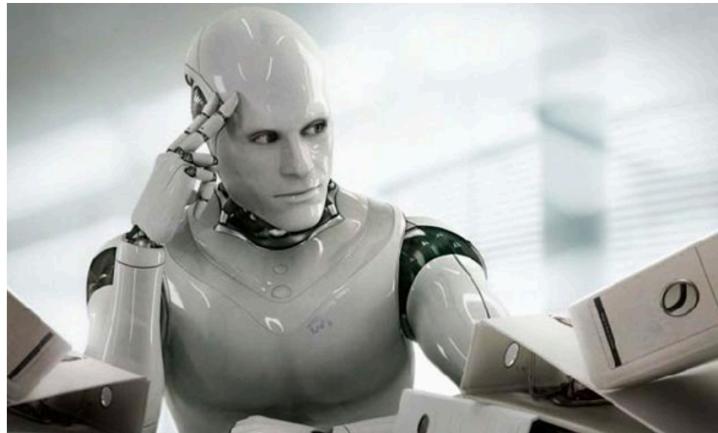
Gradient Descent

This is the “learning” of machines

Speech recognition, image recognition and alpha go, etc. use this learning approach.

→ Even alpha go using this approach.

大家以為 Learning 是



其實 Learning 只是



I hope you are not too disappointed :p

Backpropagation

- Backpropagation: an efficient way to compute $\partial L / \partial w$
 - Ref: http://speech.ee.ntu.edu.tw/~tlkagk/courses/MLDS_2015_2/Lecture/Theano%20DNN.ecm.mp4/index.html



theano

libdnn
台大周伯威
同學開發

Caffe

Microsoft
CNTK

K
keras

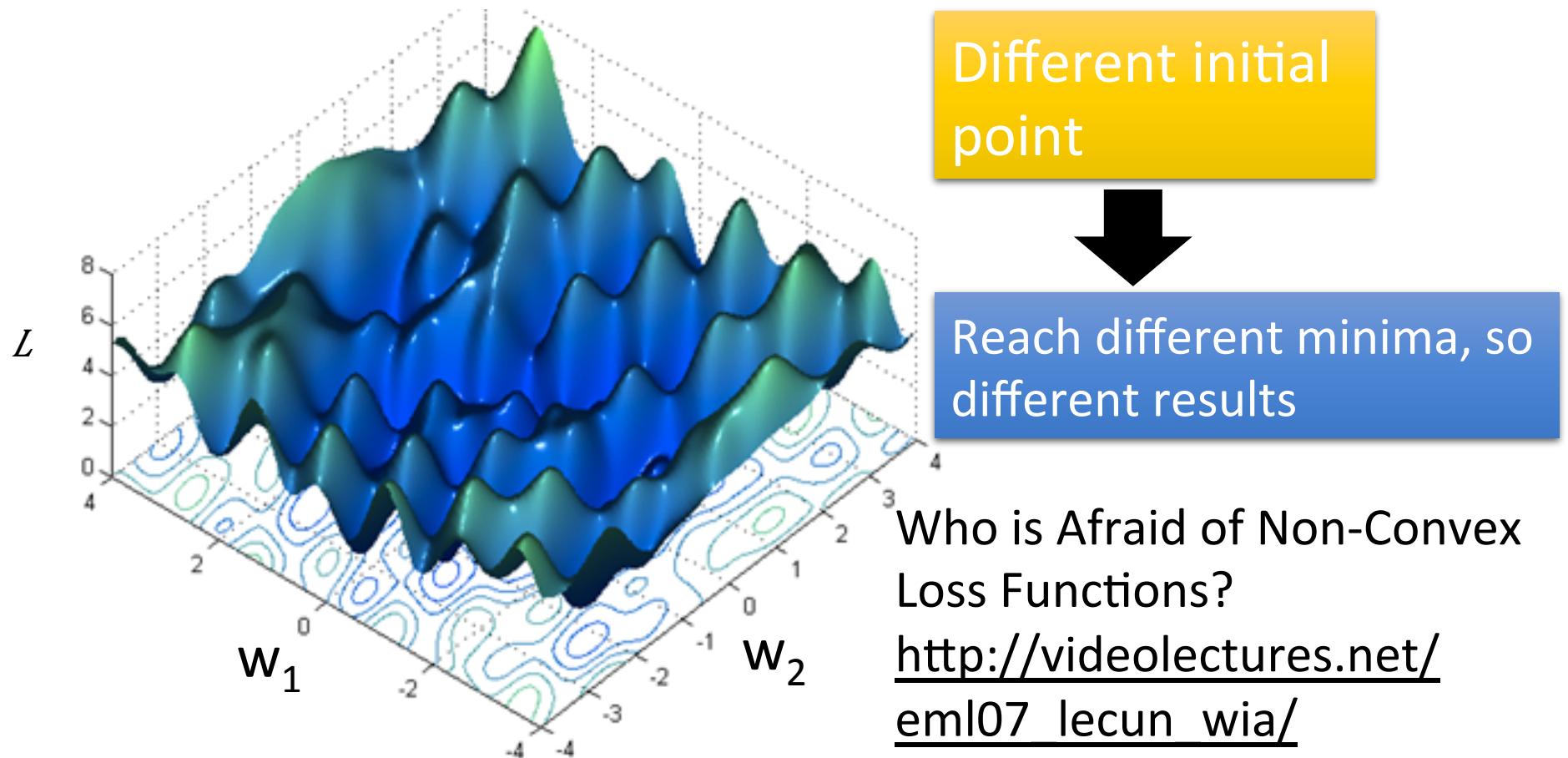
m xnet

Don't worry about $\partial L / \partial w$, the toolkits will handle it.

Local Minima

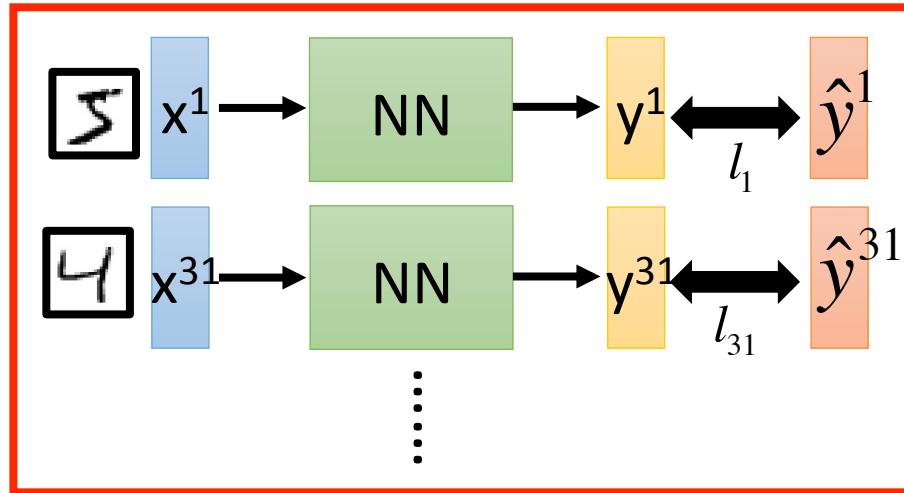
$$L(w^1, w^2, \dots, w^L) = \sigma(w^L \cdot \dots \cdot \sigma(w^2 \cdot \sigma(w^1 \cdot x + b^1) + b^2) \dots + b^L)$$

- Gradient descent never guarantee global minima

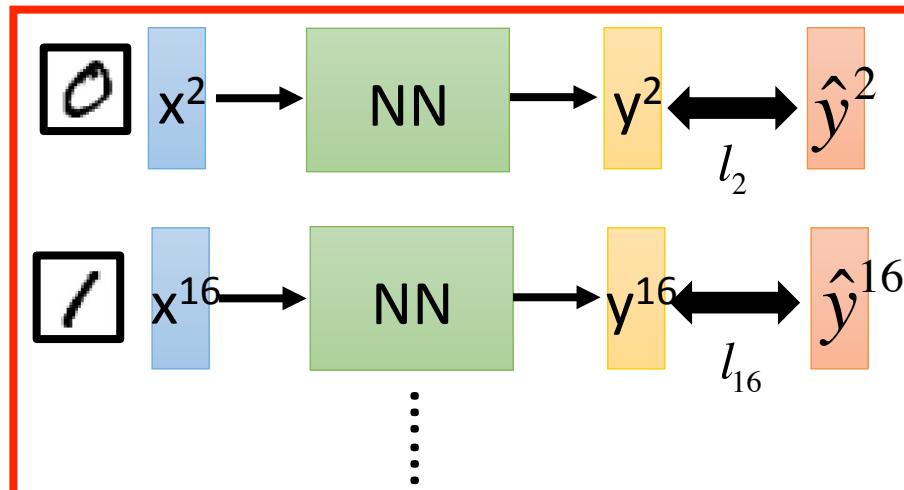


Mini-batch

Mini-batch



Mini-batch



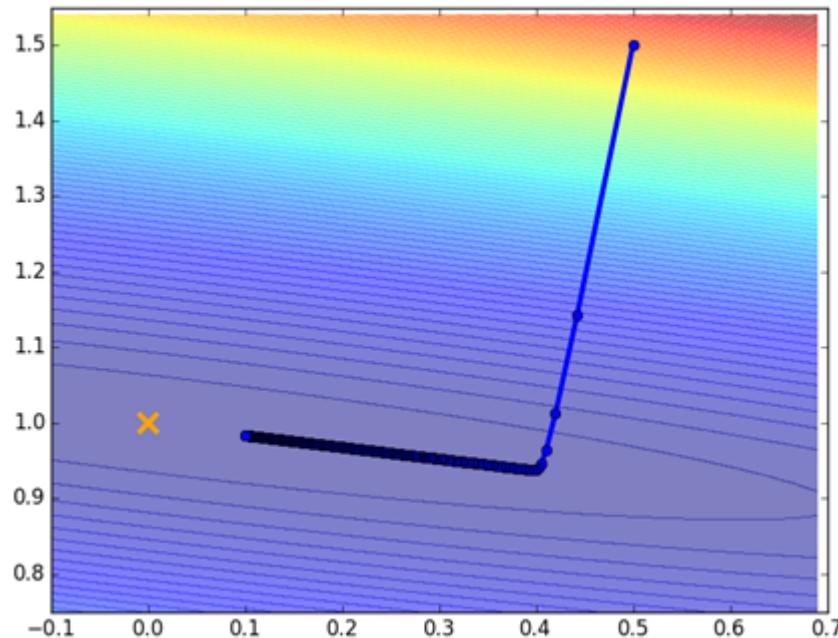
- Randomly initialize network parameters
- (Randomly) Pick the 1st batch $L = l_1 + l_{31} + \dots$
Update parameters once
- Pick the 2nd batch
 $L = l_2 + l_{16} + \dots$
Update parameters once
- ⋮

L is different each time when we update parameters!

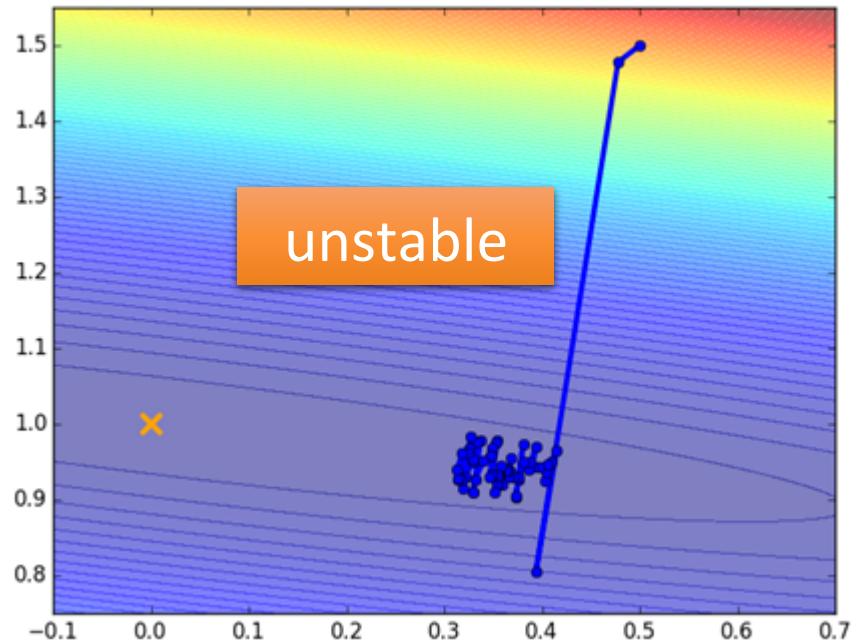
目標換來換去?!

Mini-batch

Batch Gradient Descent



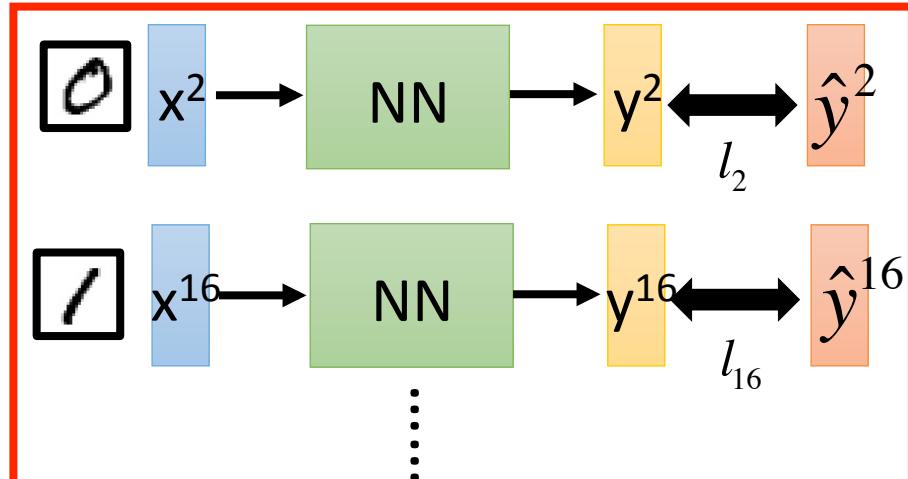
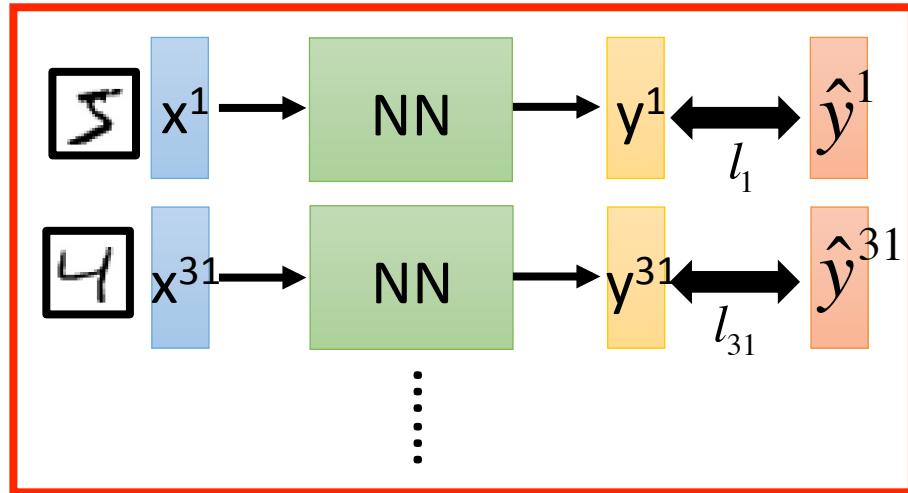
With Mini-batch



The colors represent the total loss on all training data.

Mini-batch

Mini-batch



- Randomly initialize network parameters

- Pick the 1st batch
 $L = l_1 + l_{31} + \dots$
Update parameters once
- Pick the 2nd batch
 $L = l_2 + l_{16} + \dots$
Update parameters once
⋮
- Until all mini-batches have been picked

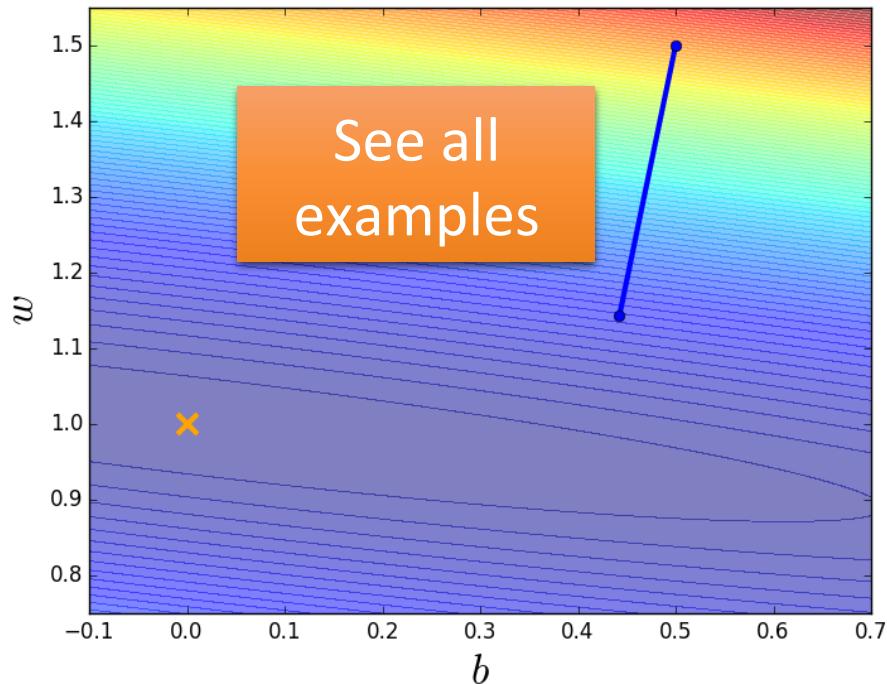
one epoch

Repeat the above process

Mini-batch is Faster

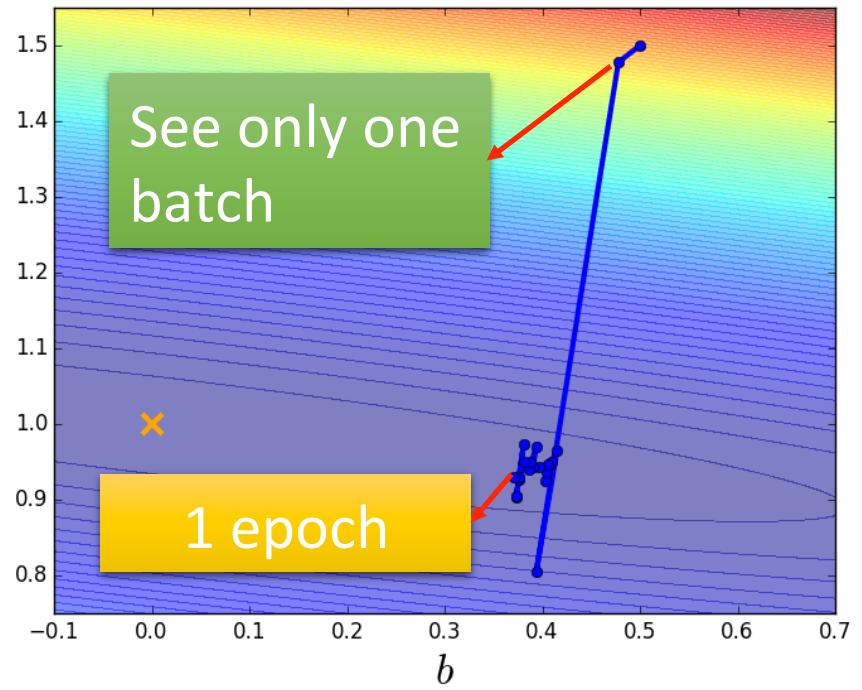
Batch Gradient Descent

Update after seeing all examples



With Mini-batch 勝!

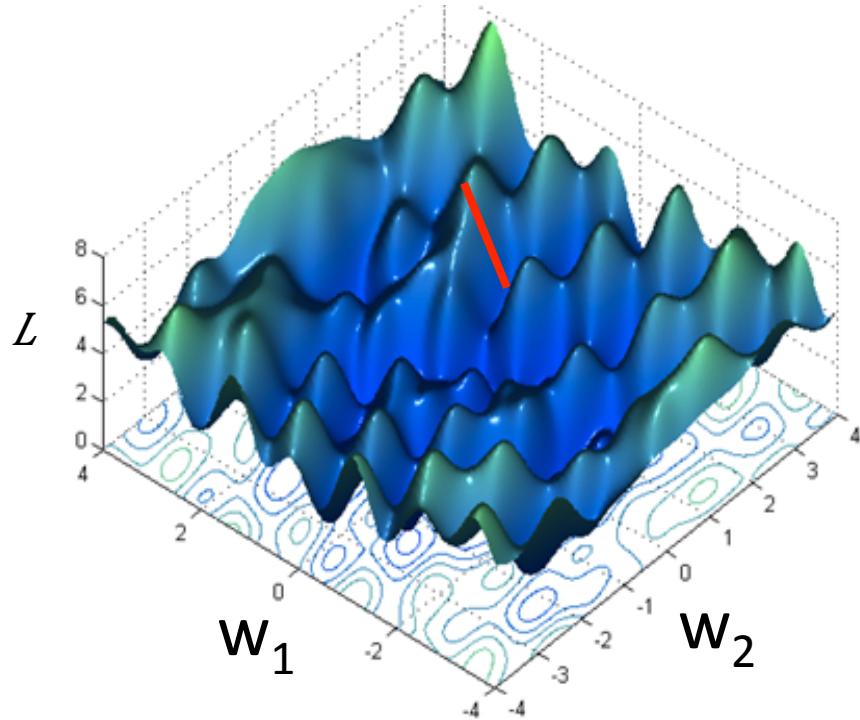
If there are 20 batches, update 20 times in one epoch.



Mini-batch is typically better

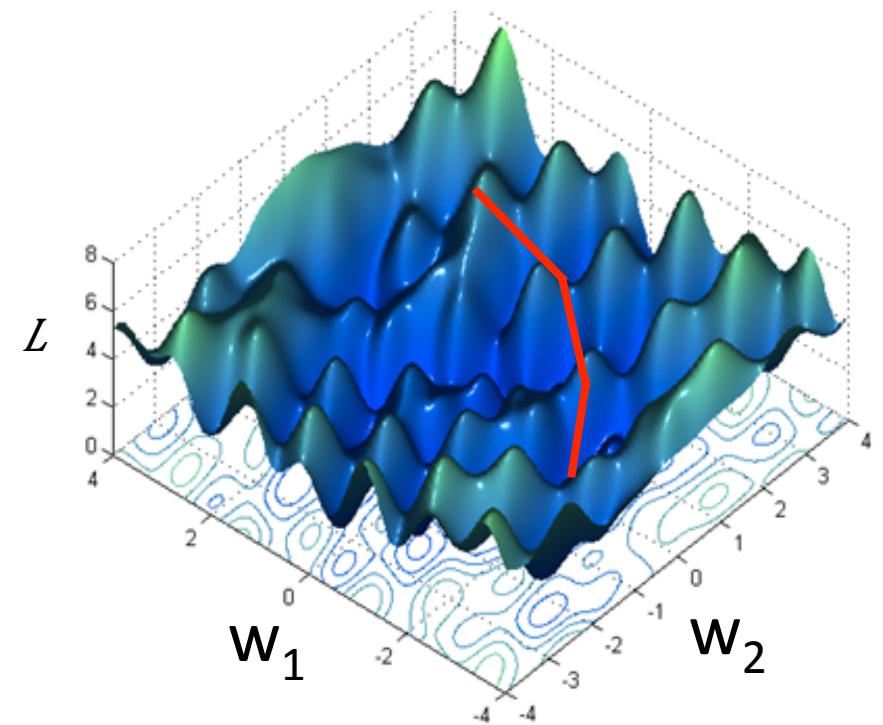
Batch Gradient Descent

Given the same initial parameters
and hyper-parameters->
Same local minimal



With Mini-batch 勝!

Explore more in space of non-convex loss function



Three Steps for Deep Learning



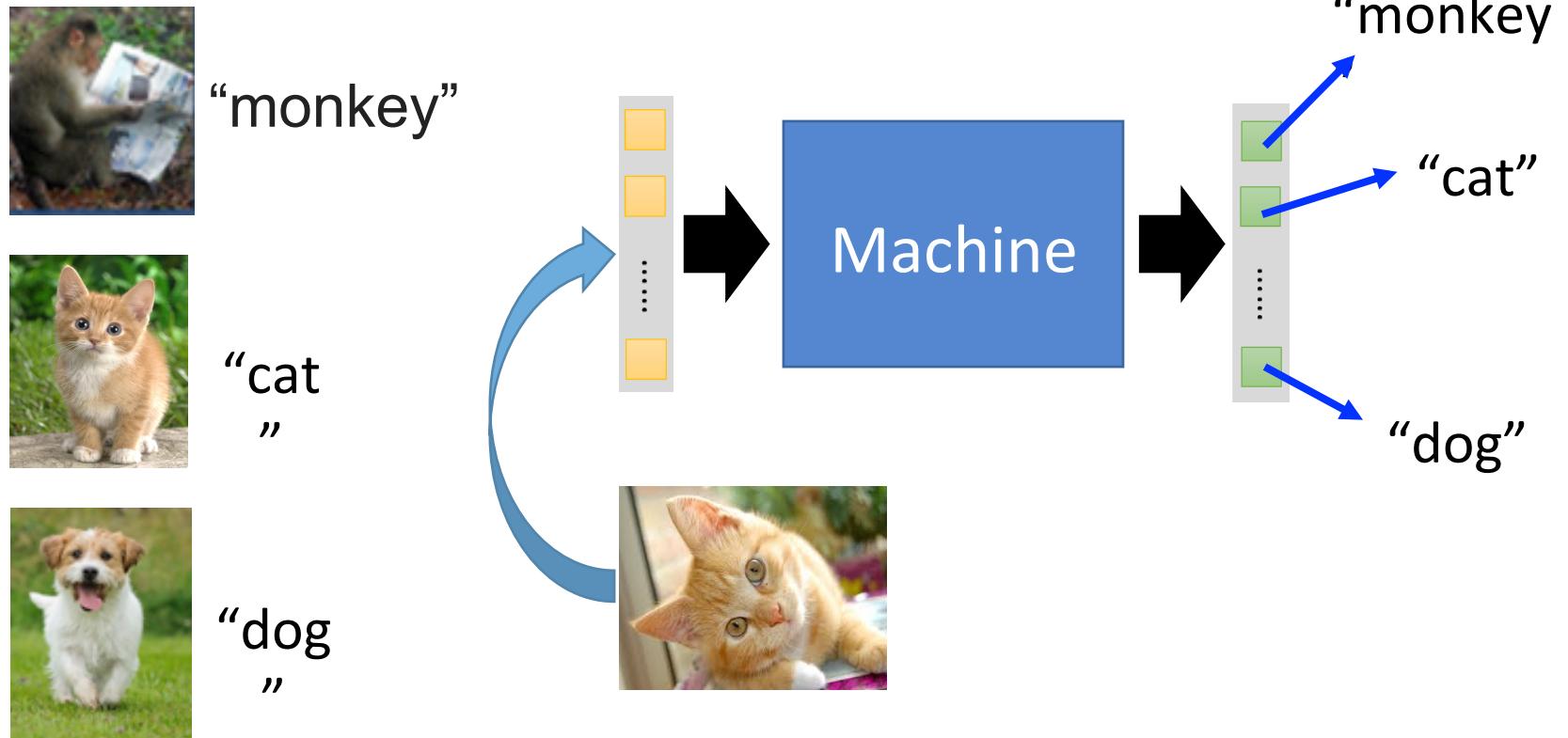
That's it!

If the input/output of the machine in your task are both vectors, you know how to deal with it.



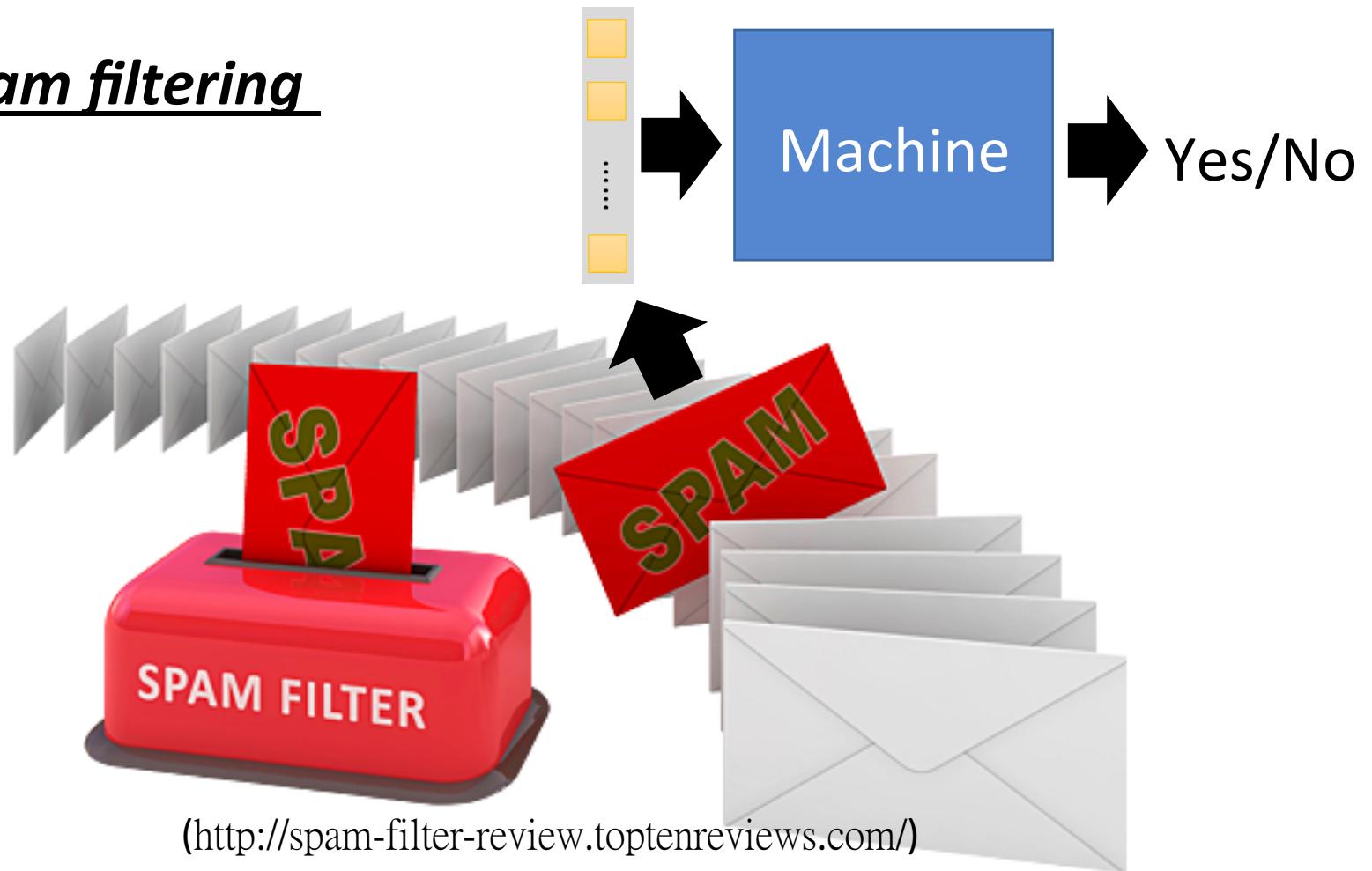
For example, you can do

- Image Recognition



For example, you can do

Spam filtering



Outline of Lecture 1

Three Steps for Deep Learning

- The three steps can also apply on other machine learning methods

Why Deep Learning?

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.

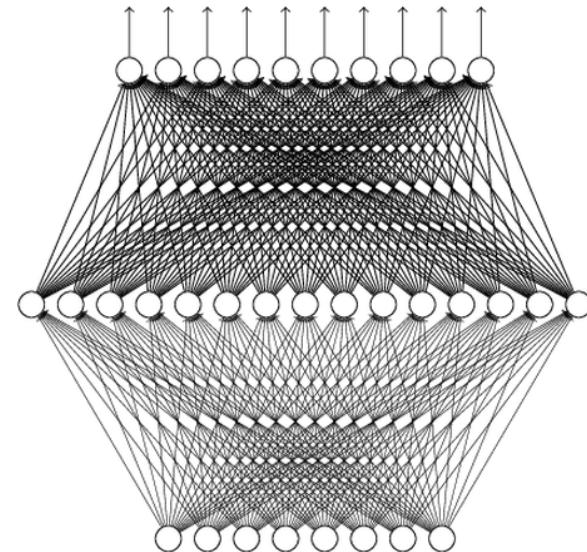
Universality Theorem

Any continuous function f

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

Can be realized by a network
with one hidden layer

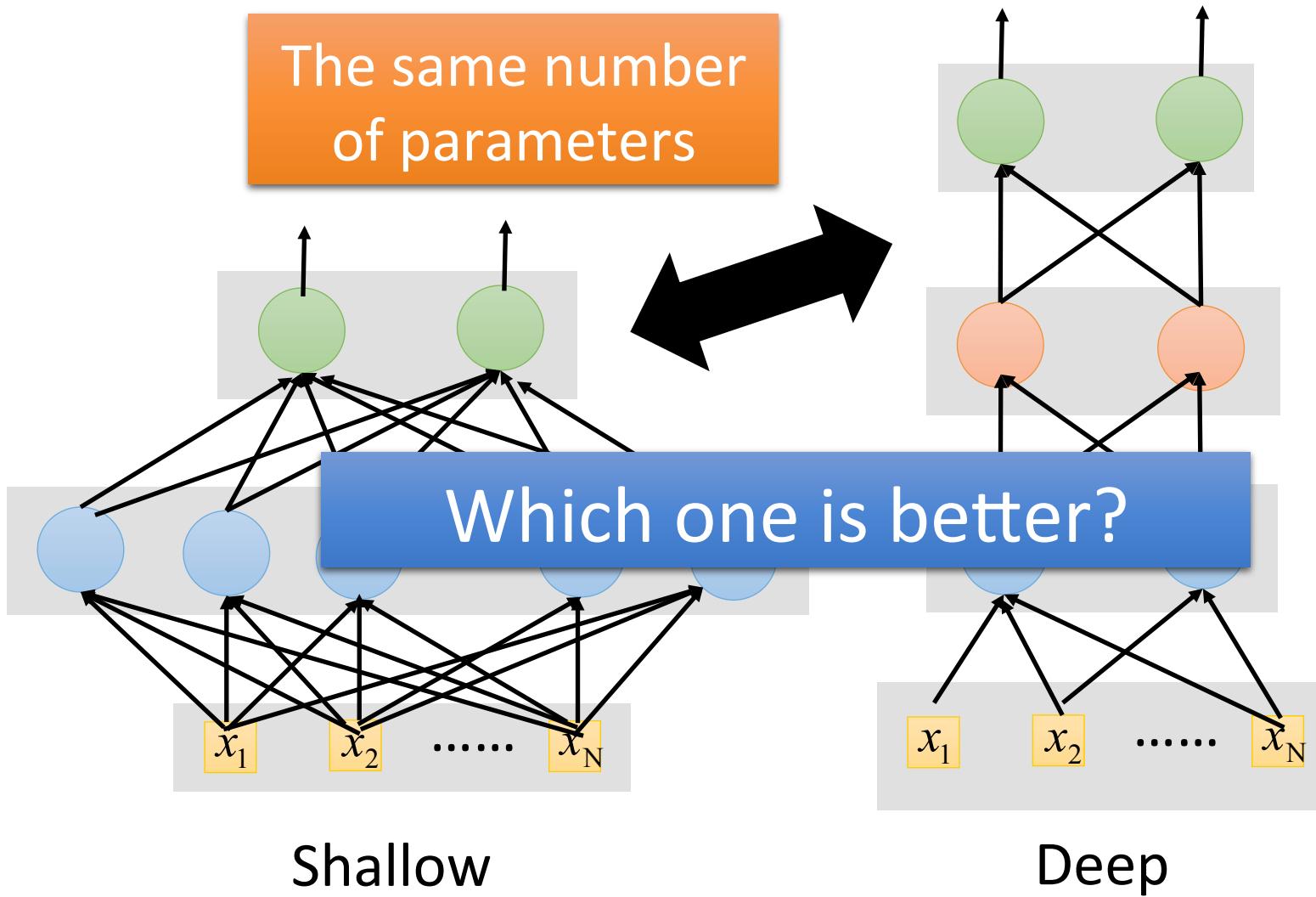
(given **enough** hidden
neurons)



Reference for the reason:
[http://
neuralnetworksanddeeplearning.
com/chap4.html](http://neuralnetworksanddeeplearning.com/chap4.html)

Why “Deep” neural network not “Fat” neural
network?

Fat + Short v.s. Thin + Tall



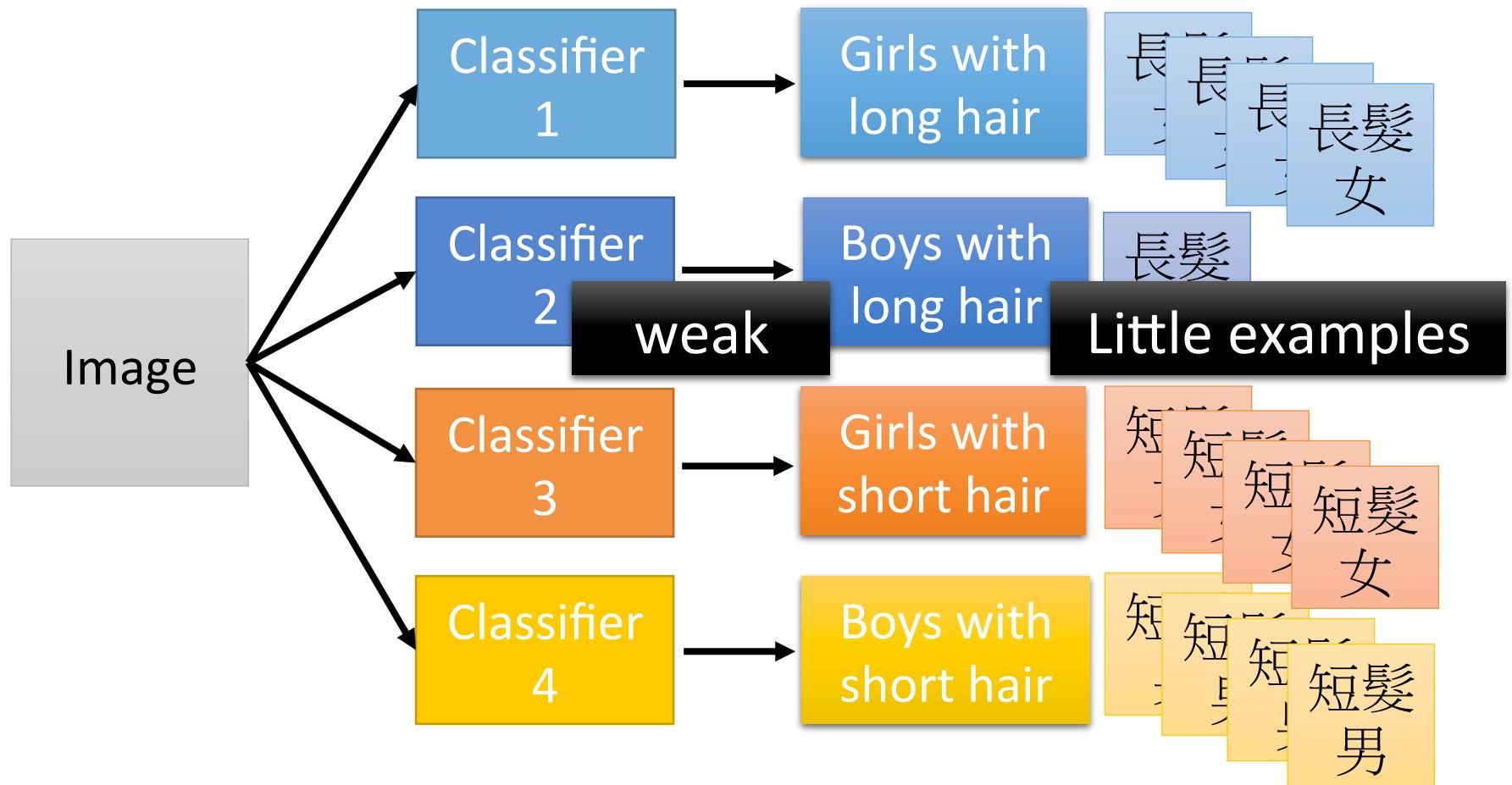
Fat + Short v.s. Thin + Tall

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.

Why Deep?

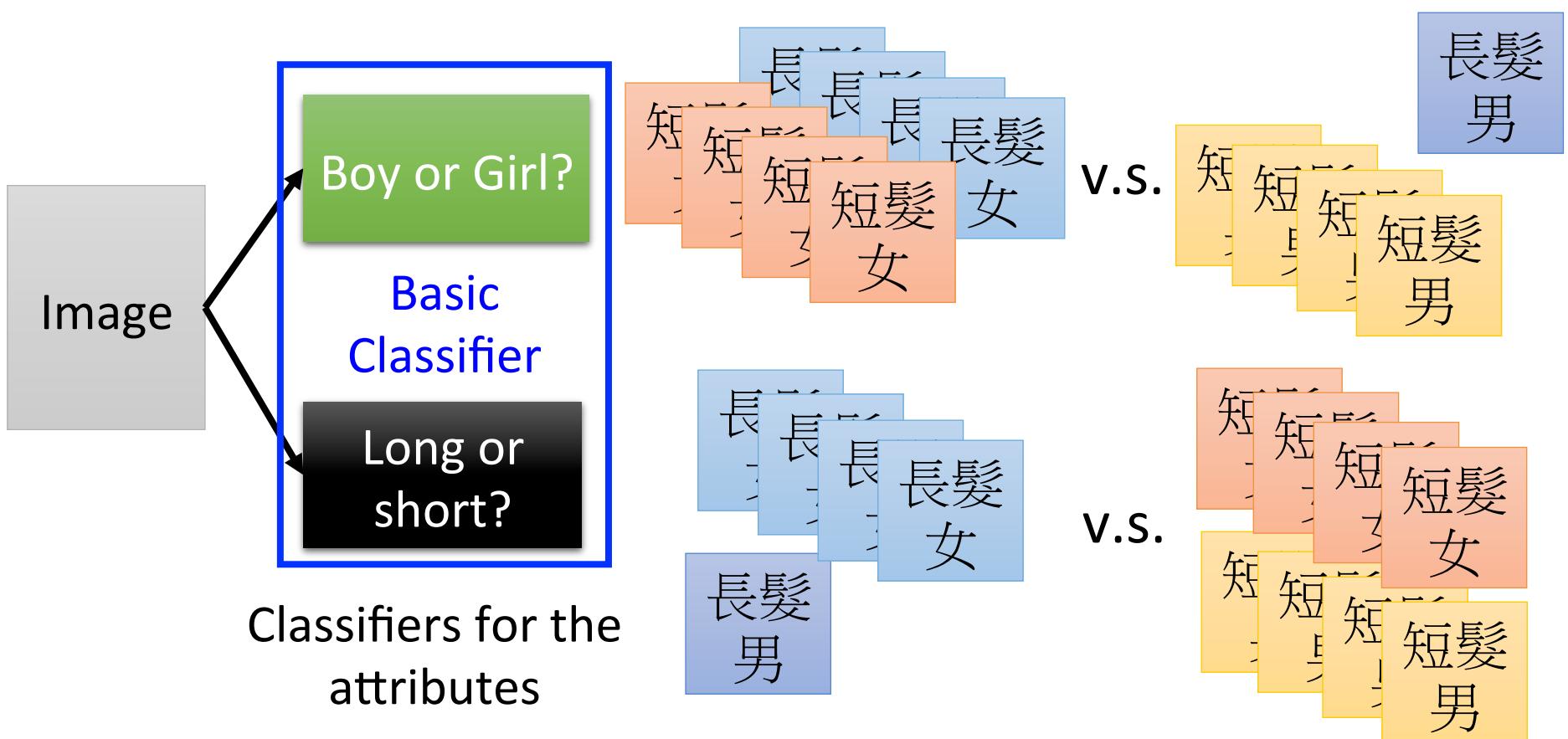
- Deep → Modularization



Why Deep?

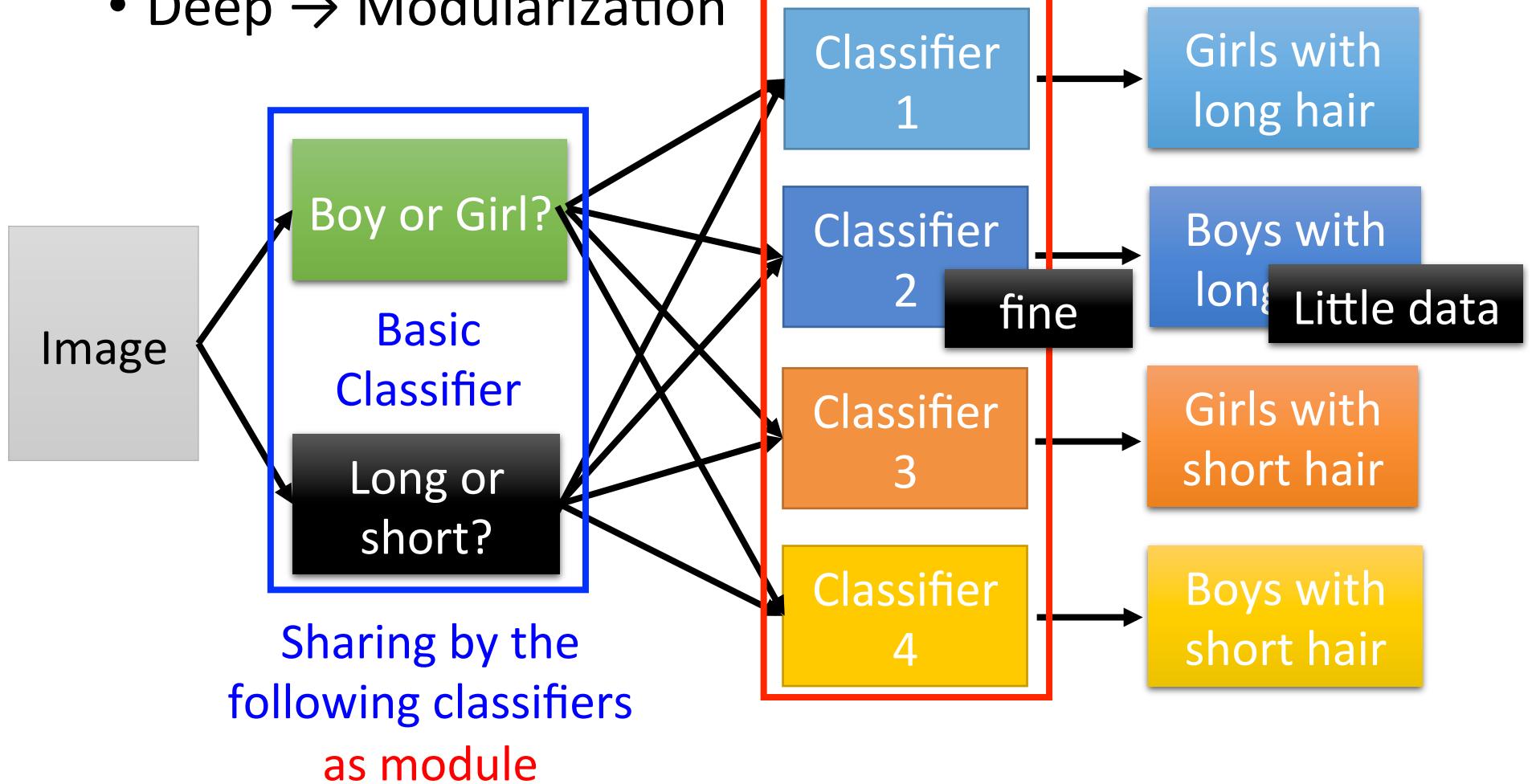
Each basic classifier can have sufficient training examples.

- Deep → Modularization



Why Deep?

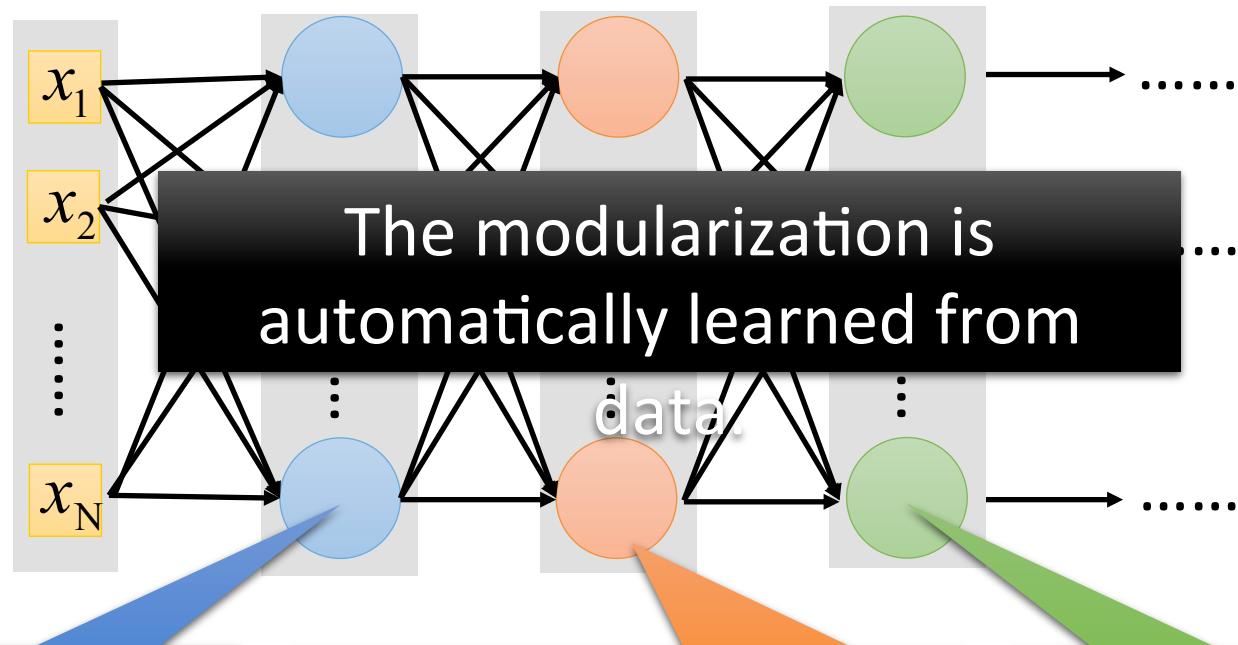
- Deep → Modularization



Why Deep?

Deep Learning also works
on small data set.

- Deep → Modularization → Less training data?



The most basic
classifiers

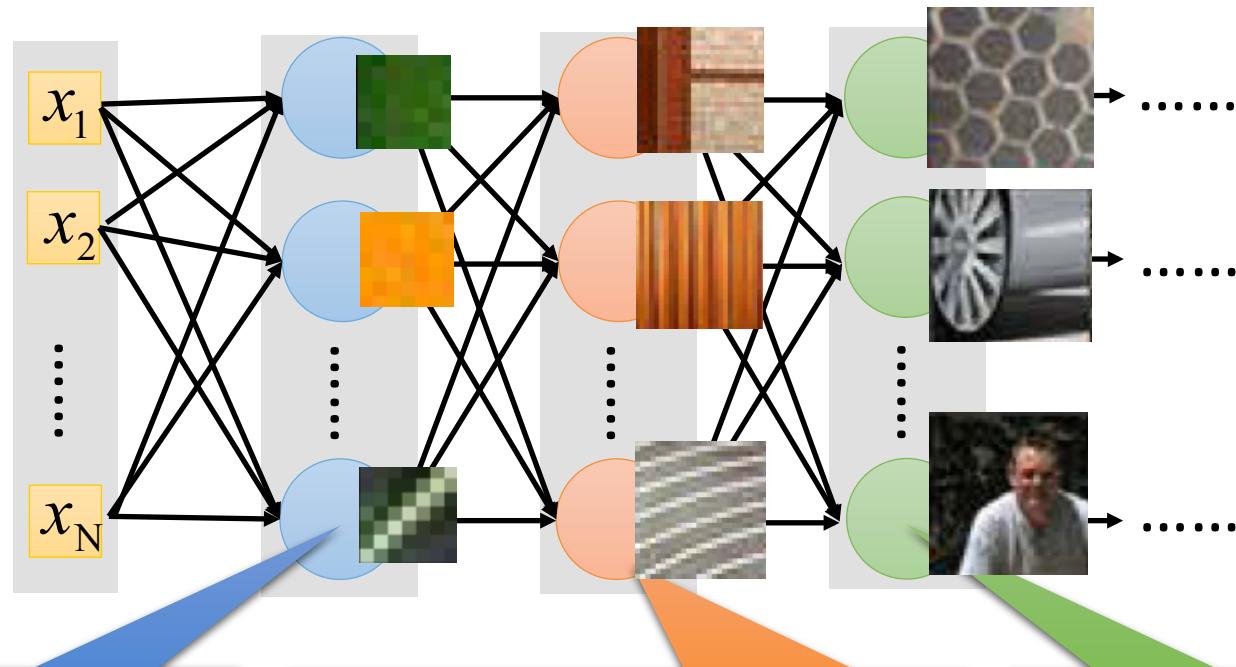
Use 1st layer as module
to build classifiers

Use 2nd layer as
module

Why Deep?

Reference: Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014* (pp. 818–833)

- Deep → Modularization

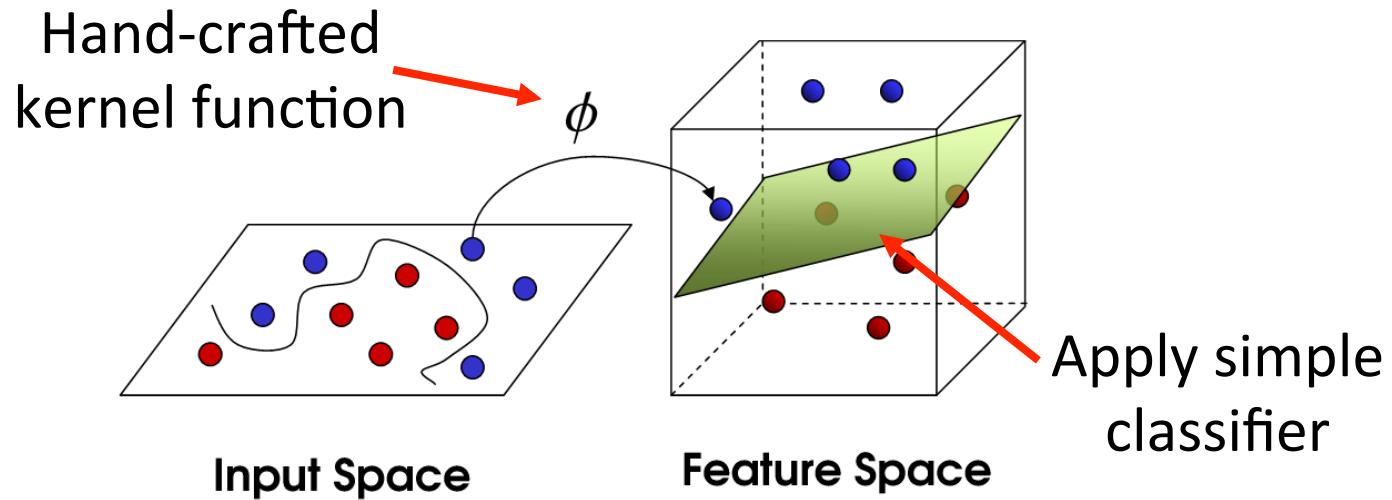


The most basic
classifiers

Use 1st layer as module
to build classifiers

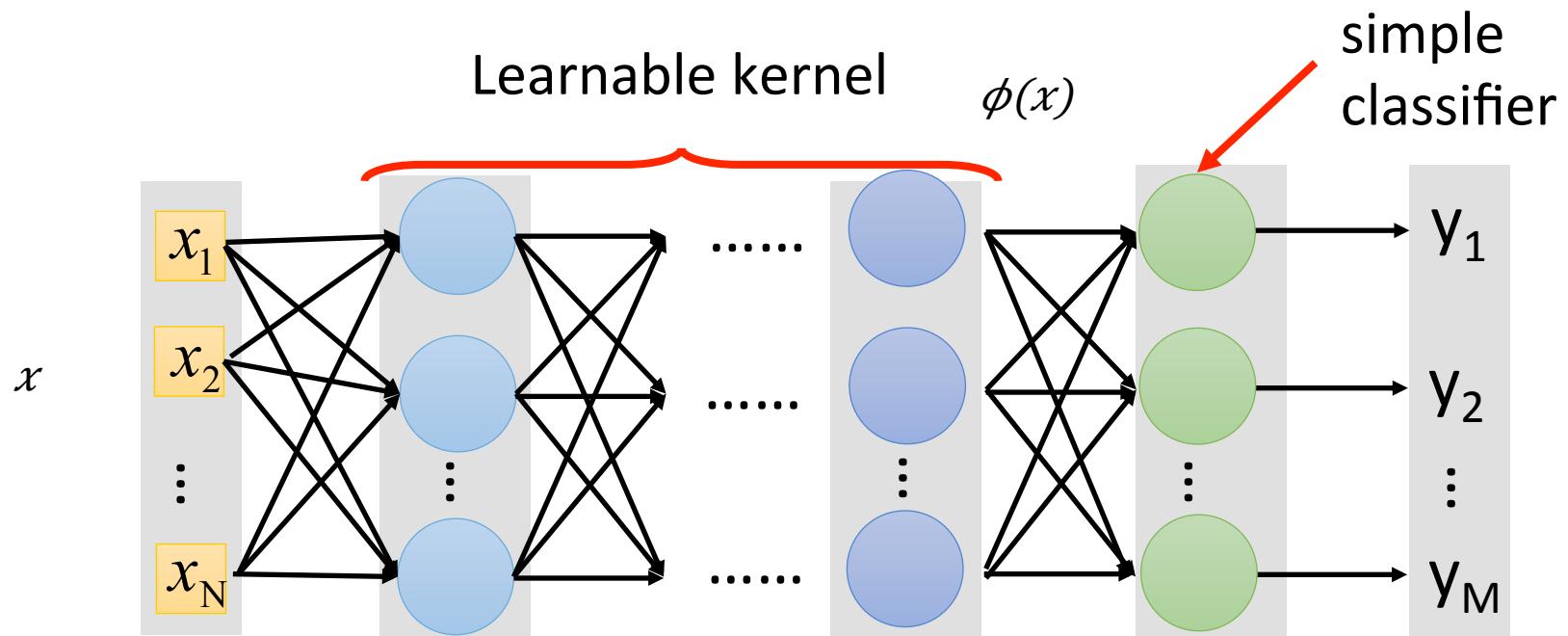
Use 2nd layer as
module

SVM

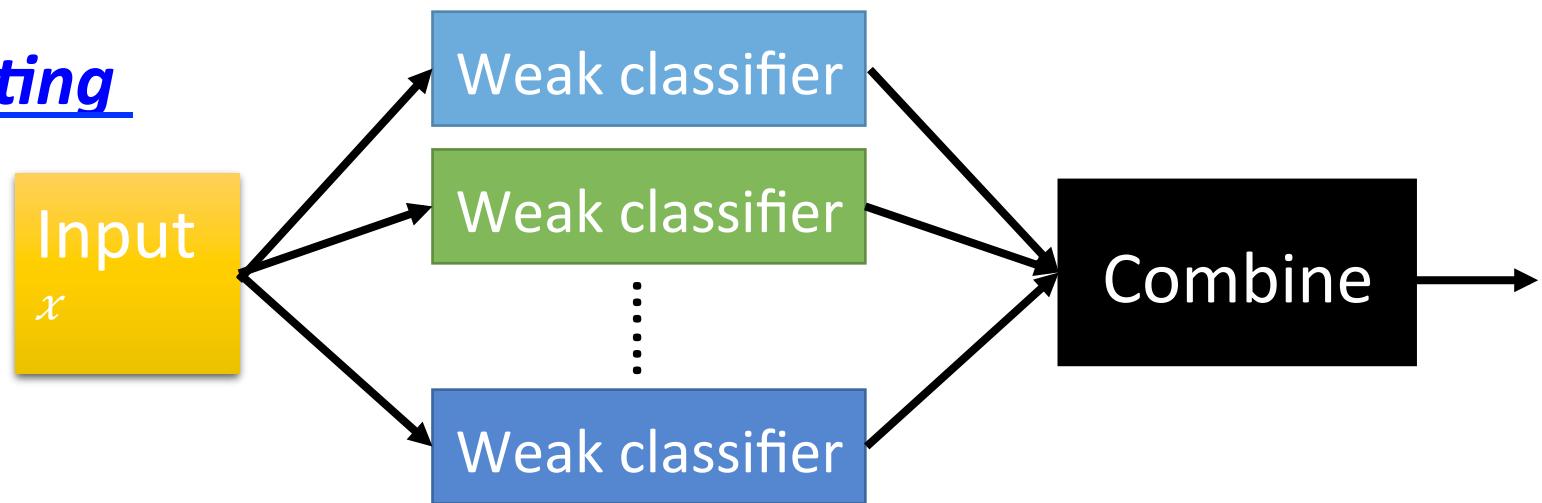


Source of image: http://www.gipsa-lab.grenoble-inp.fr/transfert/seminaire/455_Kadri2013Gipsa-lab.pdf

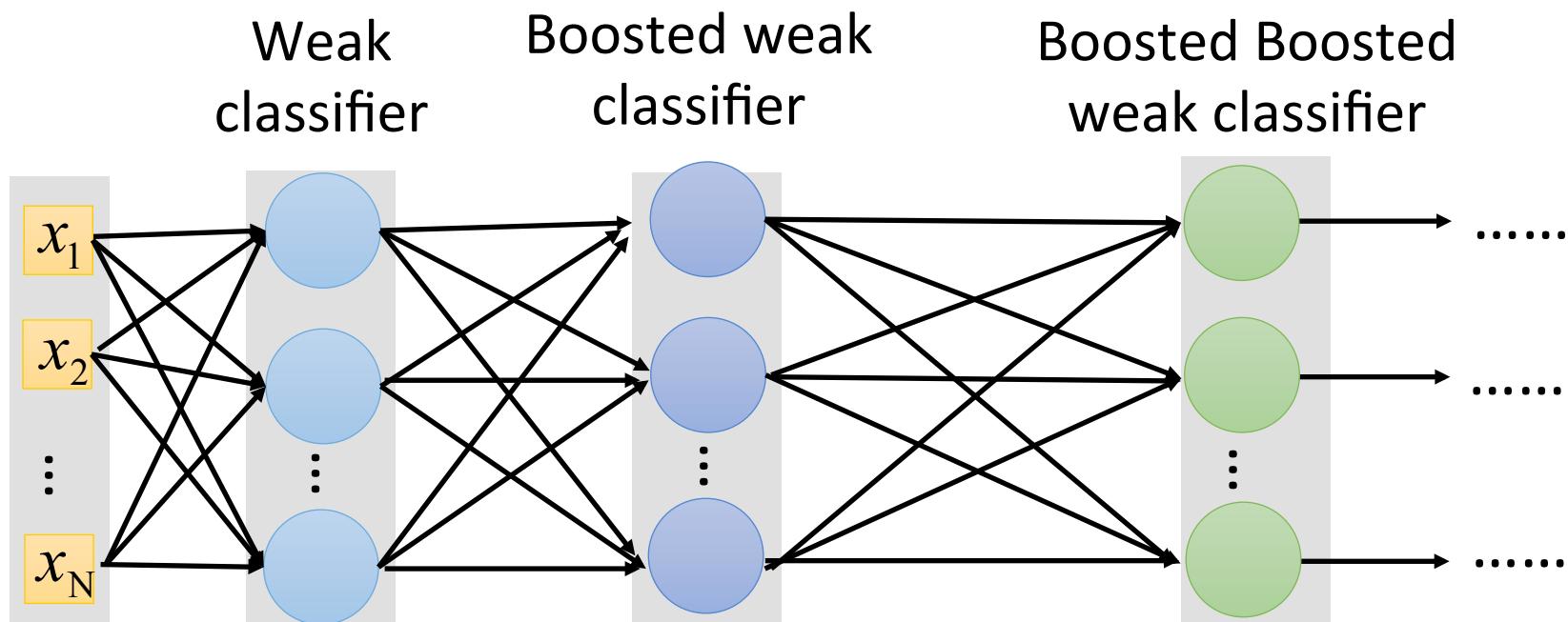
Deep Learning



Boosting



Deep Learning



Concluding Remarks of Lecture I

- Deep Learning is simple!
- Deep Learning is powerful!

Deep Learning 就像 雷神之槌



[http://ent.ltn.com.tw/news/
breakingnews/1144545](http://ent.ltn.com.tw/news/breakingnews/1144545)

無法輕易被舉起來