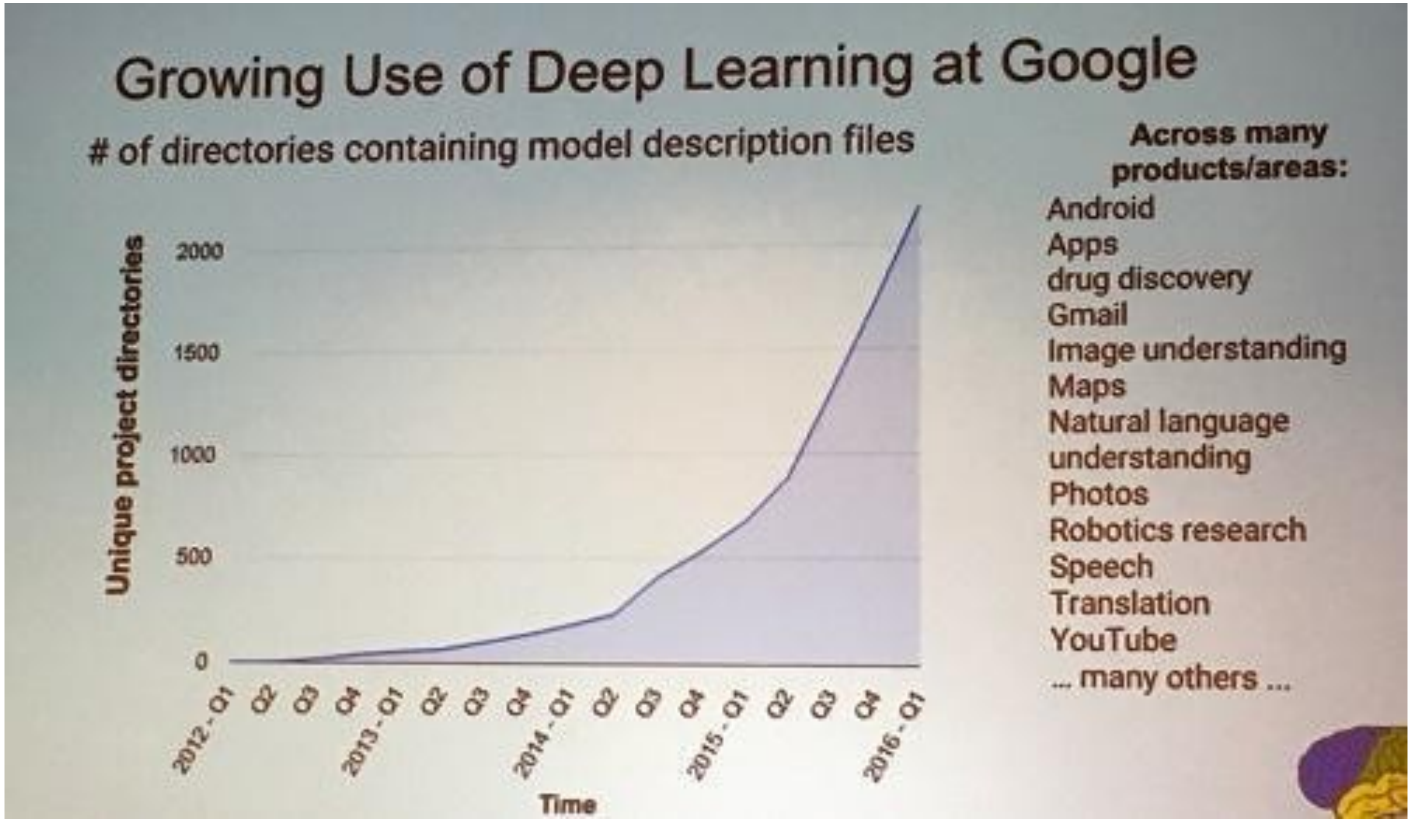
Introduction of Deep Learning

Slides are provided by Prof. Hung-yi Lee

Deep learning attracts lots of attention.

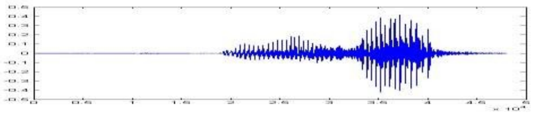
- I believe you have seen lots of exciting results before.



Deep learning trends at Google. Source: SIGMOD/Jeff Dean

Machine Learning
≈ Looking for a Function

- Speech Recognition

$f($  $)$ = "How are you"

- Image Recognition

$f($  $)$ = "Cat"

- Playing Go

$f($  $)$ = "5-5" (next move)

- Dialogue System

$f($ "Hi" $)$ = "Hello"

(what the user said) (system response)

Image Recognition:

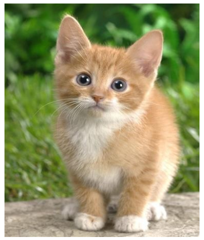
$f($  $)=$ "cat"

Framework

A set of function

Model

$f_1, f_2 \cdots$

$f_1($  $)=$ "cat"

$f_2($  $)=$ "monkey"

$f_1($  $)=$ "dog"

$f_2($  $)=$ "snake"

Framework

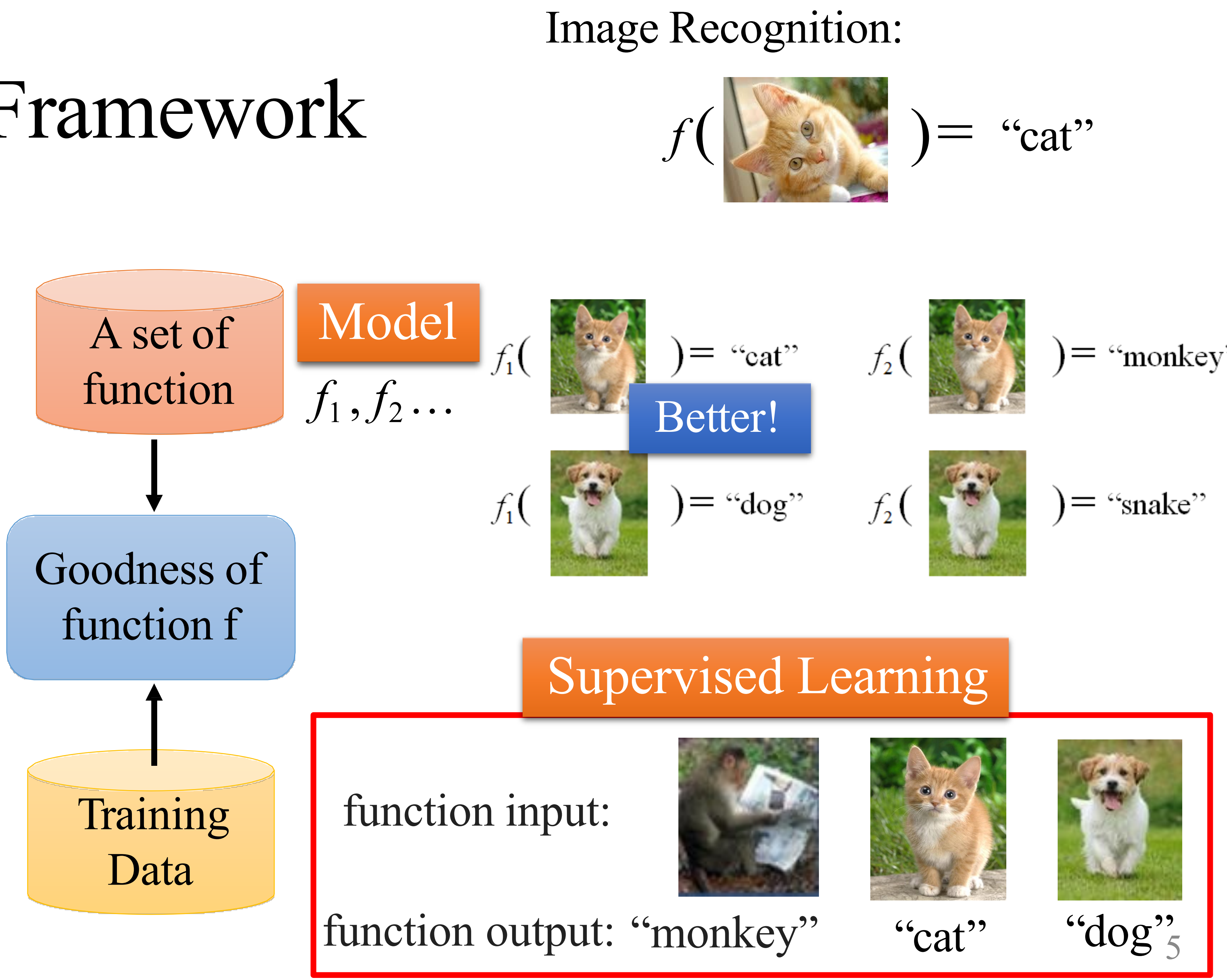
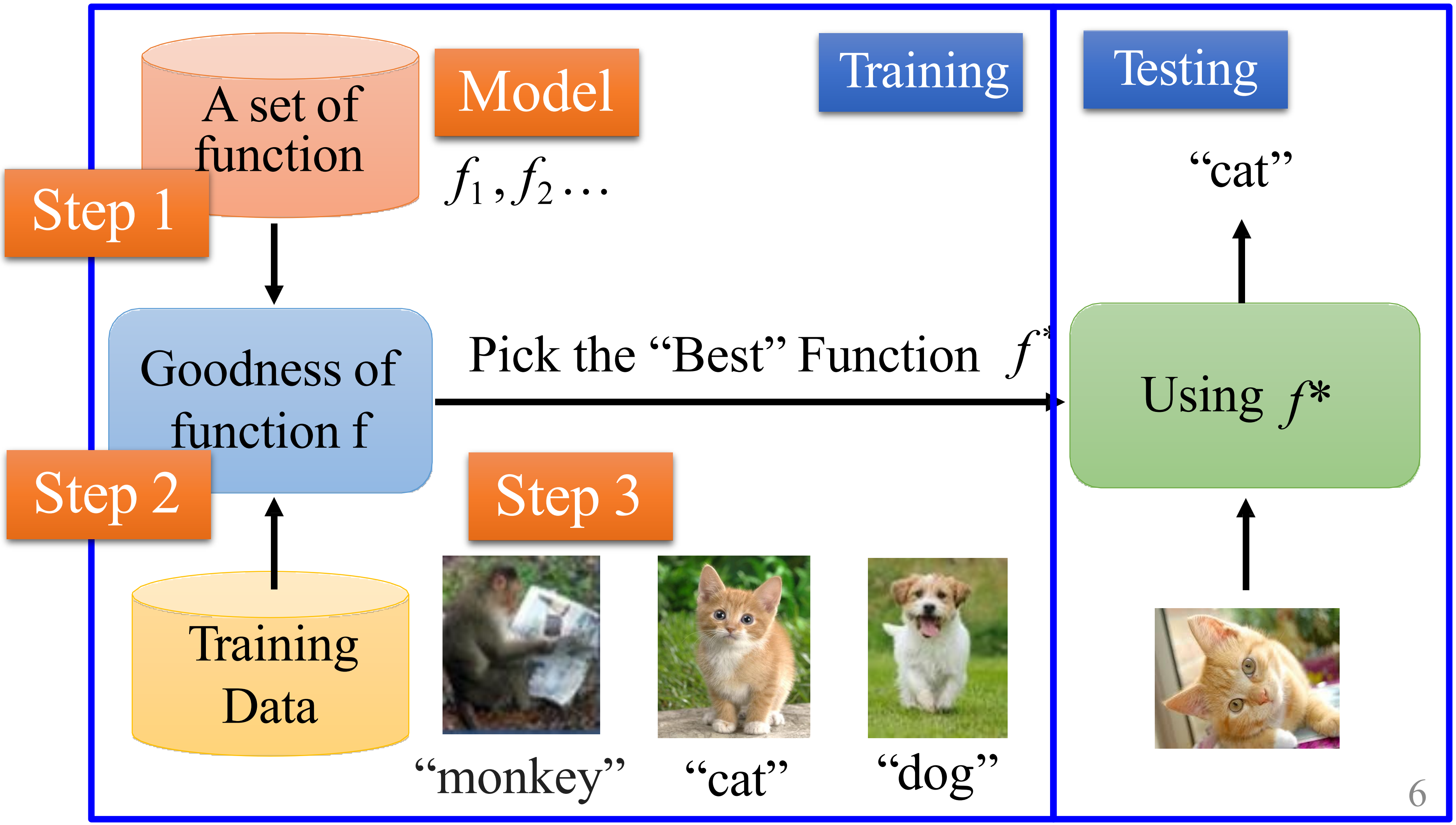


Image Recognition:
$f($ $)=$ "cat"
A set of function
Model
$f_1, f_2 \cdots$
$f_1($ $)=$ "cat"
$f_2($ $)=$ "monkey"
Better!
$f_1($ $)=$ "dog"
$f_2($ $)=$ "snake"
Goodness of function f
Supervised Learning
Training Data
function input:
function output: "monkey"
"cat"
"dog"

Framework

Image Recognition:

$f($  $)=$ “cat”



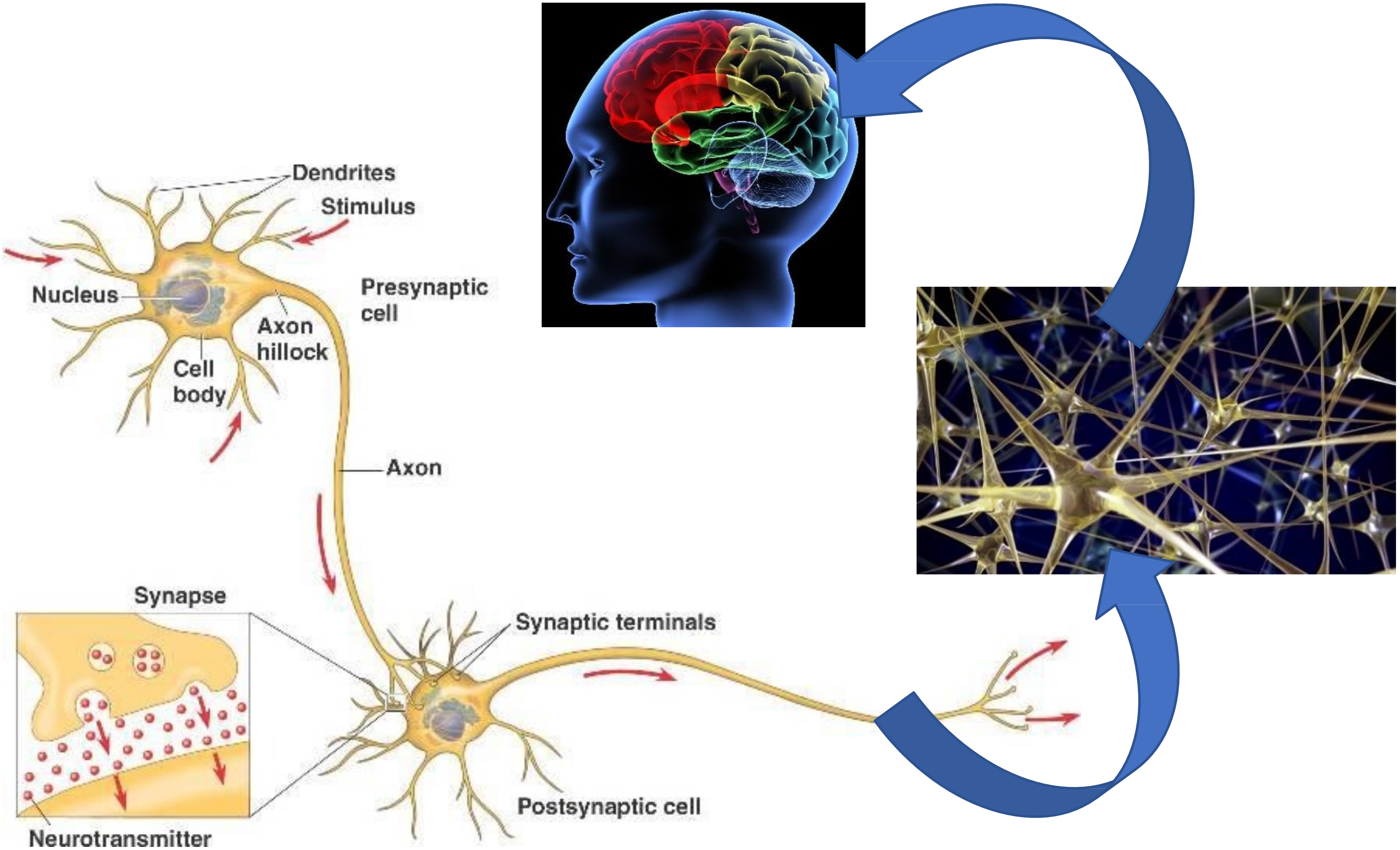
Three Steps for Deep Learning

Step 1: define a set of function → Step 2: goodness of function → Step 3: pick the best function

Three Steps for Deep Learning

Step 1: Neural Network → Step 2: goodness of function → Step 3: pick the best function

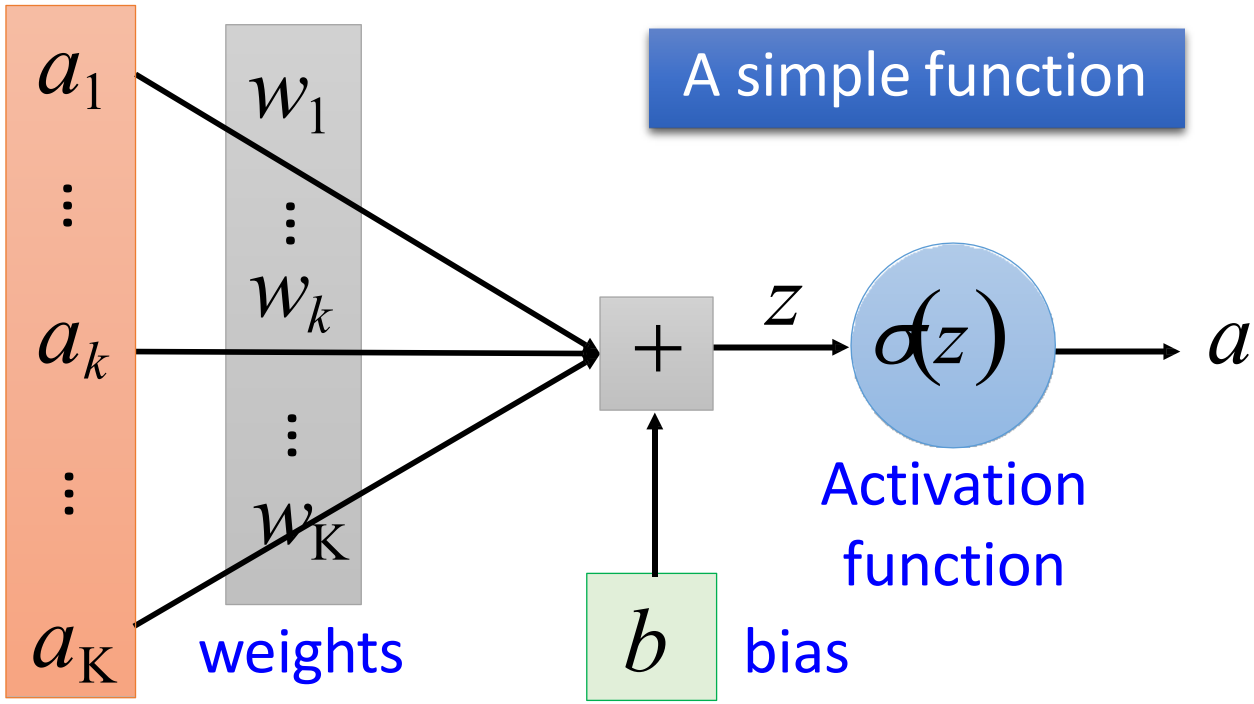
Human Brains



Neural Network

Neuron

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

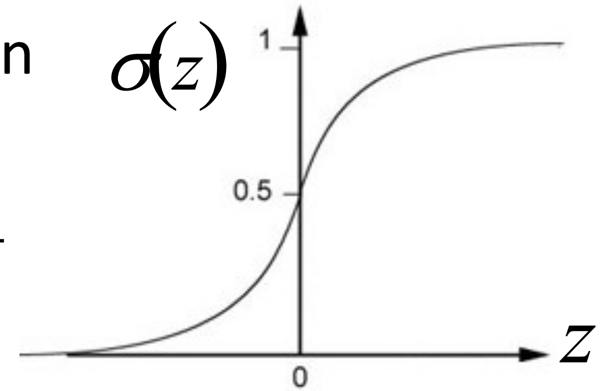


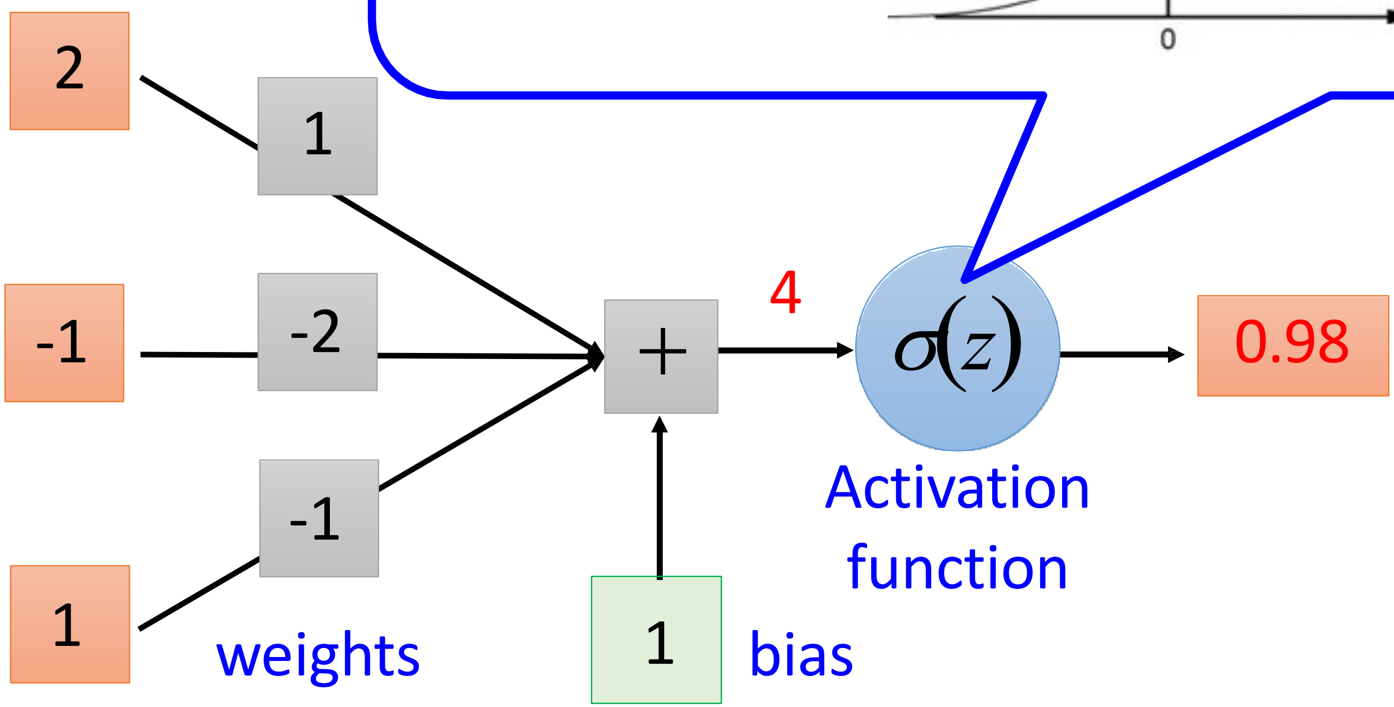
Neural Network

Neuron

Sigmoid Function

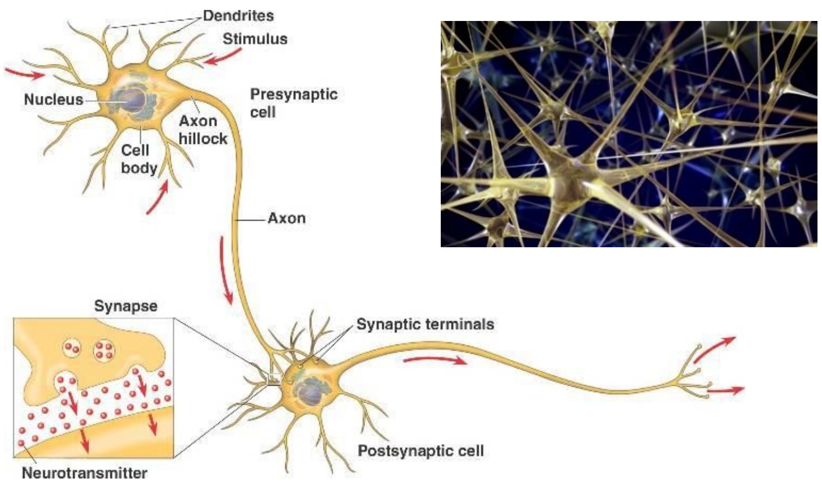
$$\sigma(z) = \frac{1}{1+e^{-z}}$$

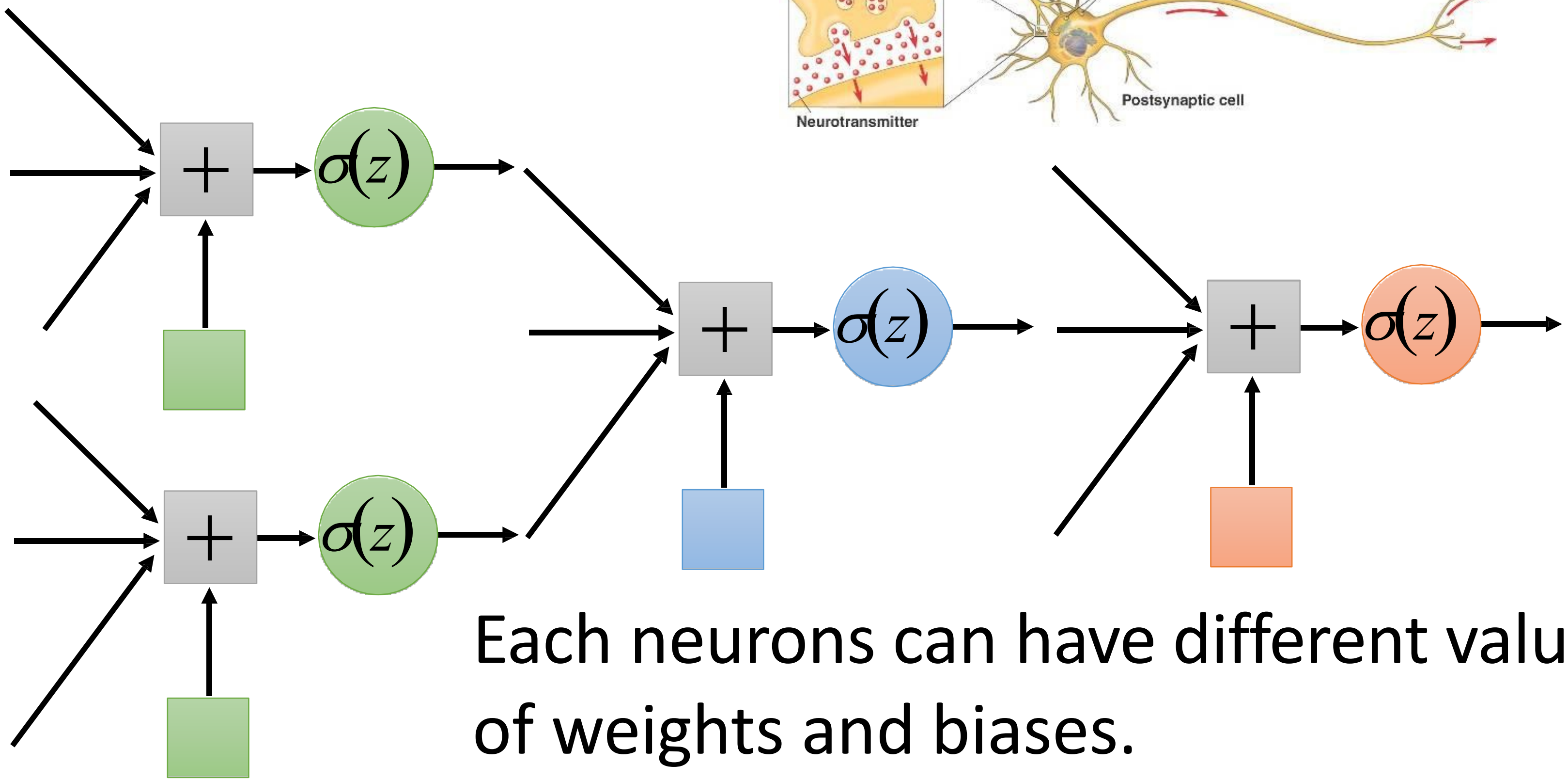




Neural Network

Different connections leads to different network structure

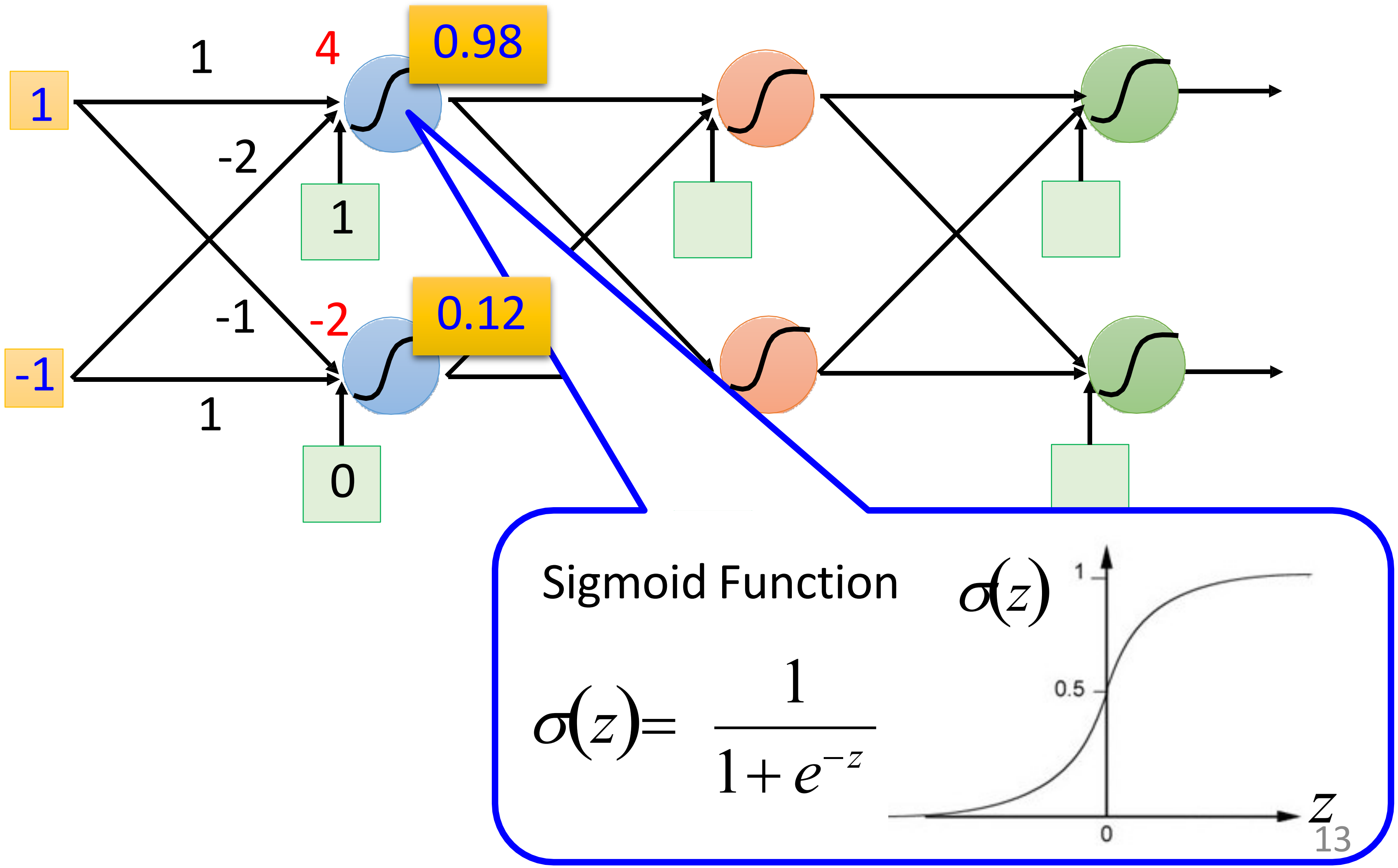




Each neurons can have different values of weights and biases.

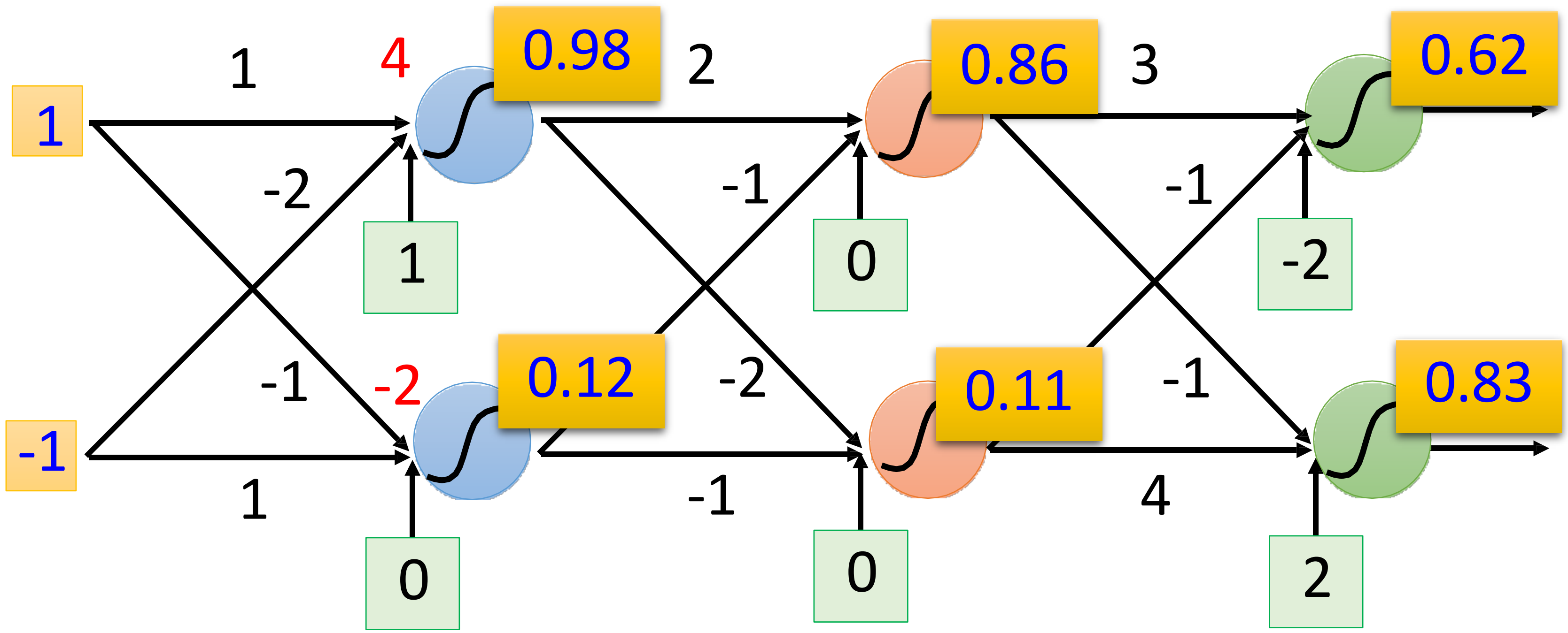
Weights and biases are network parameters θ

Fully Connect Feedforward Network

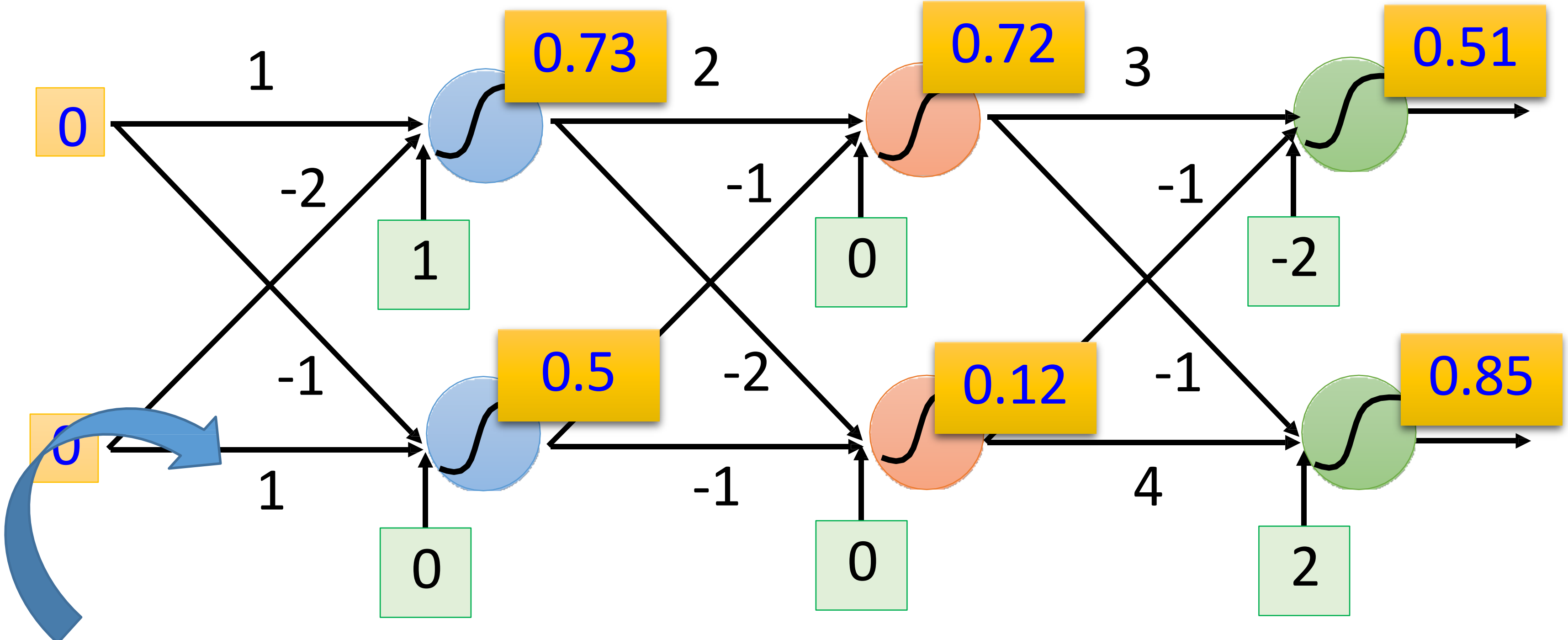


$$\sigma(z) = \frac{1}{1+e^{-z}}$$

Fully Connect Feedforward Network



Fully Connect Feedforward Network



This is a function.

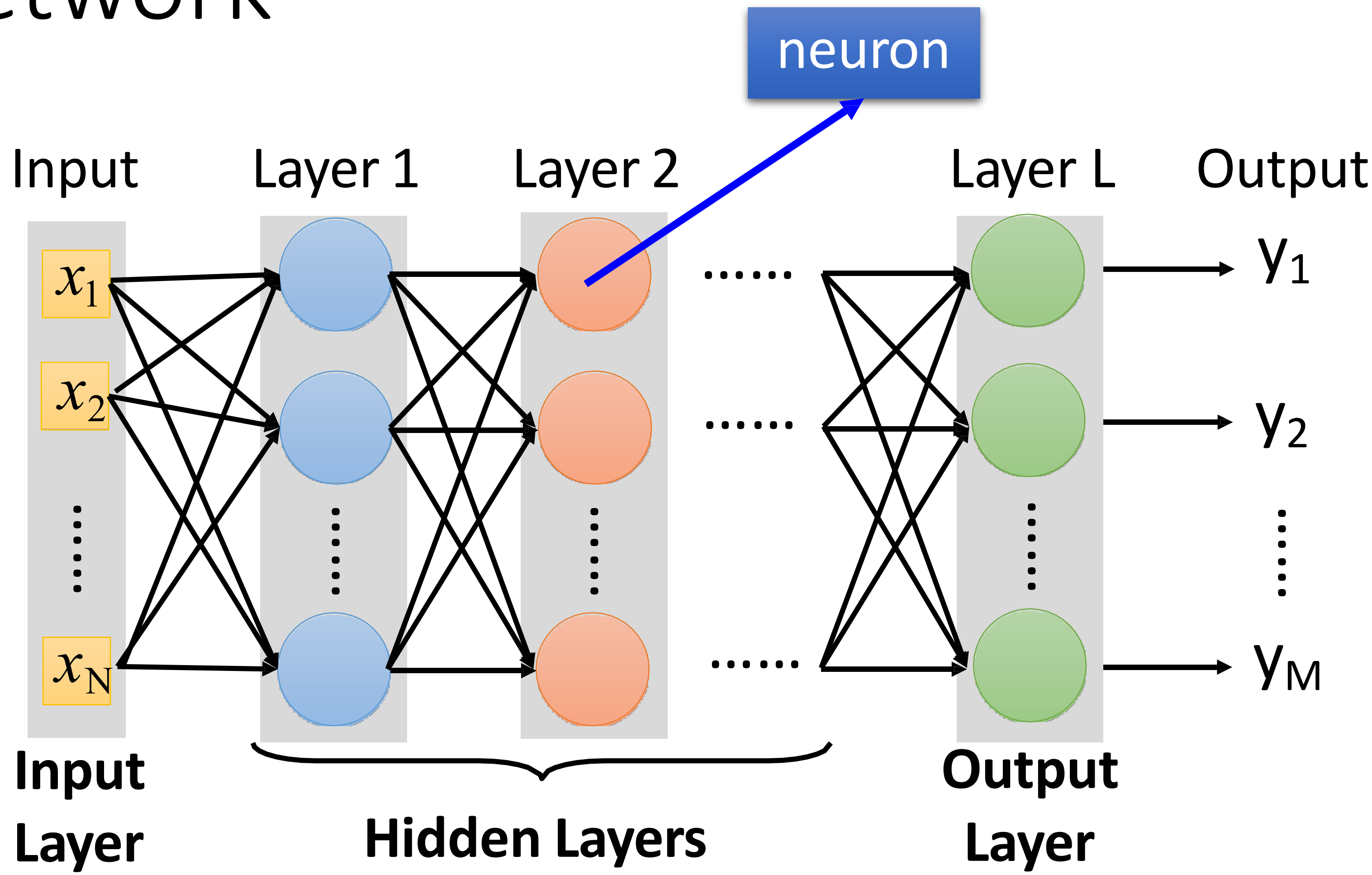
Input vector, output vector

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

Given parameters θ, define a function

Given network structure, define ***a function set***

Fully Connect Feedforward Network

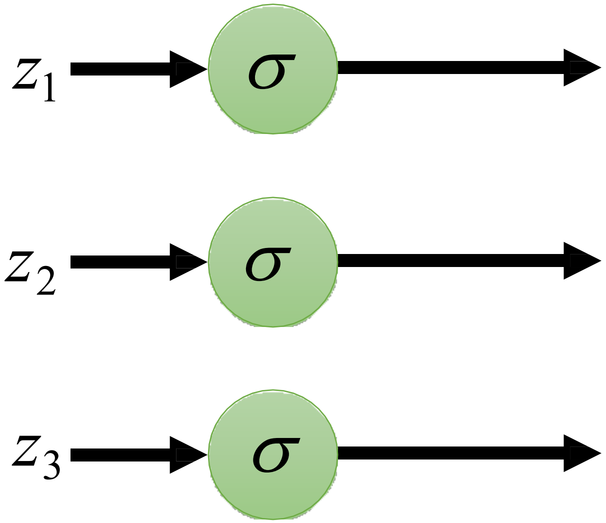


Deep means many hidden layers

Output Layer (Option)

- 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

Output Layer (Option)

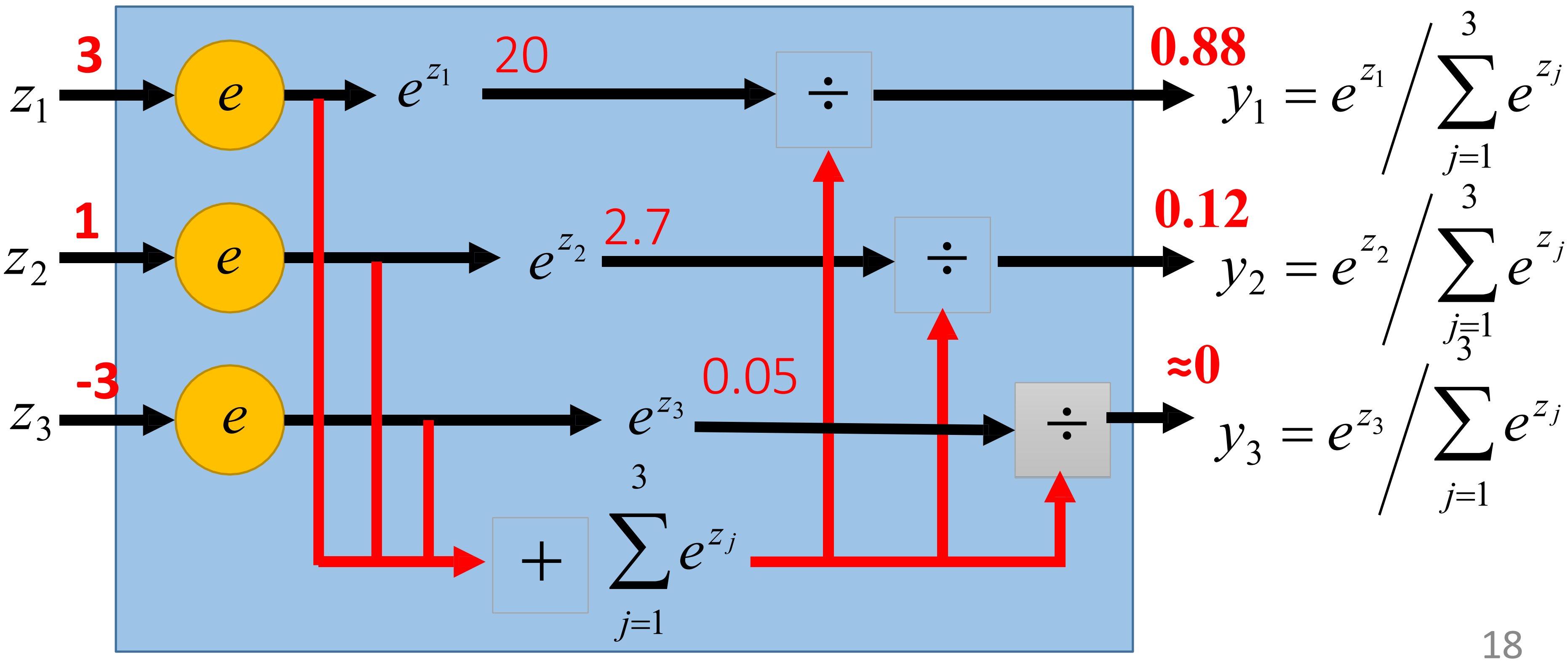
- Softmax layer as the output layer

Probability:

■ $1 > y_i > 0$

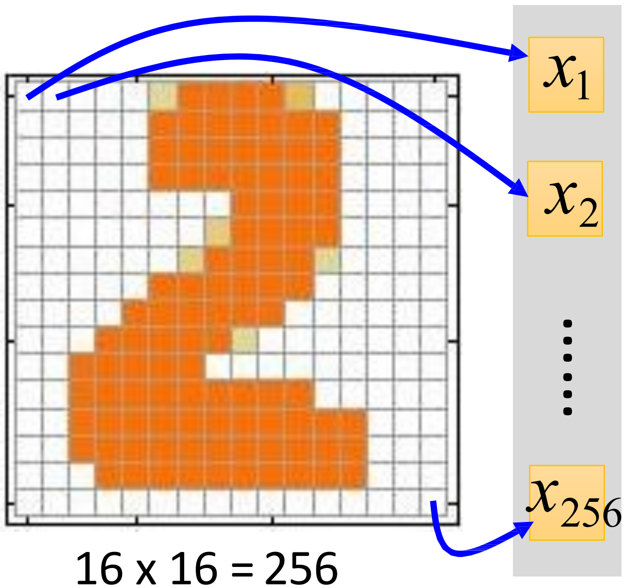
■ $_i\ y_i = 1$

Softmax Layer



Example Application

Input

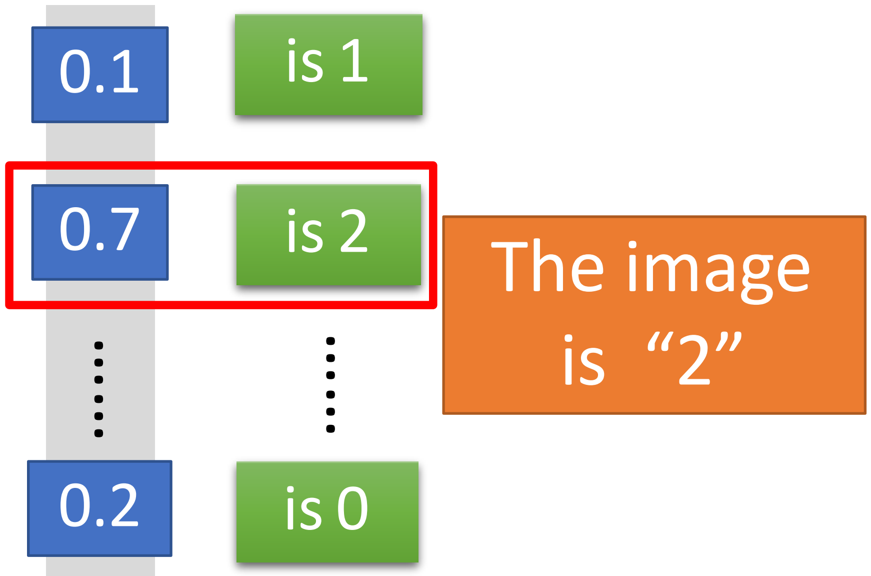


16 x 16 = 256

Ink → 1

No ink → 0

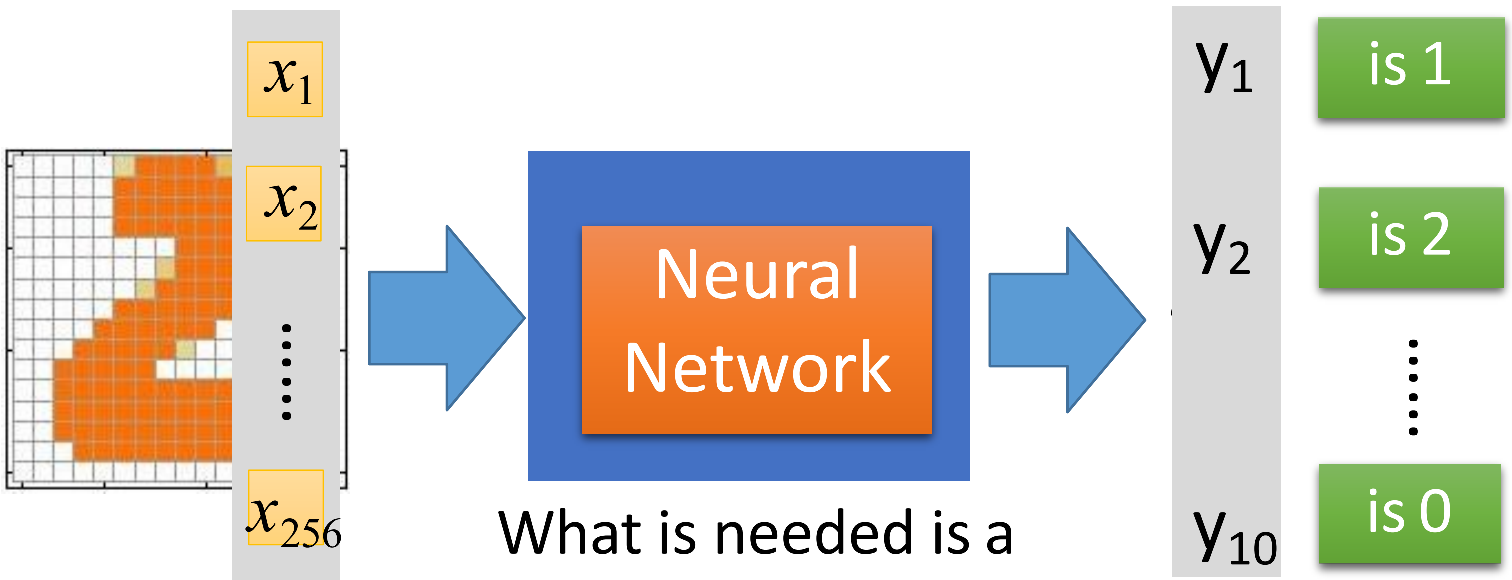
Output



Each dimension represents the confidence of a digit.

Example Application

- Handwriting Digit Recognition

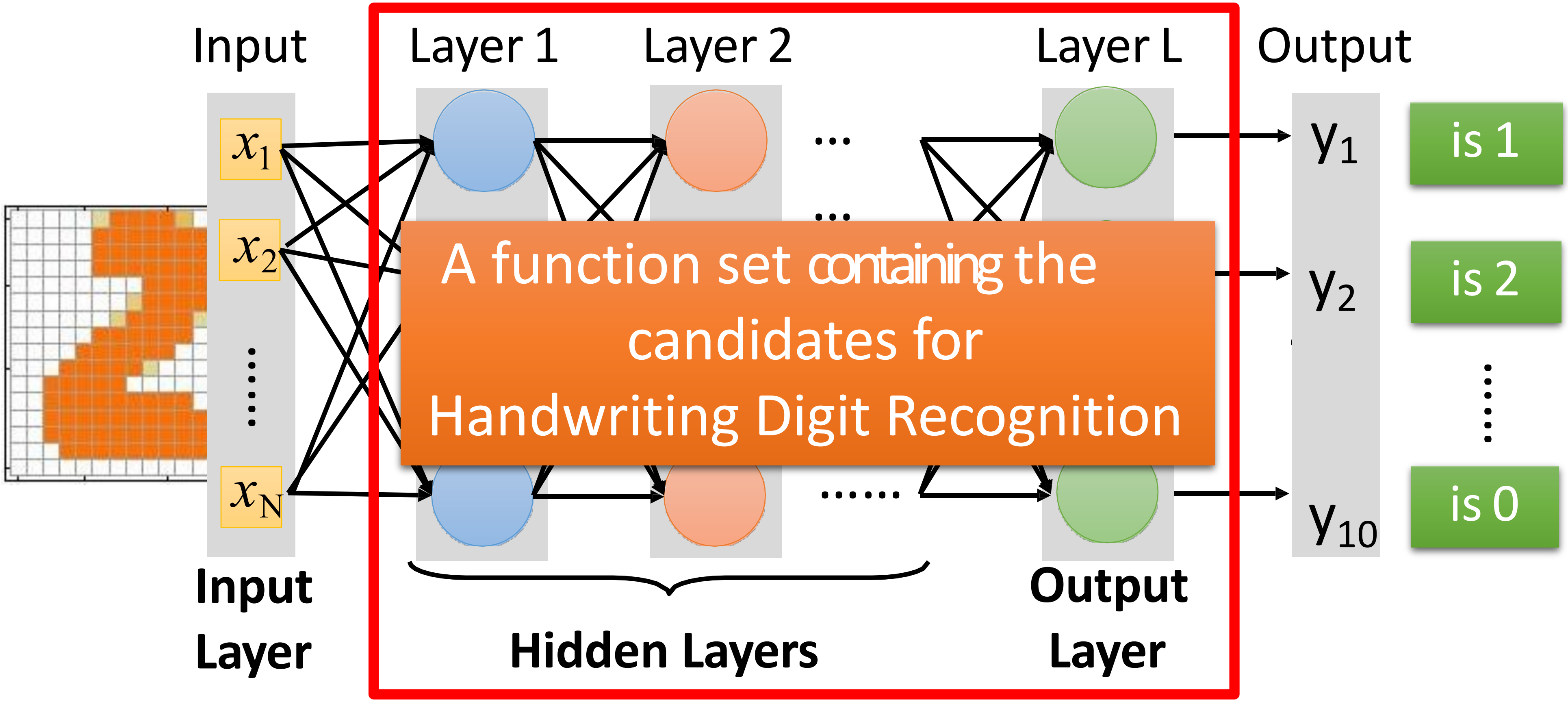


What is needed is a function ……

Input:
256-dim vector

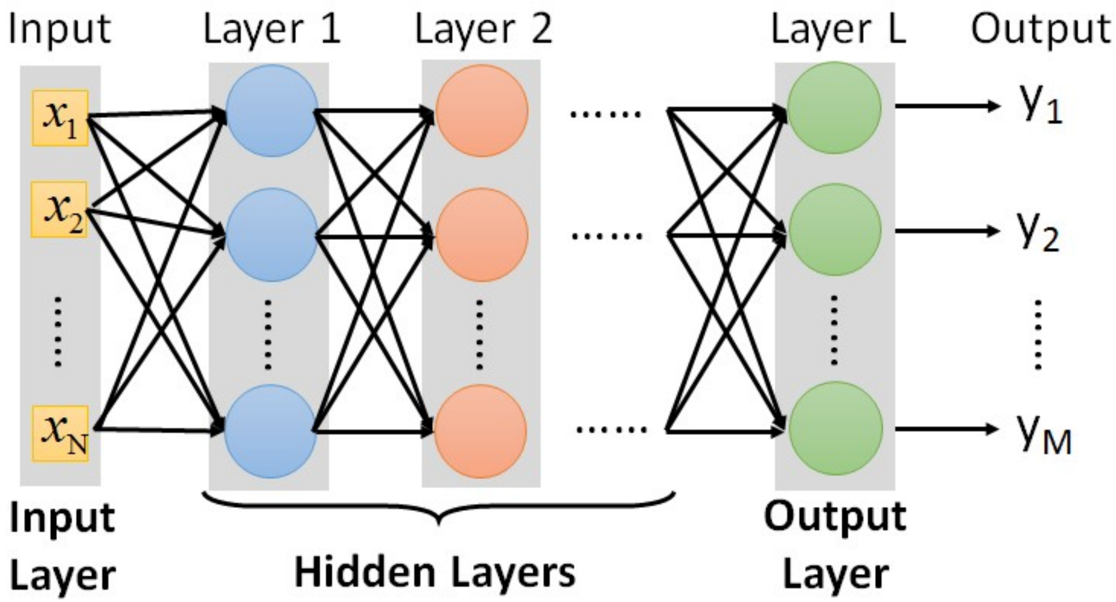
output:
10-dim vector

Example Application



You need to decide the network structure to let a good function in your function set.

FAQ



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

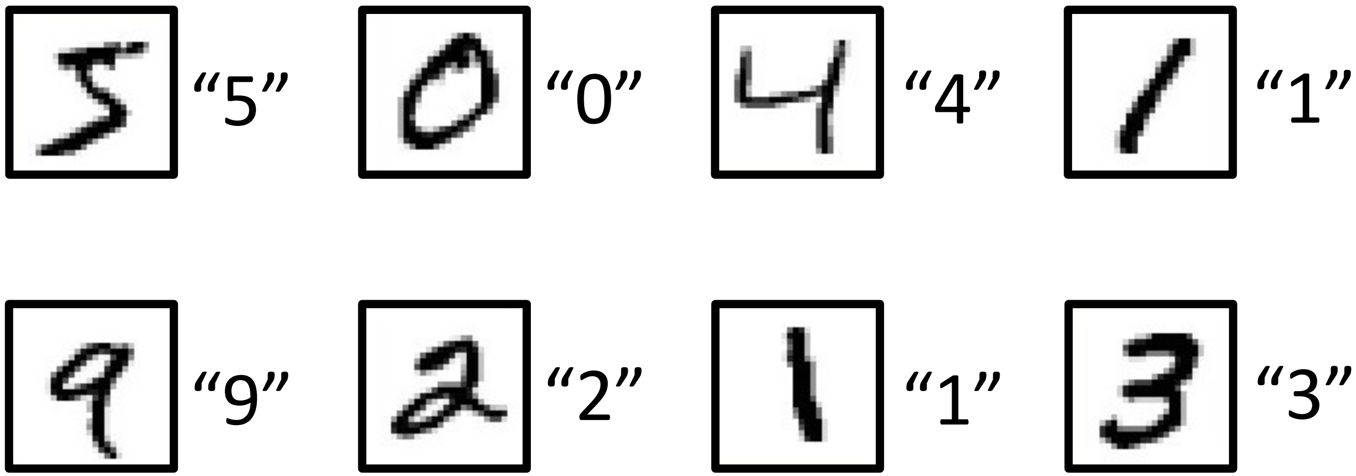
Trial and Error + Intuition

- Q: Can the structure be automatically determined?

Three Steps for Deep Learning

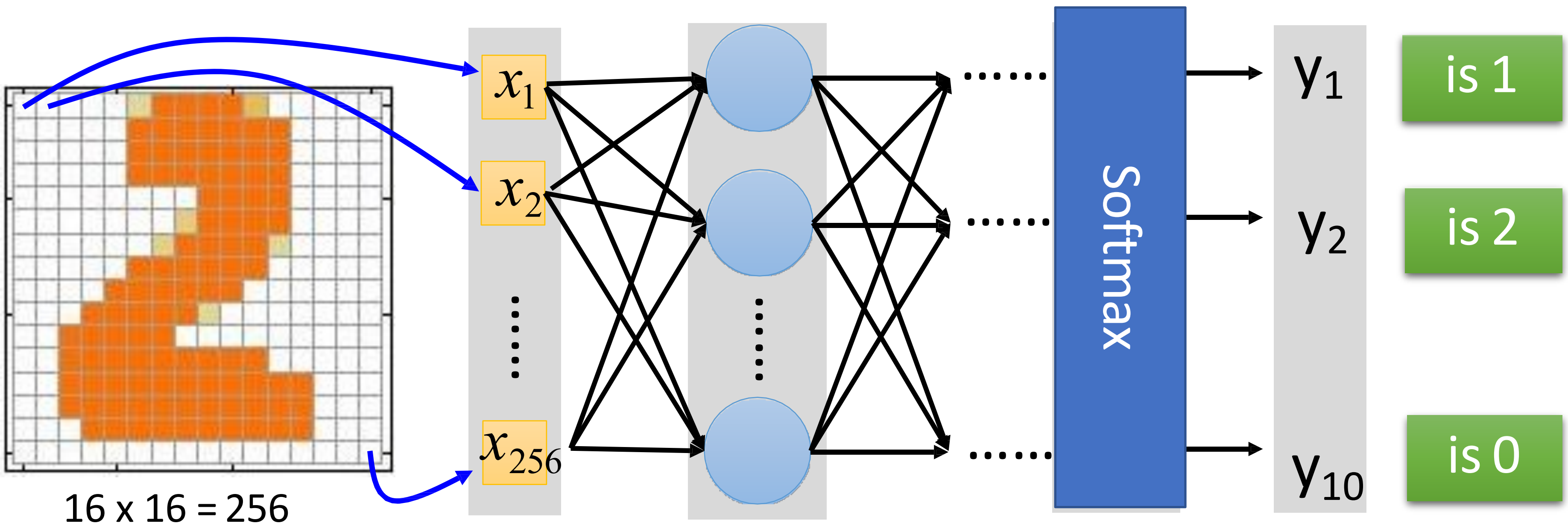
Training Data

- Preparing training data: images and their labels



The learning target is defined on the training data.

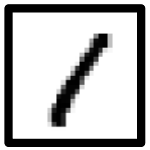
Learning Target



Ink → 1

No ink → 0

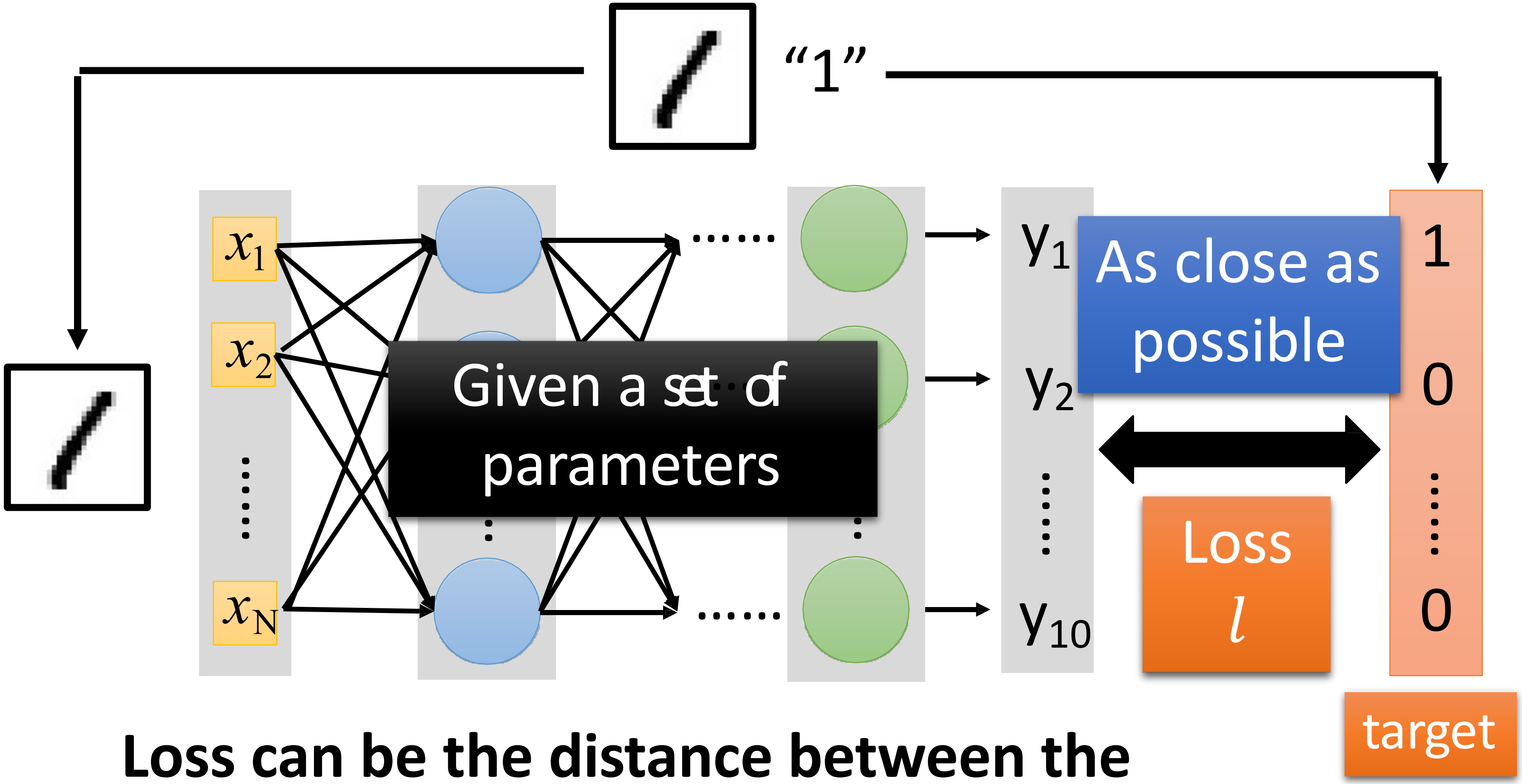
The learning target is ……

Input:  → y_1 has the maximum value

Input:  → y_2 has the maximum value

Loss

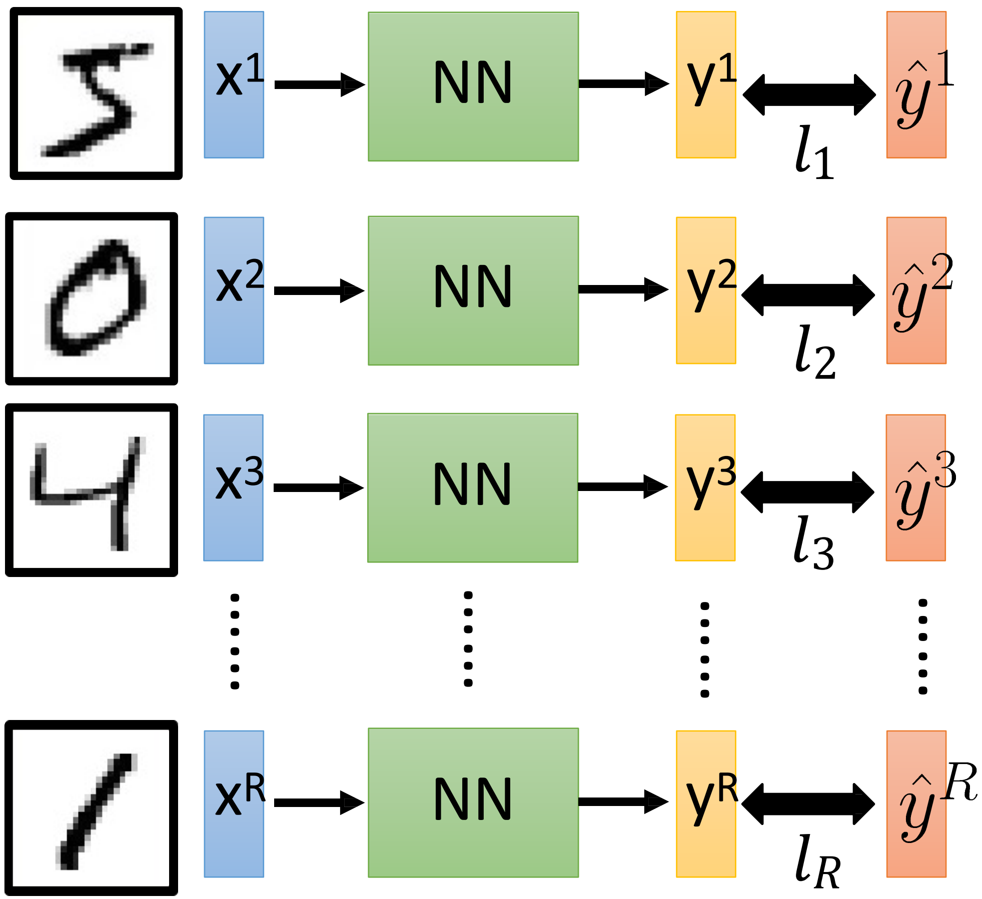
A good function should make the loss of all examples as small as possible.



Loss can be the distance between the network output and target

Total Loss

For all training data …



Total Loss:

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

As small as possible

Find ***a function in function set*** that minimizes total loss L

Find ***the network parameters θ^**** that minimize total loss L

Three Steps for Deep Learning

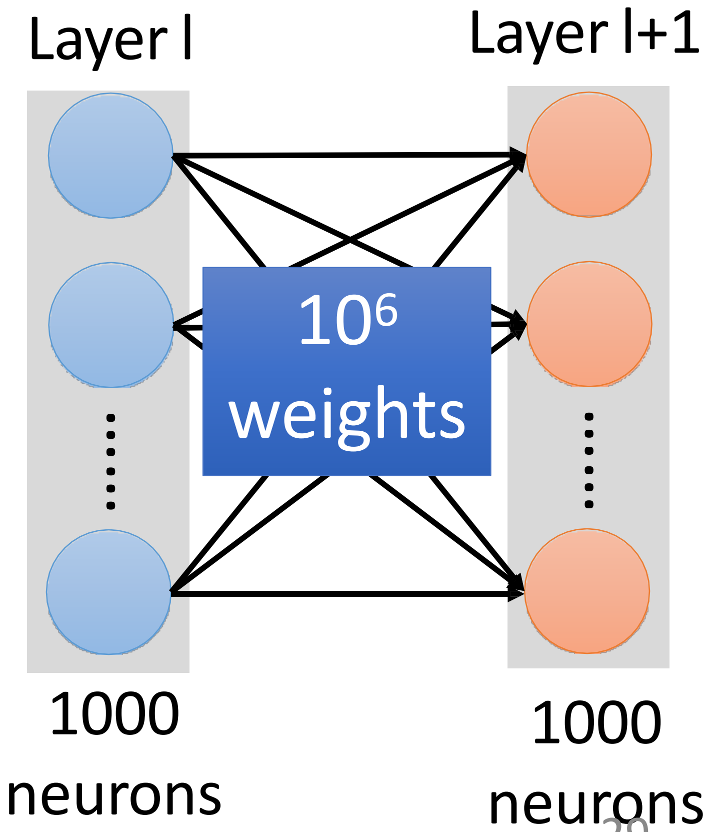
How to pick the best function

Find ***network parameters θ^**** that minimize total loss L

Enumerate all possible values

Network parameters $\theta =$
$\underbrace{\{w_1, w_2, w_3, \cdots, b_1, b_2, b_3, \cdots\}}$
Millions of parameters

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



Gradient Descent

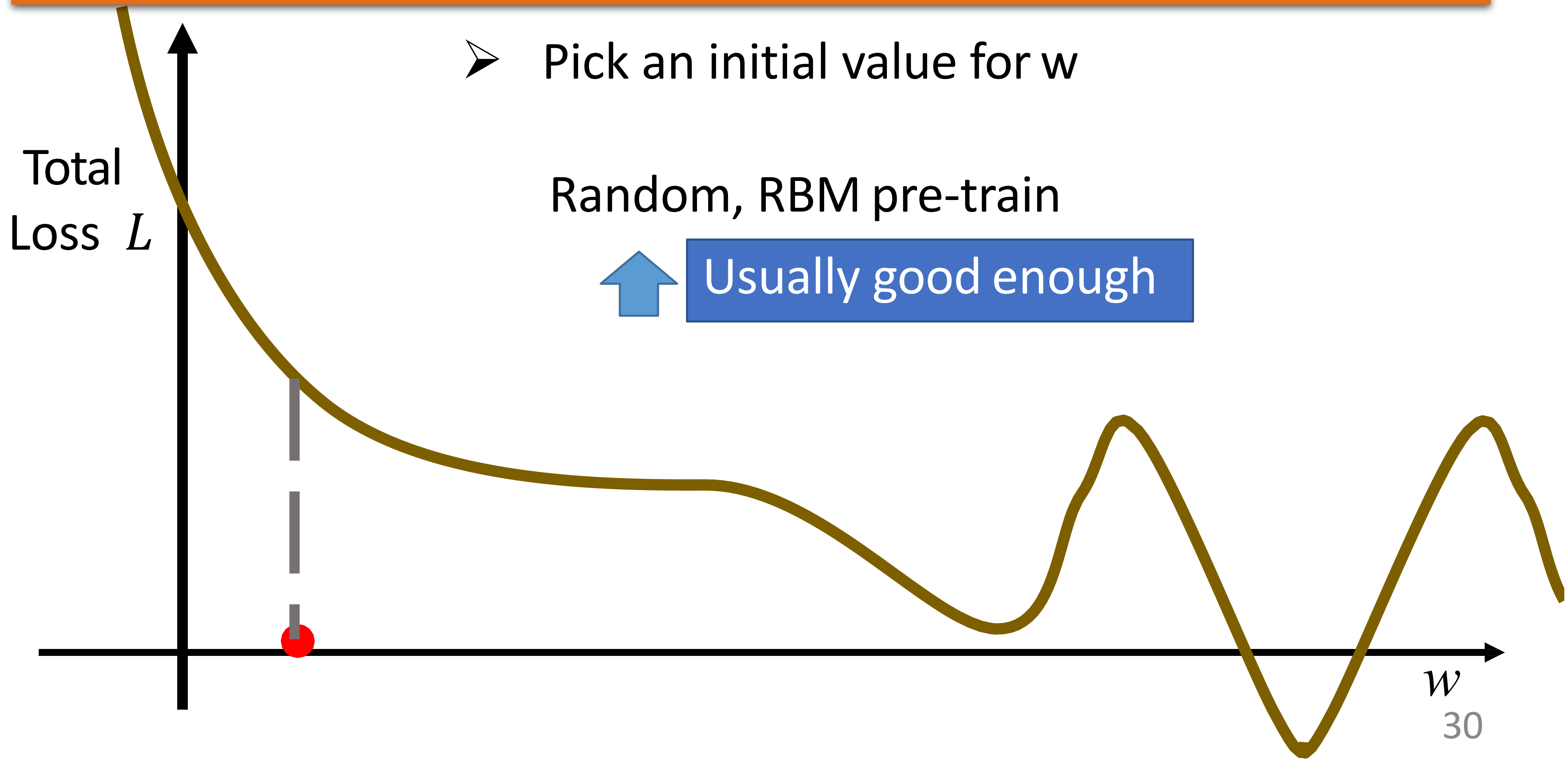
Network parameters $\theta = \{w_1, w_2, \cdots, b_1, b_2, \cdots\}$

Find ***network parameters*** $\boldsymbol{\theta^*}$ that minimize total loss L

- Pick an initial value for w

Random, RBM pre-train

Usually good enough



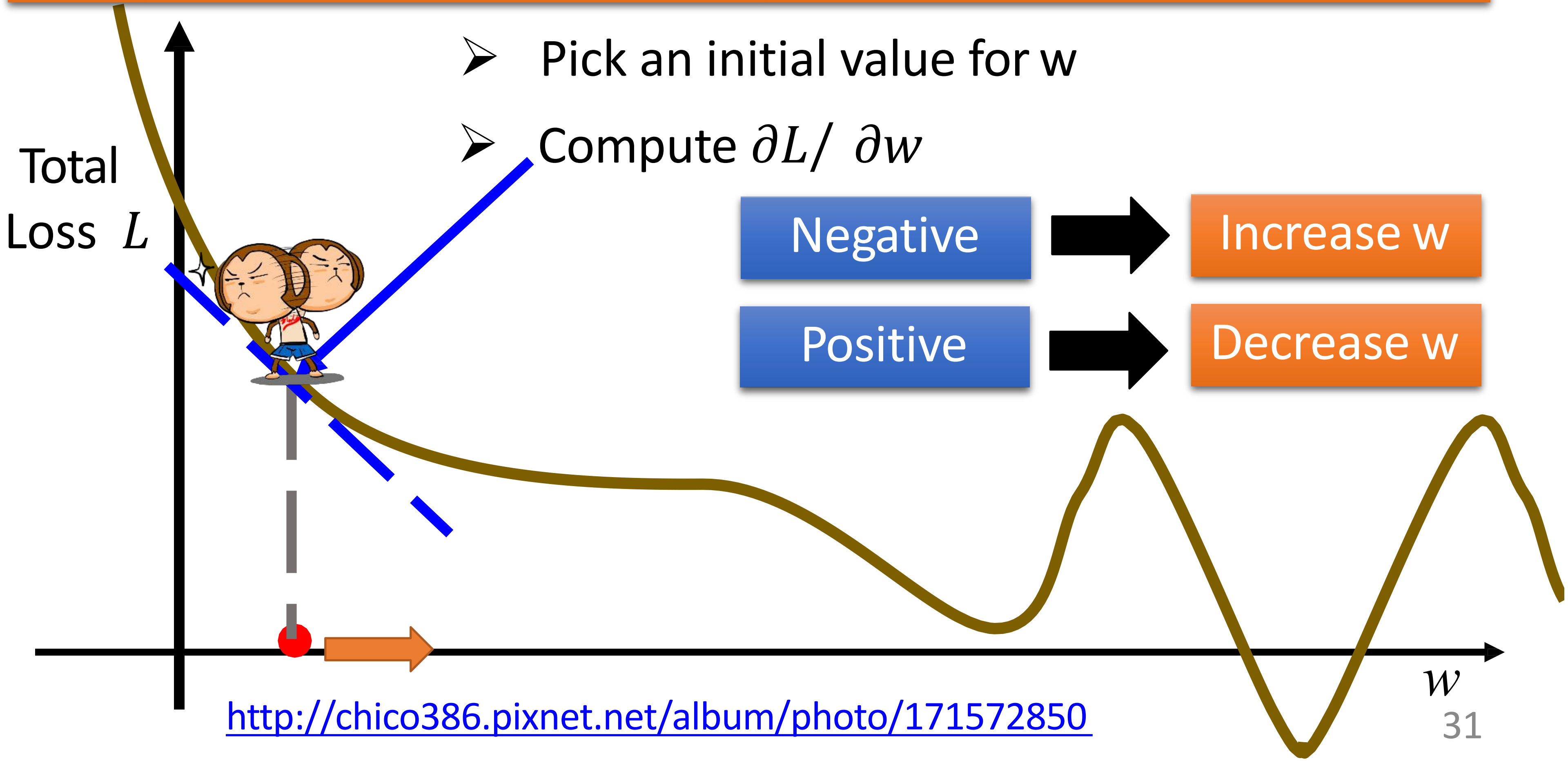
Gradient Descent

Network parameters $\theta = \{w_1, w_2, \cdots, b_1, b_2, \cdots\}$

Find ***network parameters*** $\boldsymbol{\theta}^*$ that minimize total loss L

- Pick an initial value for w
- Compute $\partial L / \partial w$

Negative	→	Increase w
Positive	→	Decrease w



http://chico386.pixnet.net/album/photo/171572850

Gradient Descent

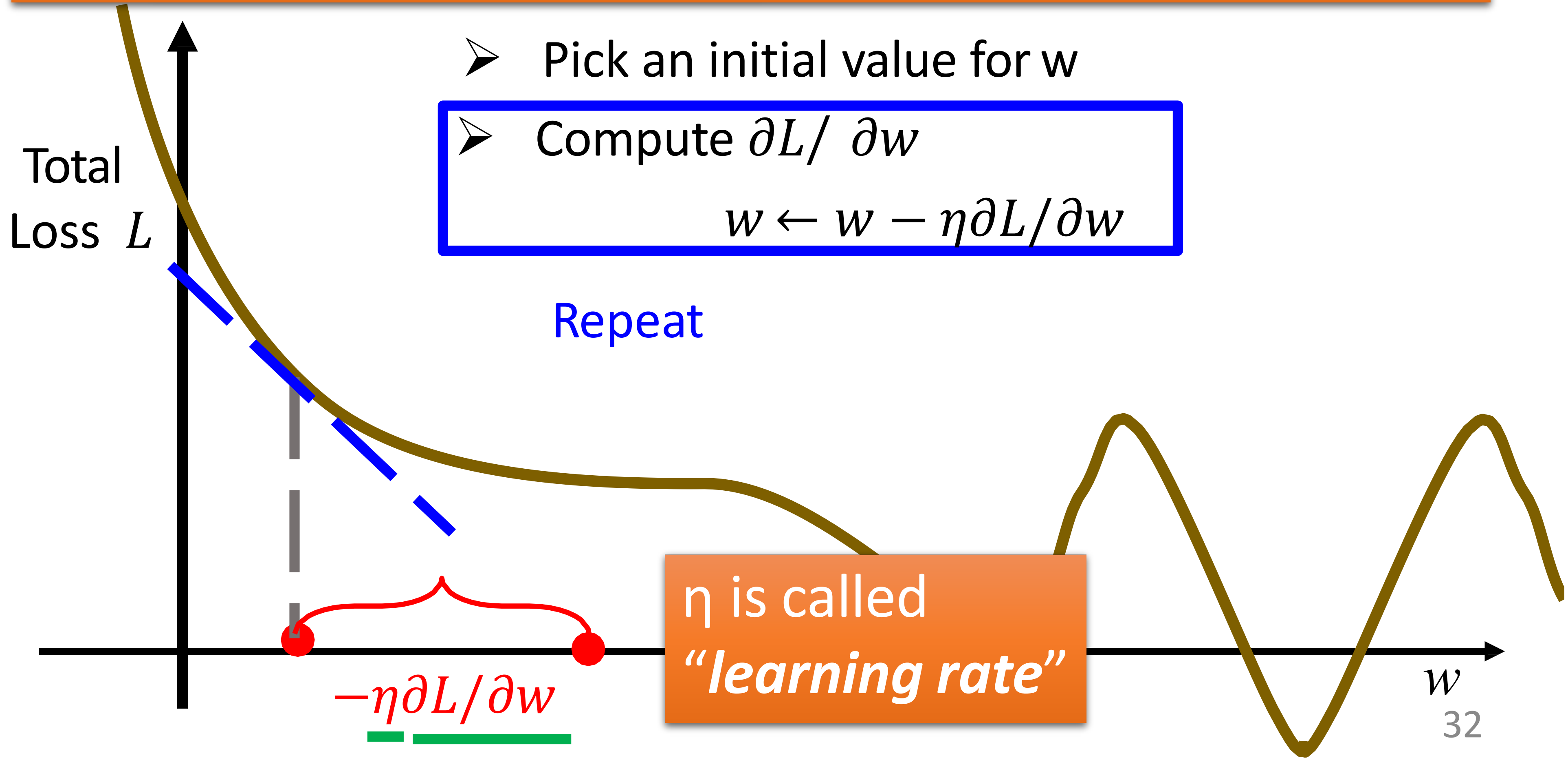
Network parameters $\theta = \{ w_1, w_2, \cdots, b_1, b_2, \cdots \}$

Find ***network parameters*** $\boldsymbol{\theta}^*$ that minimize total loss L

- Pick an initial value for w
- Compute $\partial L / \partial w$

  $w \leftarrow w - \eta \partial L / \partial w$

Repeat



η is called ***"learning rate"***

Gradient Descent

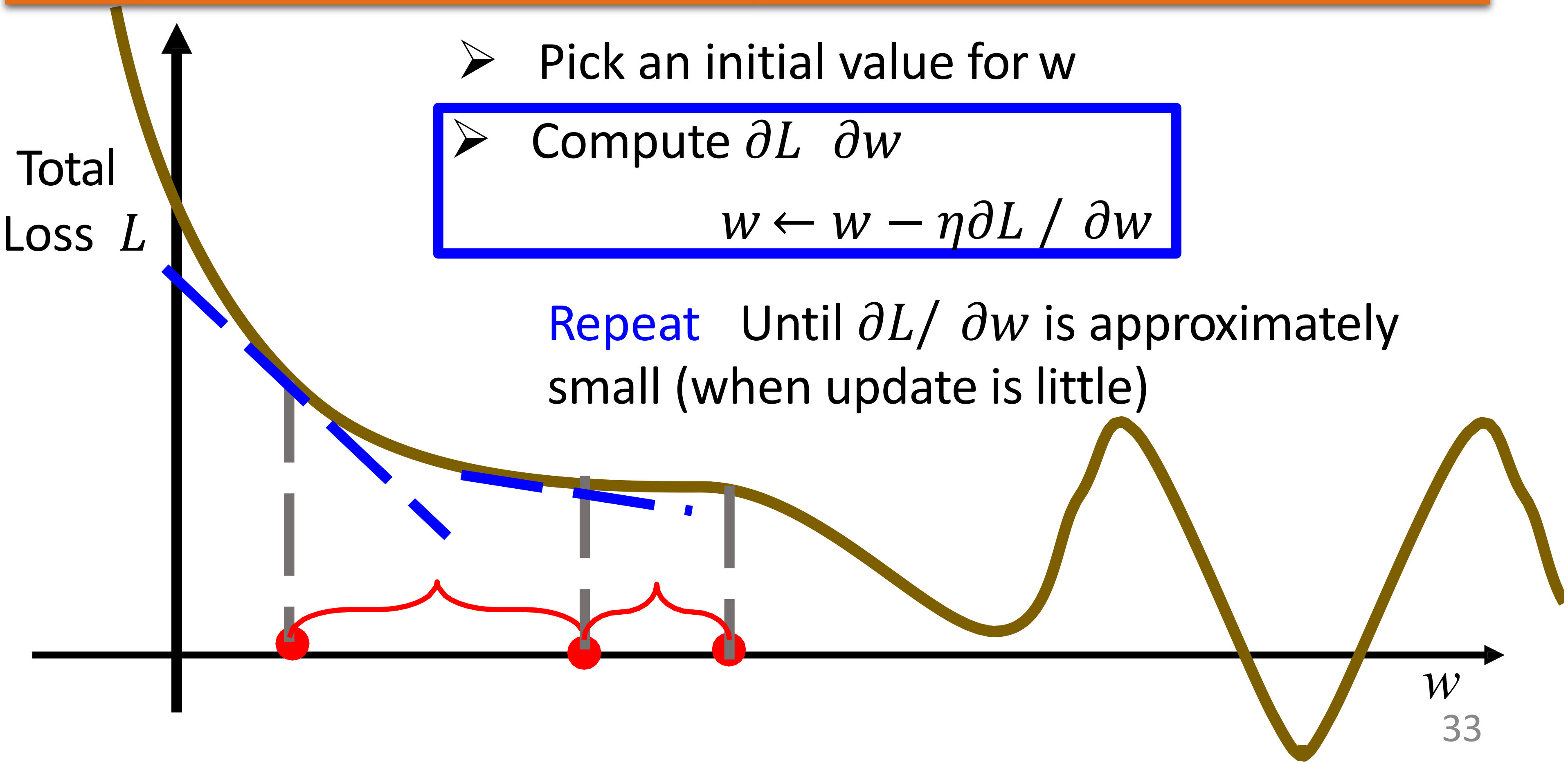
Network parameters $\theta = \{w_1, w_2, \cdots, b_1, b_2, \cdots\}$

Find *network parameters* $\boldsymbol{\theta}^*$ that minimize total loss L

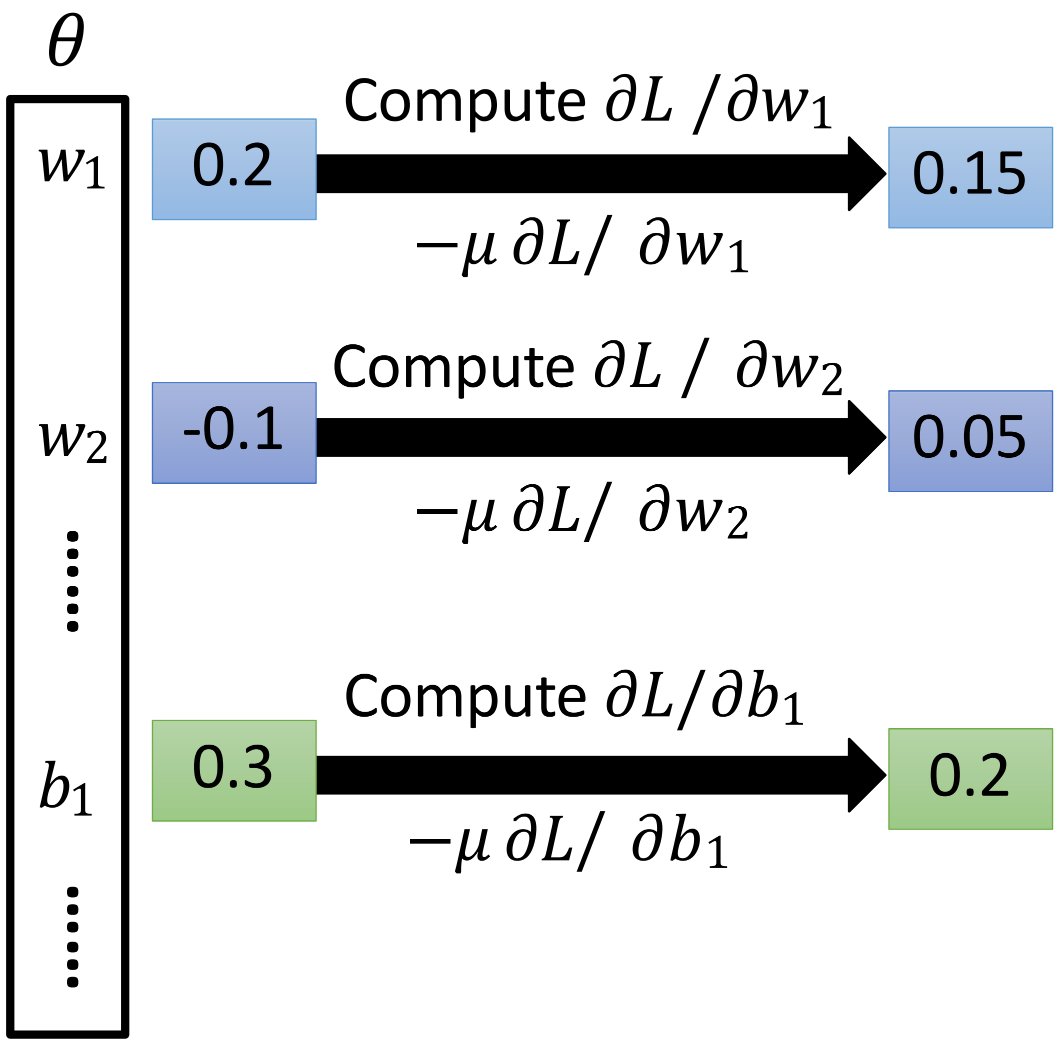
- Pick an initial value for w
- Compute $\partial L \quad \partial w$

  $w \leftarrow w - \eta \partial L \, / \, \partial w$

Repeat Until $\partial L / \, \partial w$ is approximately small (when update is little)



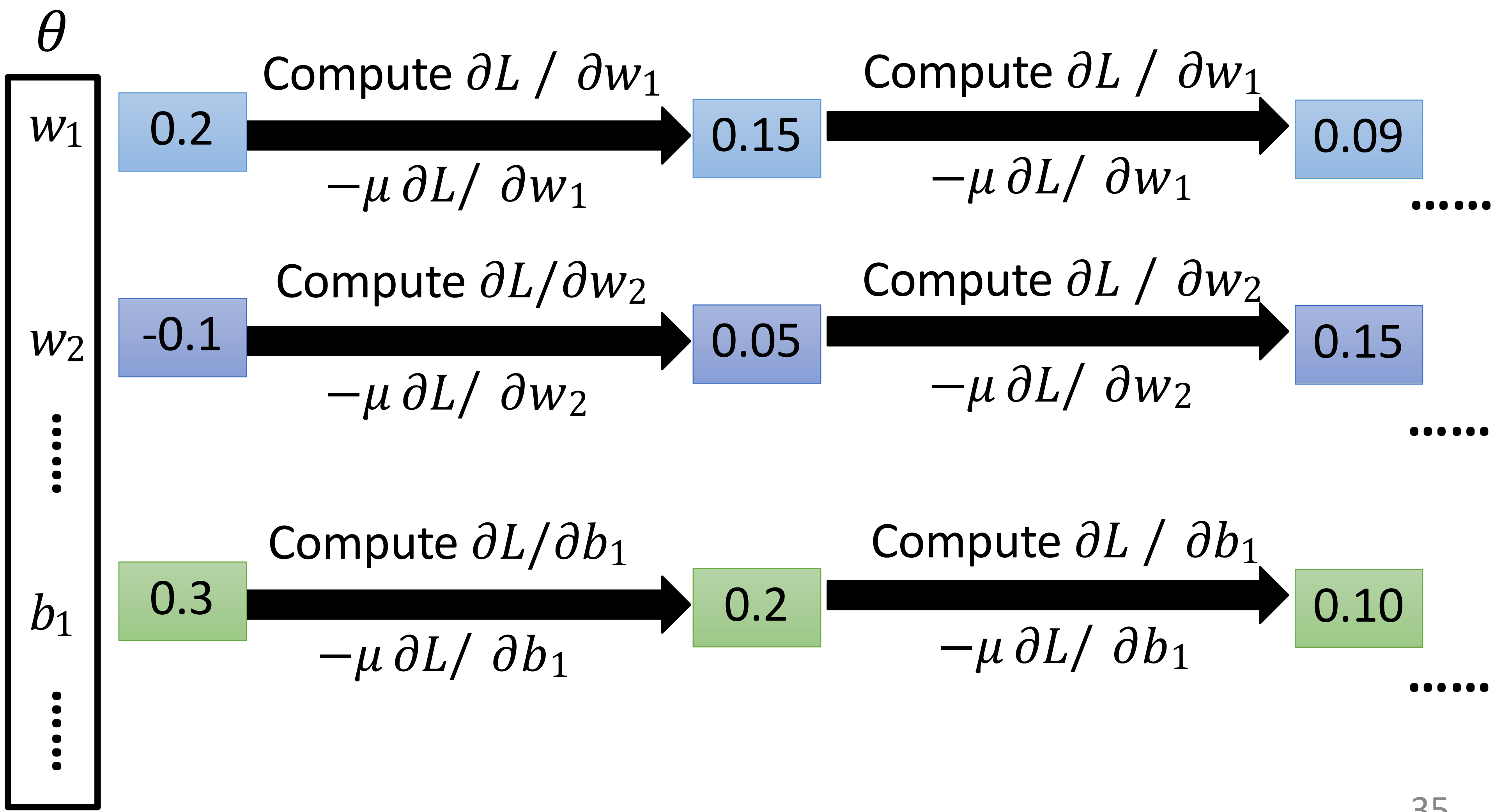
Gradient Descent



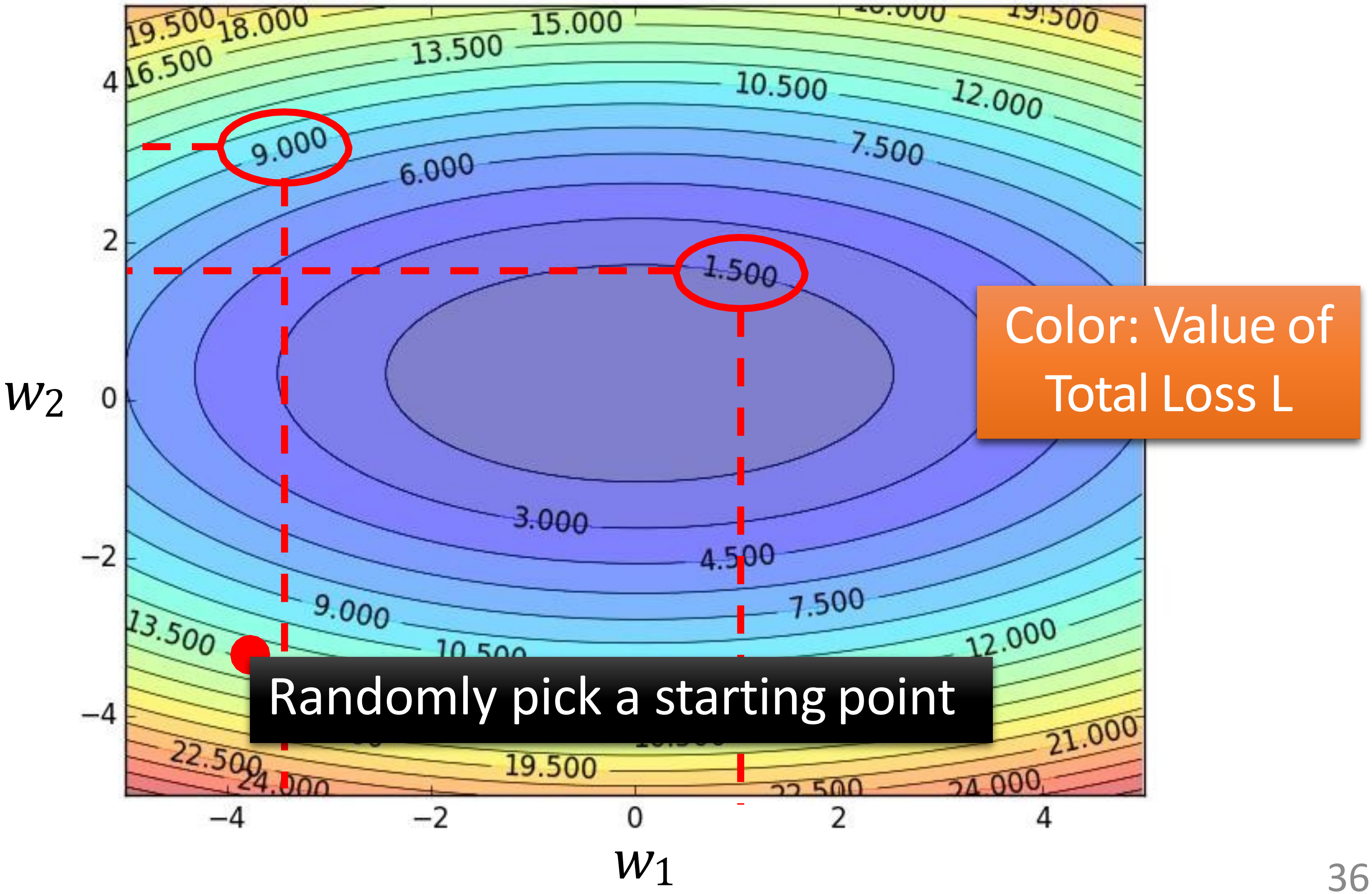
$$\nabla L = \begin{bmatrix} \dfrac{\partial L}{\partial w_1} \\ \dfrac{\partial L}{\partial w_2} \\ \vdots \\ \dfrac{\partial L}{\partial b_1} \\ \vdots \end{bmatrix}$$

gradient

Gradient Descent

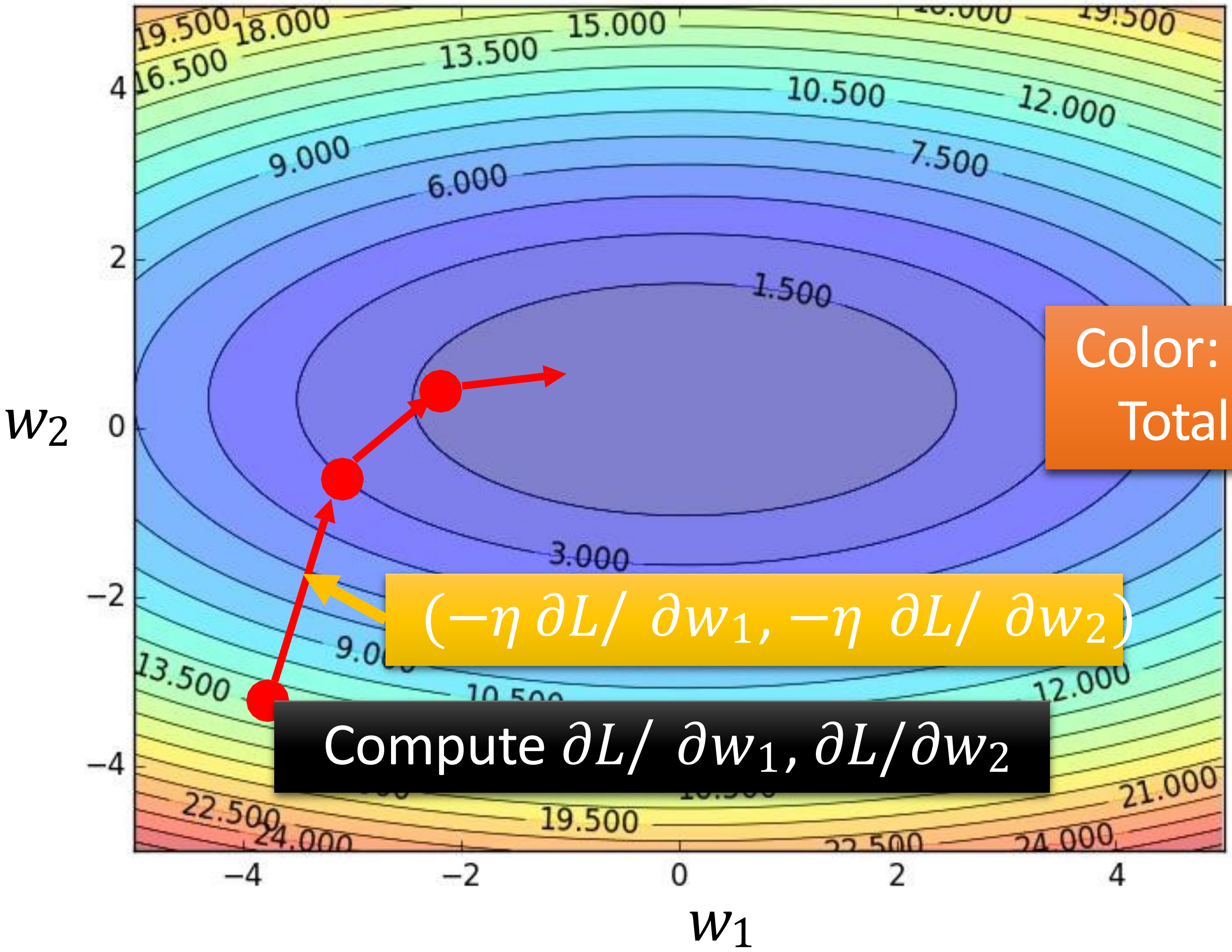


Gradient Descent



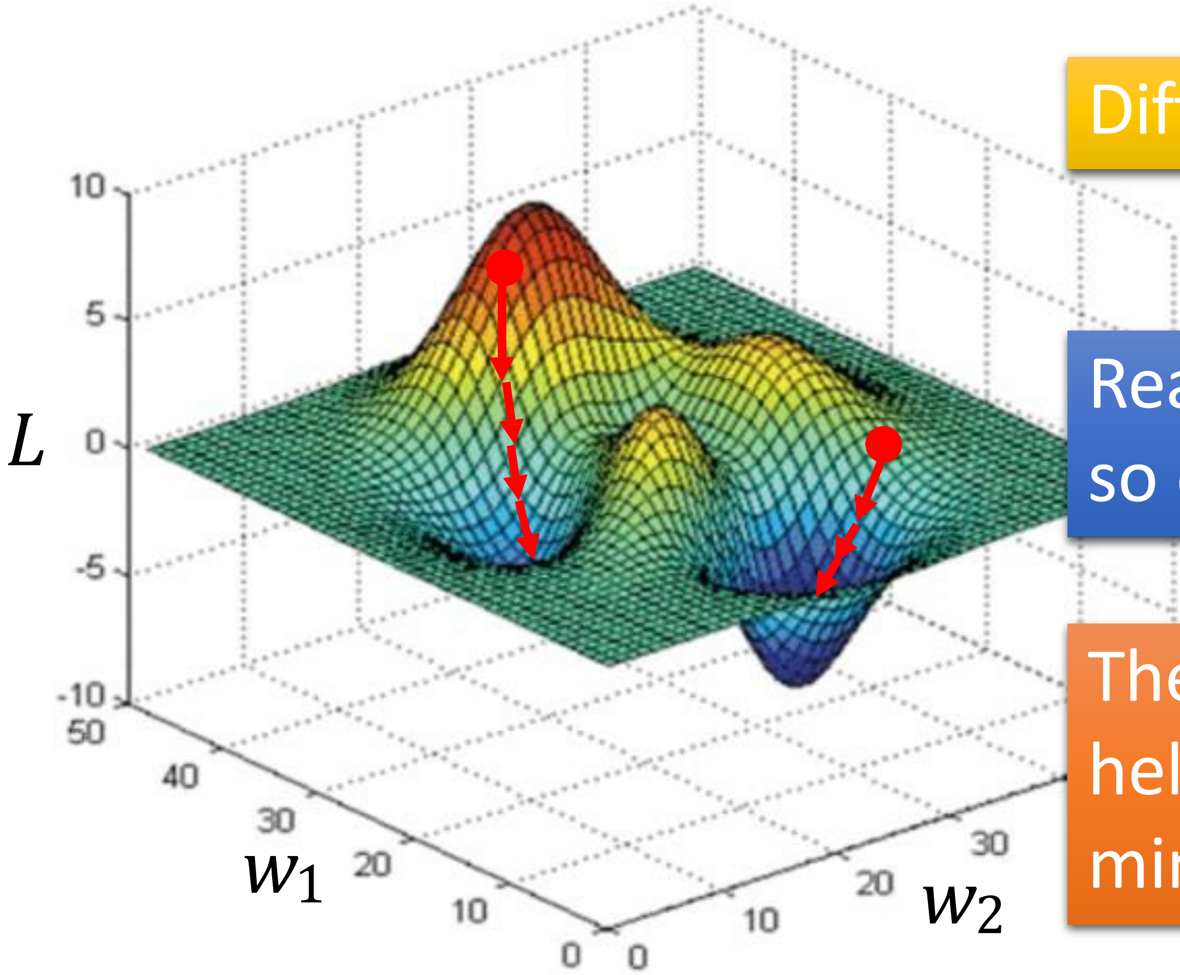
Gradient Descent

Hopfully, we would reach a minima .....



Gradient Descent - Difficulty

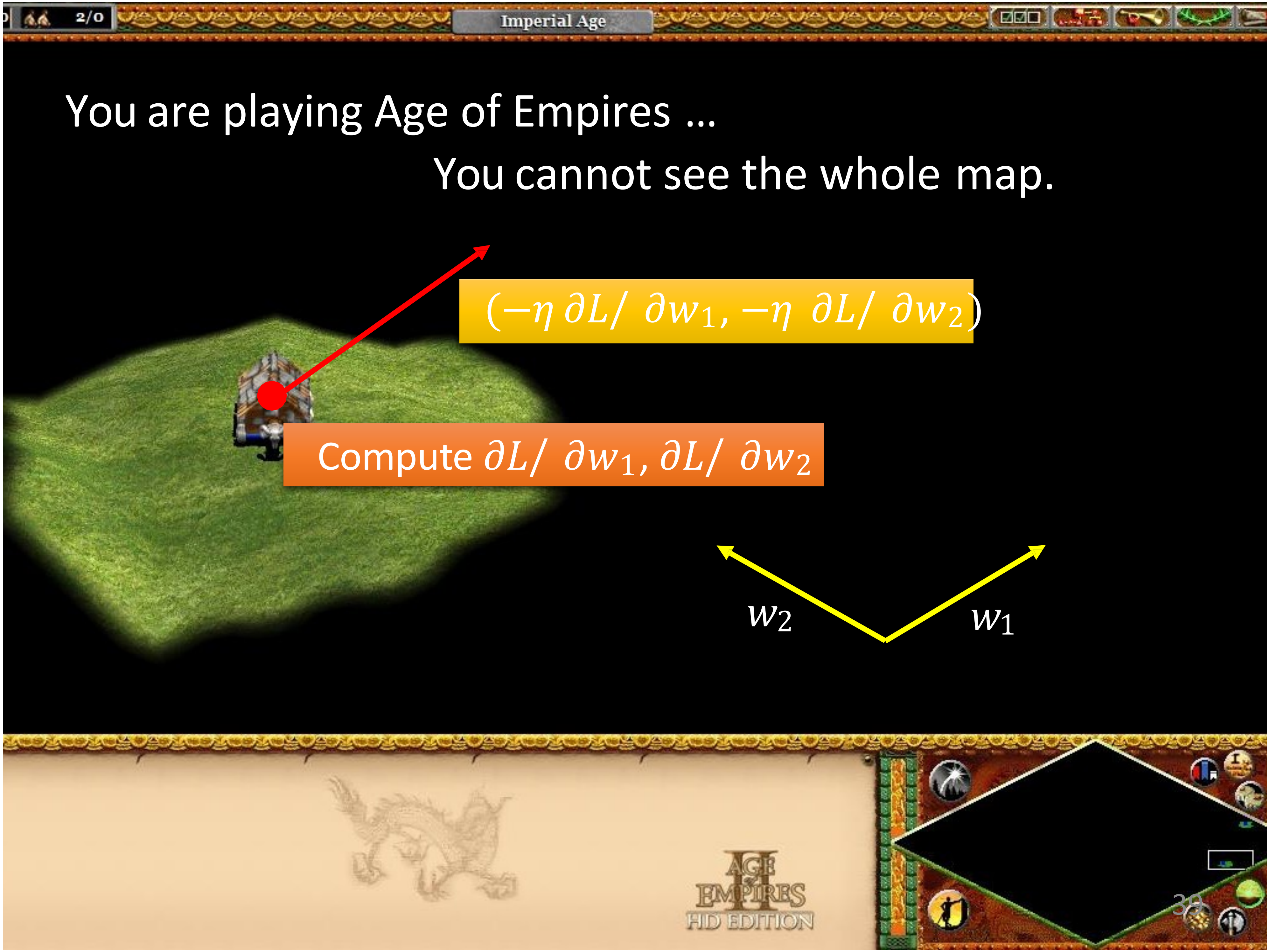
- Gradient descent never guarantee global minima



Different initial point

Reach different minima, so different results

There are some tips to help you avoid local minima, no guarantee.


2/0
Imperial Age
You are playing Age of Empires …
You cannot see the whole map.
$(-\eta\, \partial L/\, \partial w_1, -\eta\ \partial L/\ \partial w_2)$
Compute $\partial L/\ \partial w_1, \partial L/\ \partial w_2$
w_2
w_1
AGE of EMPIRES II HD EDITION

Backpropagation

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

TensorFlow torch theano libdnn

台大周伯威
同學開發

Caffe Microsoft CNTK Chainer mxnet

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