深度学习概述

Adept form Hung-yi Lee’s tutorial

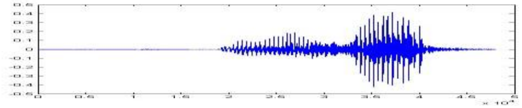
Outline

Introduction to Machine Learning

“Three Steps” for Deep Learning

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)

Framework

Image Recognition:

$f($  $)=$ "cat"

A set of function

Model

$f_1, f_2 \cdots$

$f_1($  $)=$ "cat"

$f_2($  $)=$ "money"

$f_1($  $)=$ "dog"

$f_2($  $)=$ "snake"

Framework

Image Recognition:

$f($  $) =$ "cat"

A set of function

Model

$f_1, f_2 \cdots$

$f_1($  $) =$ "cat"

$f_2($  $) =$ "money"

Better!

$f_1($  $) =$ "dog"

$f_2($  $) =$ "snake"

Goodness of function f

Training Data

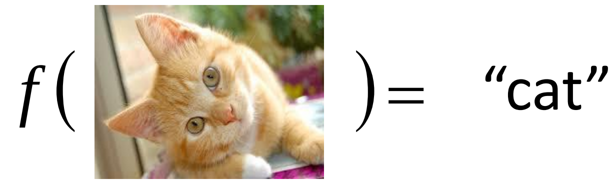
Supervised Learning

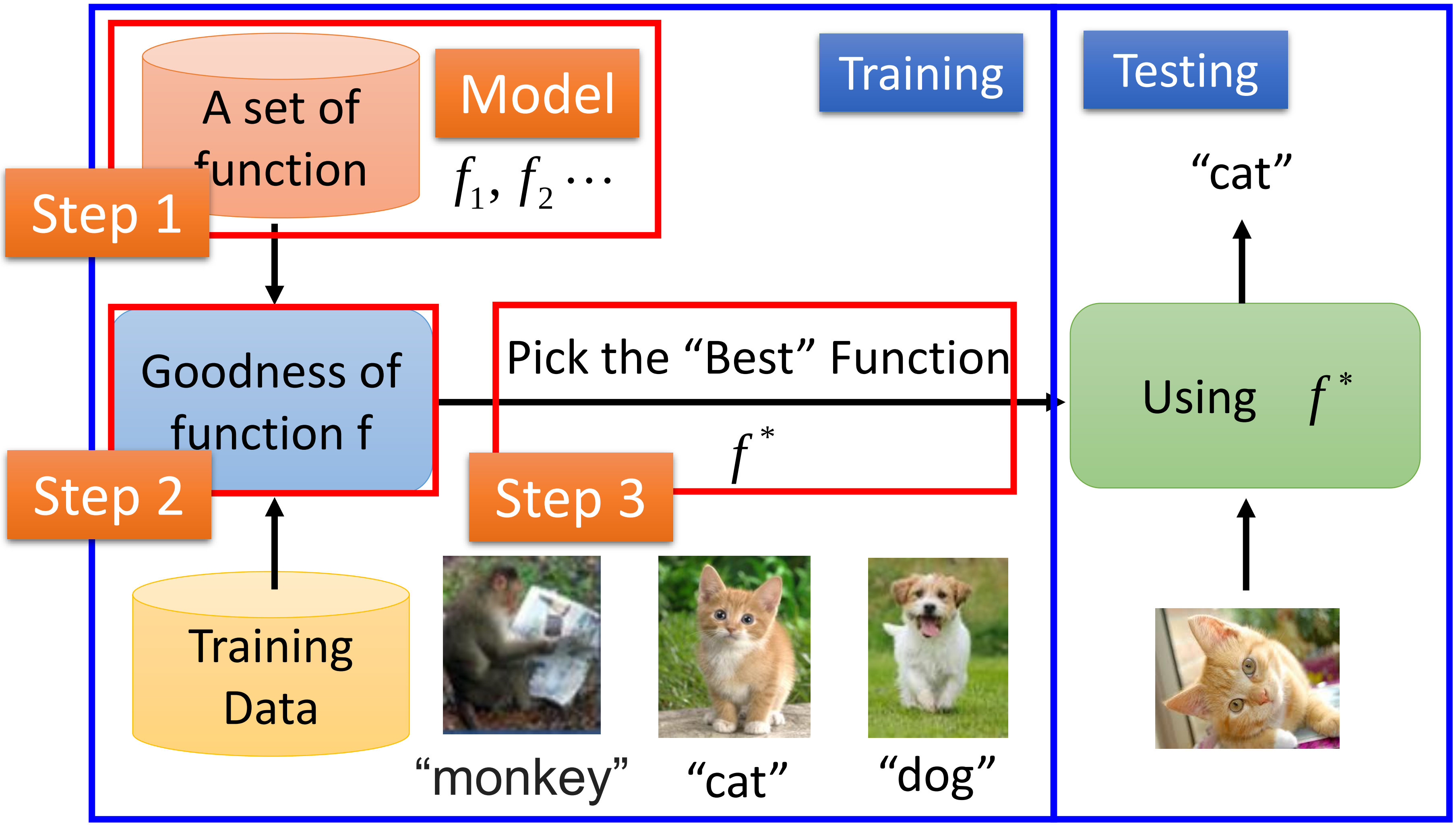
function input:   

function output: "monkey" "cat" "dog"

Framework

Image Recognition:





II. Three Steps for Deep Learning

Step 1: define a set of function

Neural Network

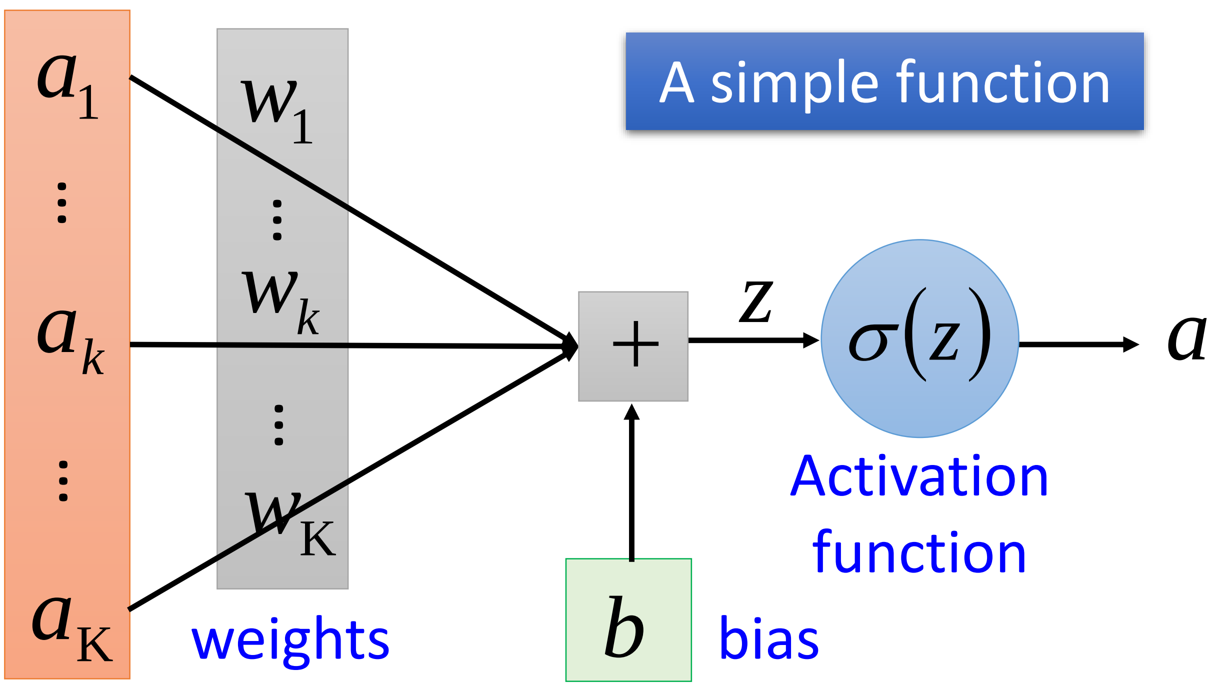
Step 2: goodness of function

Step 3: pick the best function

Neural Network

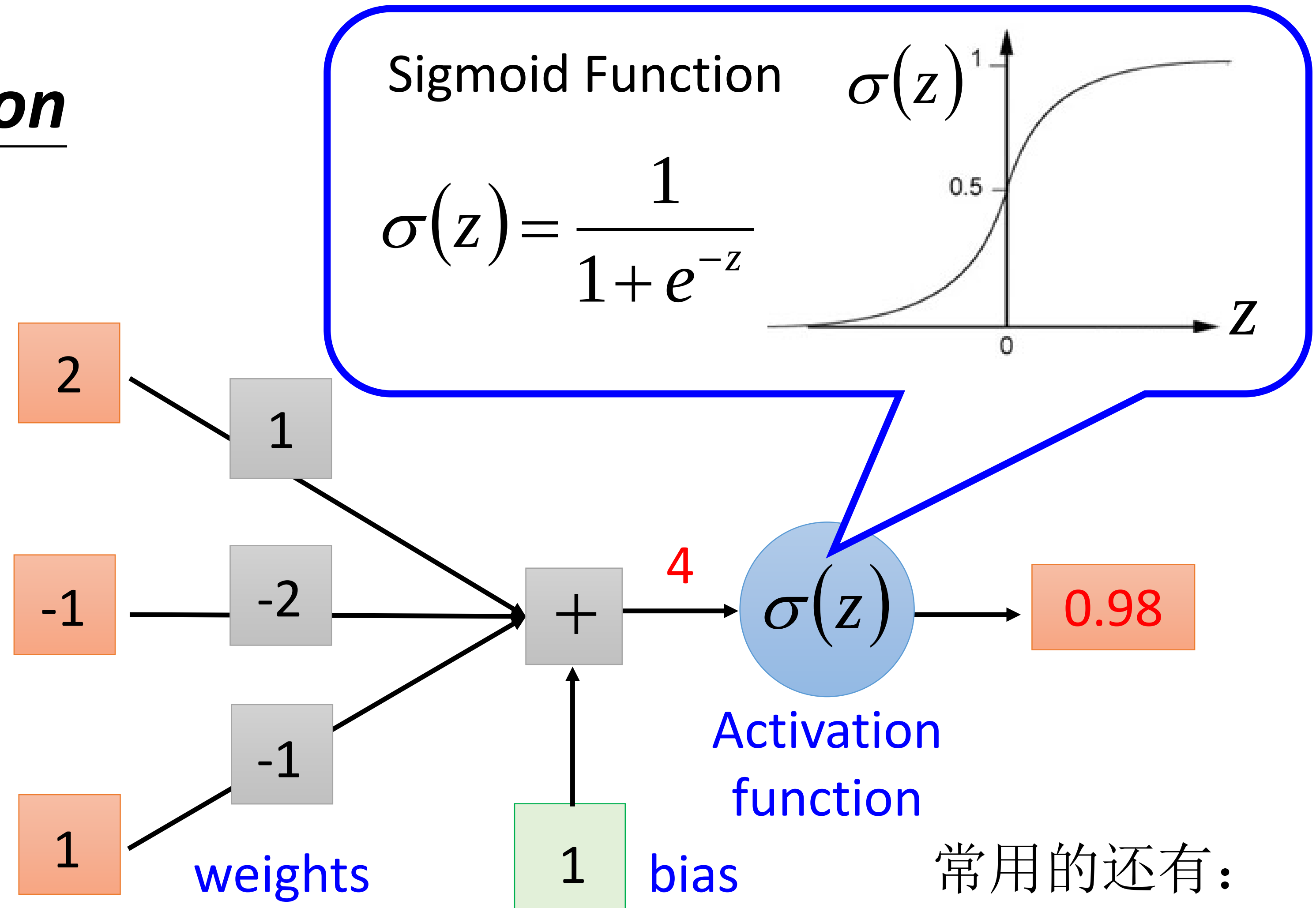
Neuron

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



Neural Network

Neuron



常用的还有：
Relu, Tanh

神经元与Logistic Regression单元

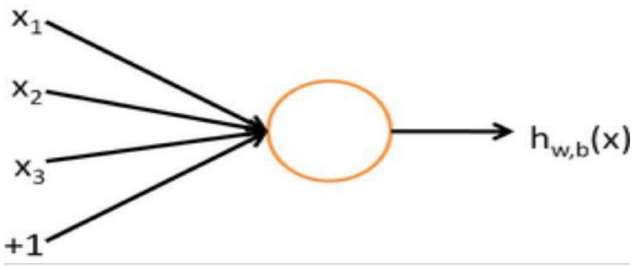
- 线性回归（Linear Regression）
  - $y = \sum_i w_i x_i + b$
- Logistic回归（Logistic Regression）
  - Logistic函数: (-∞, + ∞)

$$y = \frac{L}{1 + \mathrm{e}^{-k(z - z_0)}}$$

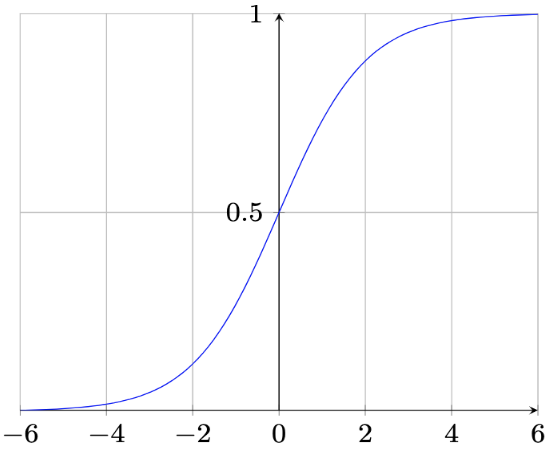
  - 设$z = \sum_i w_i x_i + b$
  - Sigmoid函数（Logistic函数的特例）

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

  - 处理二元分类问题，y为输出的概率
    - 垃圾邮件过滤、褒贬识别

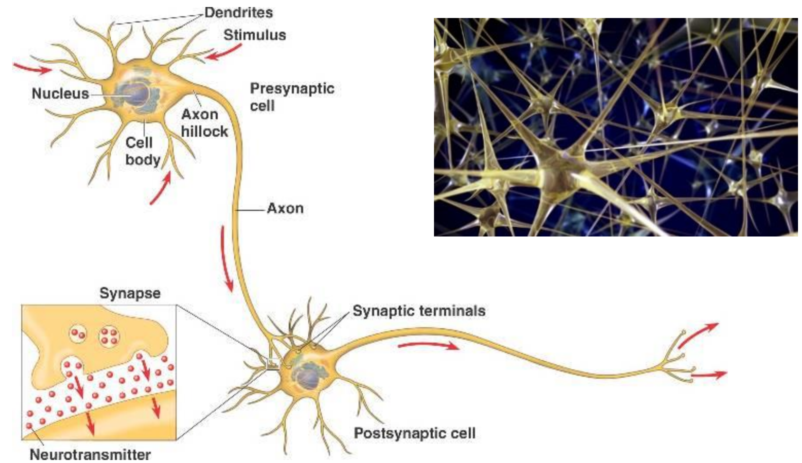


w, b are the parameters of this neuron
i.e., this logistic regression model

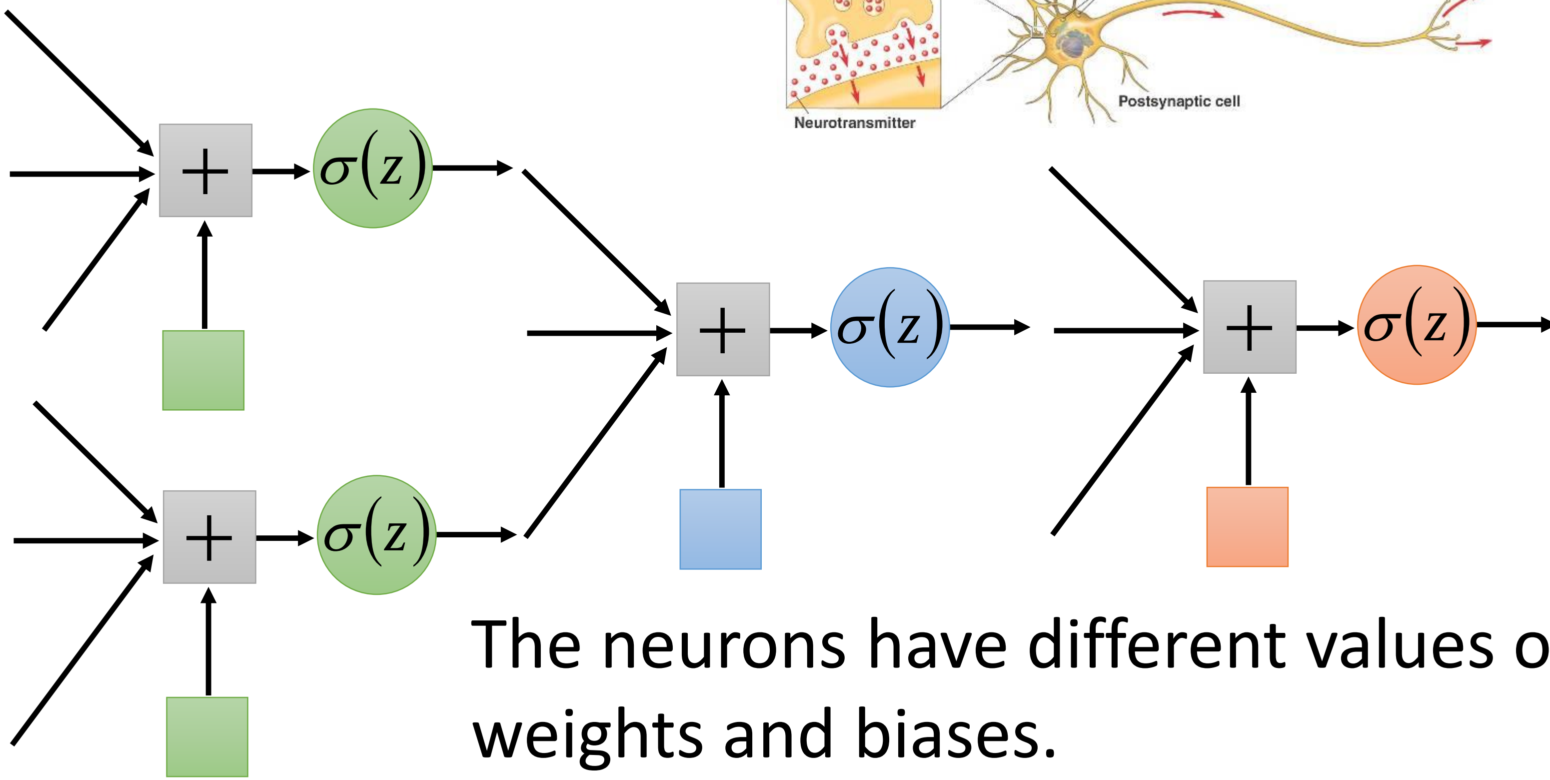


Sigmoid函数示意图

Neural Network



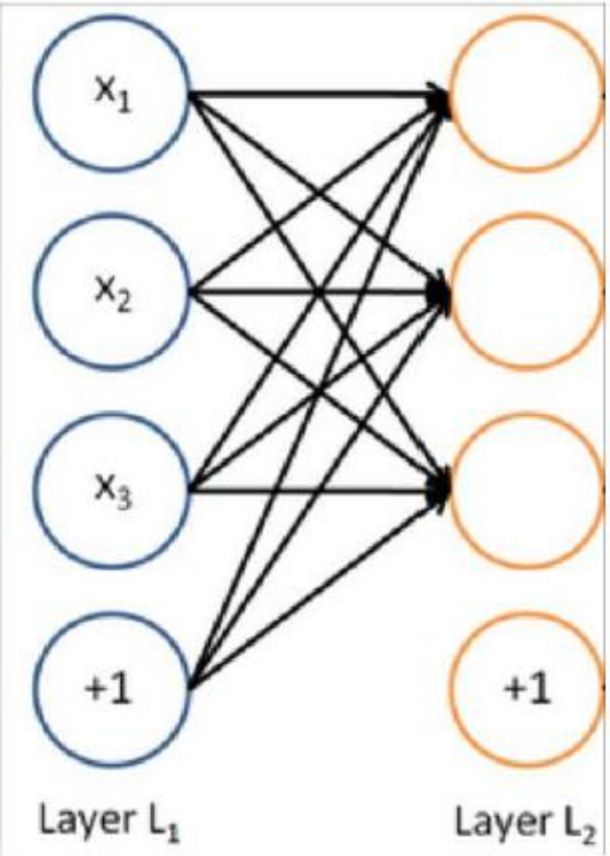
Different connections lead to different network structures



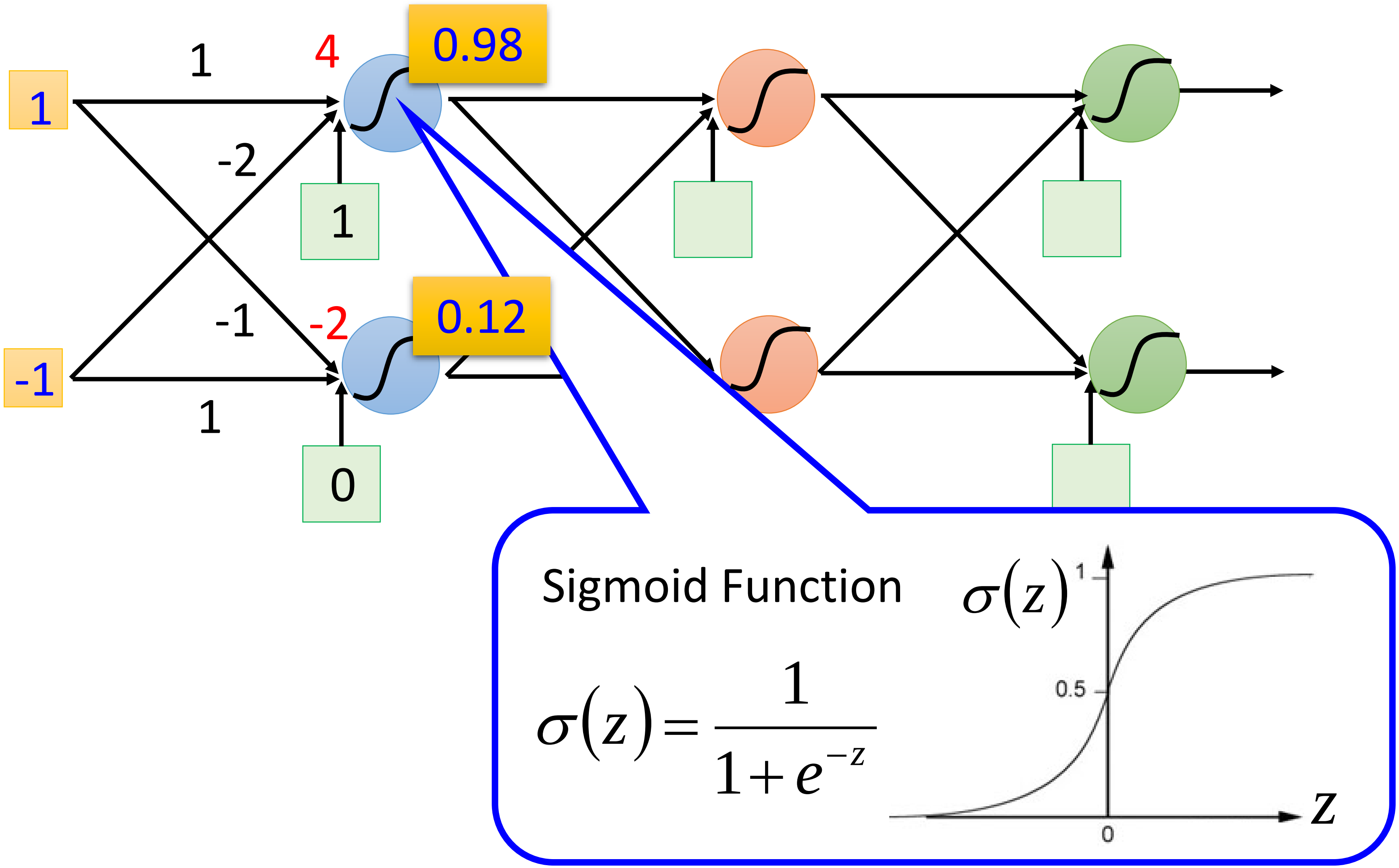
The neurons have different values of weights and biases.

Weights and biases are network parameters θ

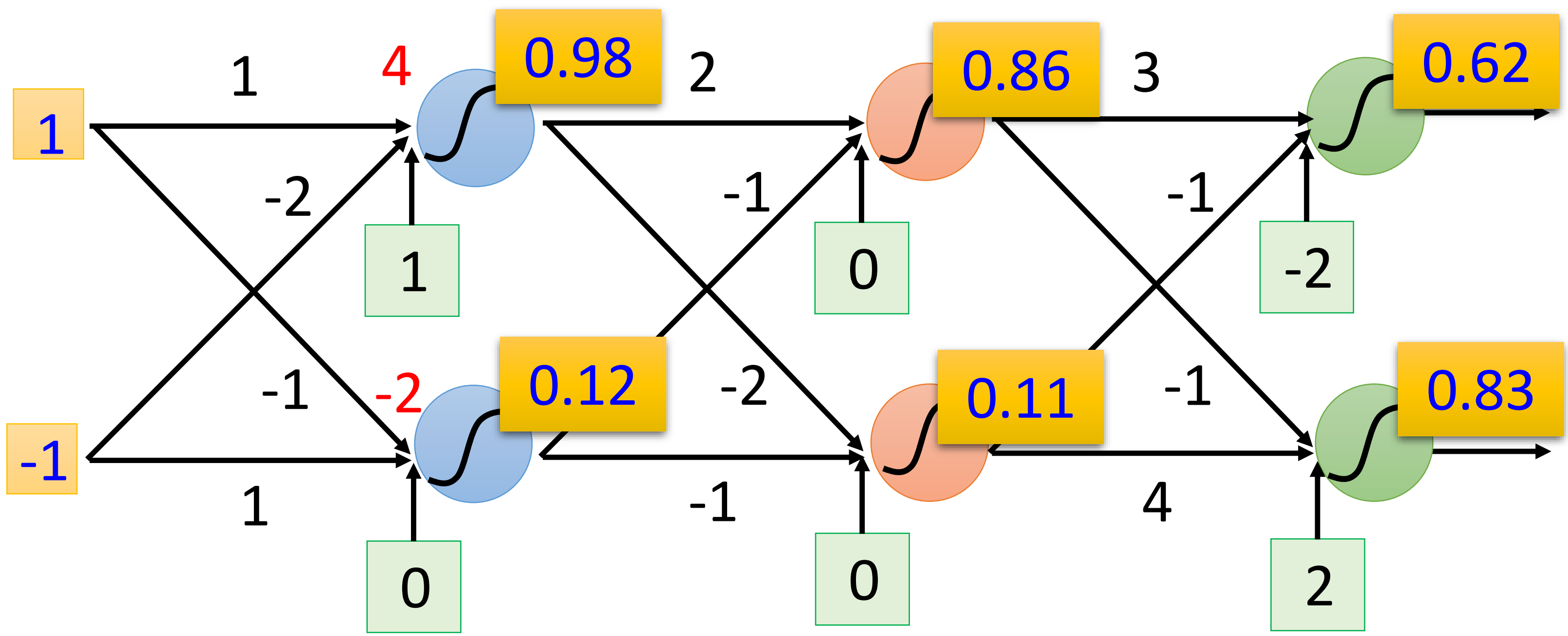
一个神经网络 i.e. 多个logistic regression单元同时运行



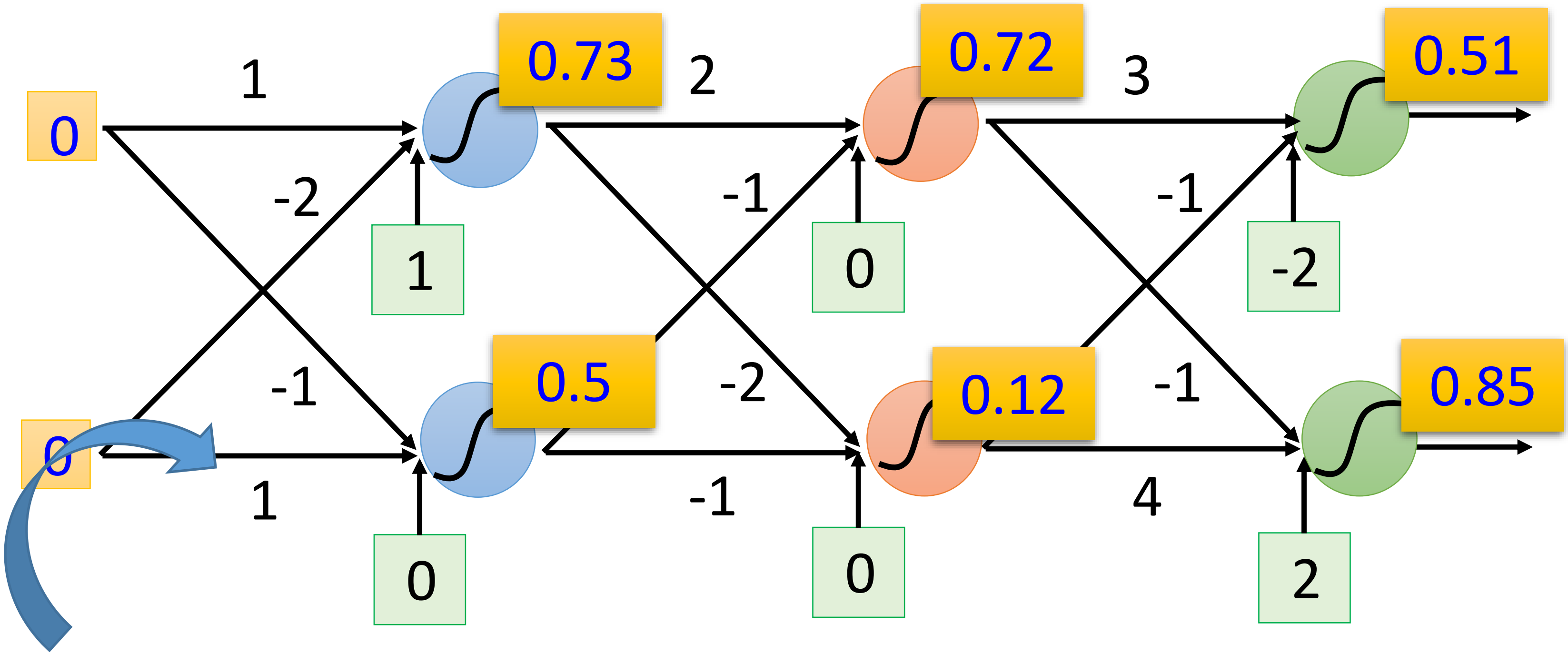
Fully Connect Feedforward Network



Fully Connect Feedforward Network



Fully Connect Feedforward Network



This is a function.

Input vector, output vector

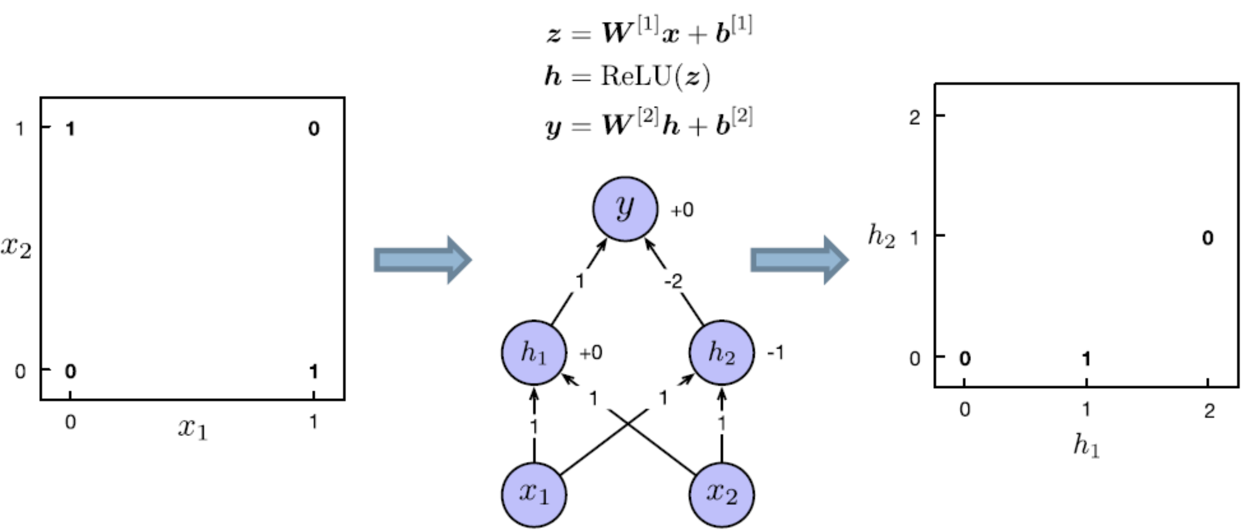
$$f\left(\begin{bmatrix}1\\-1\end{bmatrix}\right)=\begin{bmatrix}0.62\\0.83\end{bmatrix} \quad f\left(\begin{bmatrix}0\\0\end{bmatrix}\right)=\begin{bmatrix}0.51\\0.85\end{bmatrix}$$

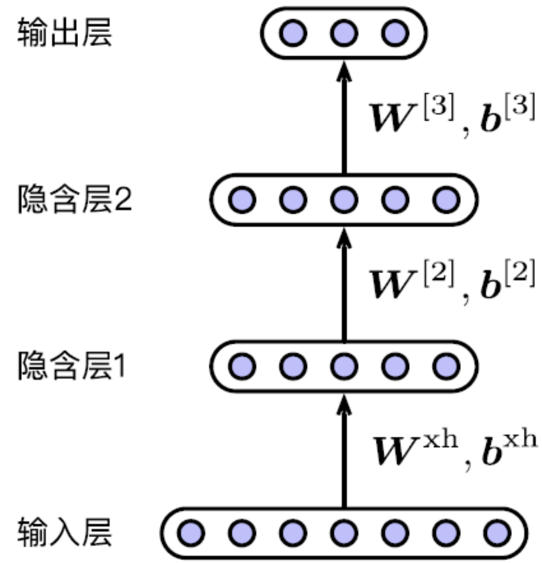
Given parameters θ, define a function

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

神经网络特长1

- 可解决**线性不可分**问题
  - 如：异或（XOR）问题





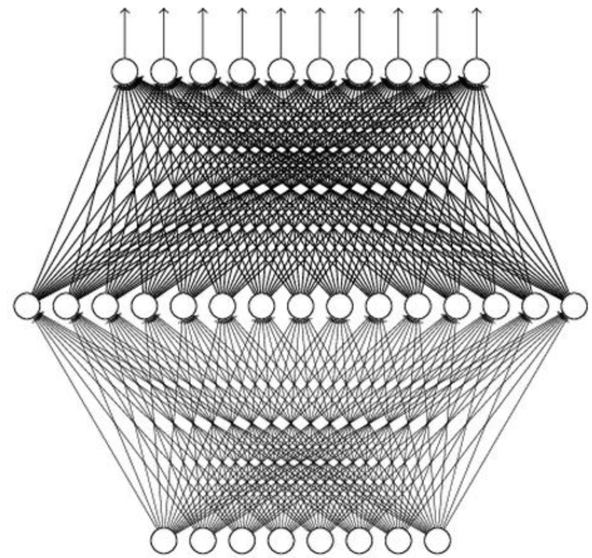
神经网络特长2： Universality Theorem

Any continuous function f

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

Can be realized by a network with one hidden layer

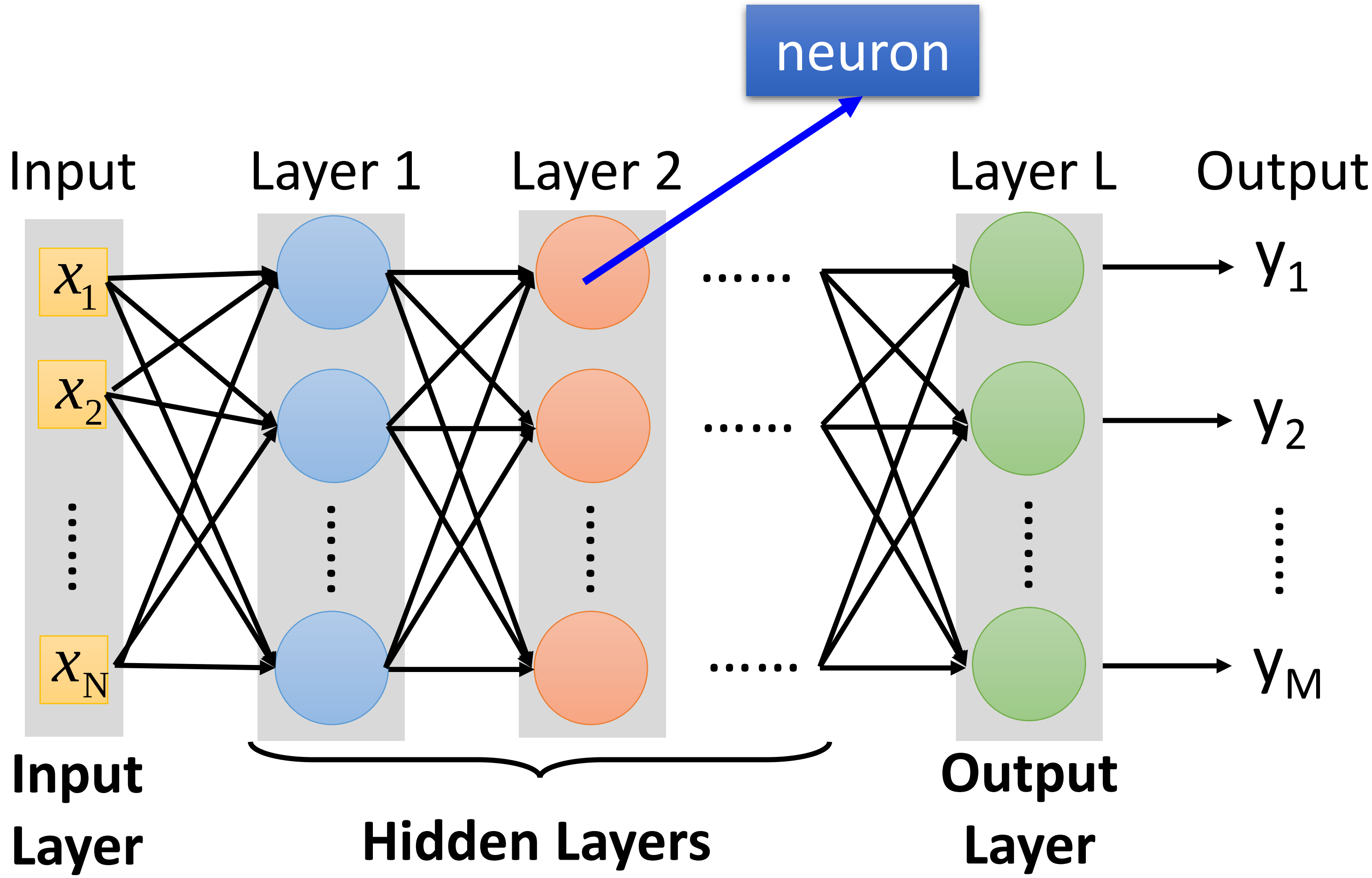
(given **enough** hidden neurons)



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

通用近似定理：即使是二层神经网络，都可以以任意精度近似任意一个连续函数。

Deep means many hidden layers



Why Deep? Analogy

Logic circuits

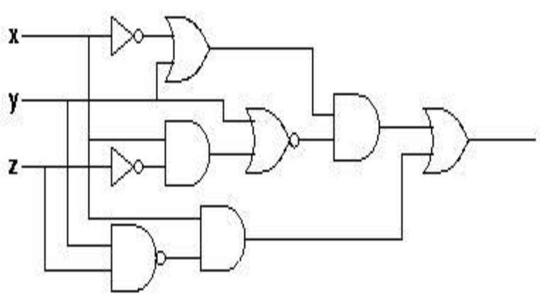
- Logic circuits consists of **gates**
- **A two layers of logic gates** can represent **any Boolean function.**
- Using multiple layers of logic gates to build some functions are much simpler

→ less gates needed

Neural network

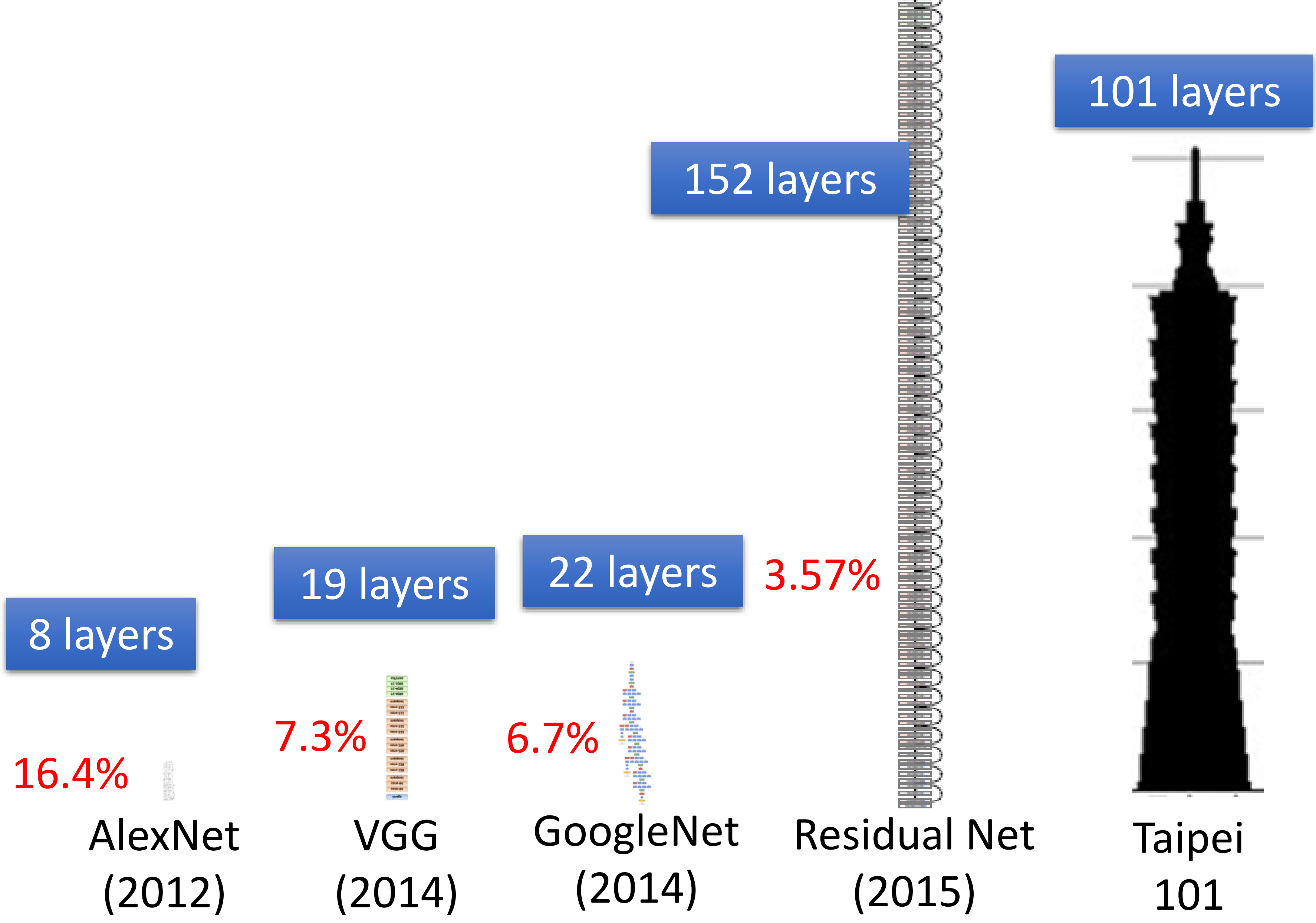
- Neural network consists of **neurons**
- **A hidden layer network** can represent **any continuous function.**
- Using multiple layers of neurons to represent some functions are much simpler

→ less parameters → less data?



定理 2. 存在这样的布尔函数，可以由深度为 k 的多项式复杂度的逻辑门电路表示，等价的深度为 k-1 的逻辑门电路变为指数复杂度。这里深度指从输入到输出的最长路径的长度，复杂度指逻辑门的个数。

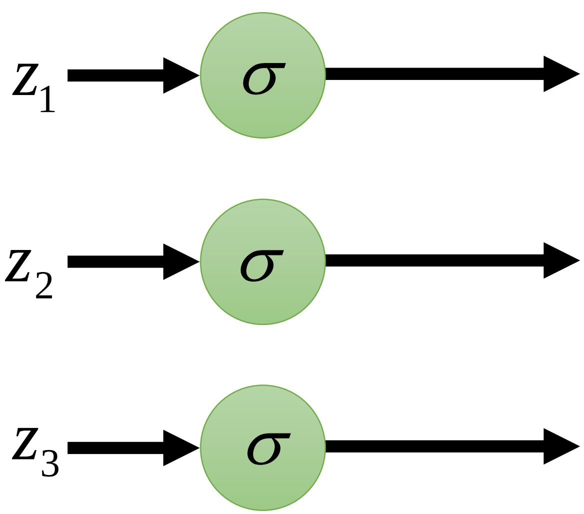
Deep = Many hidden layers



Output Layer

- 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

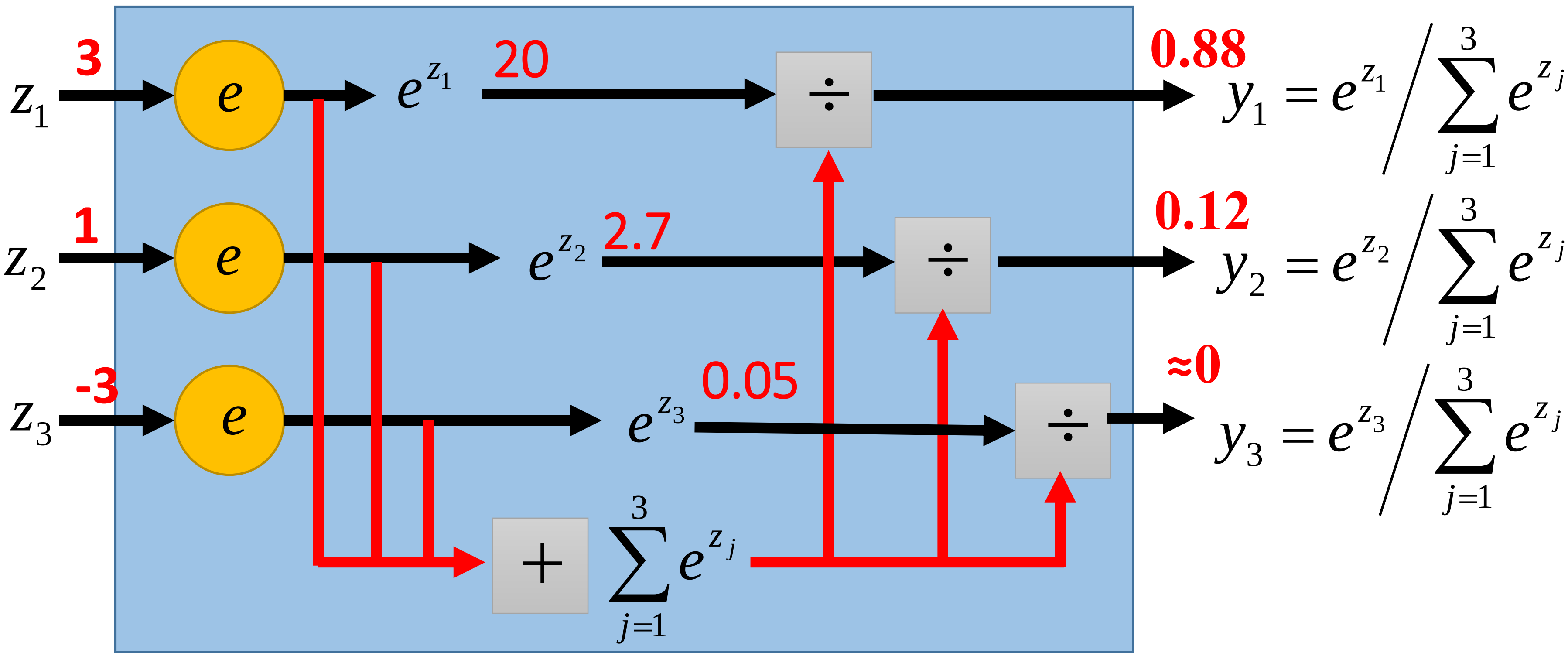
Output Layer

- Softmax layer as the output layer

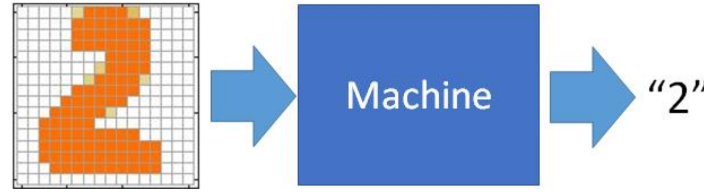
Probability:

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

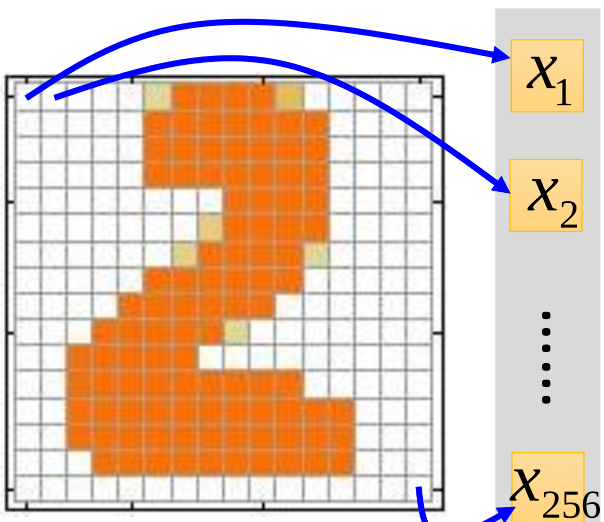
Softmax Layer



Example Application



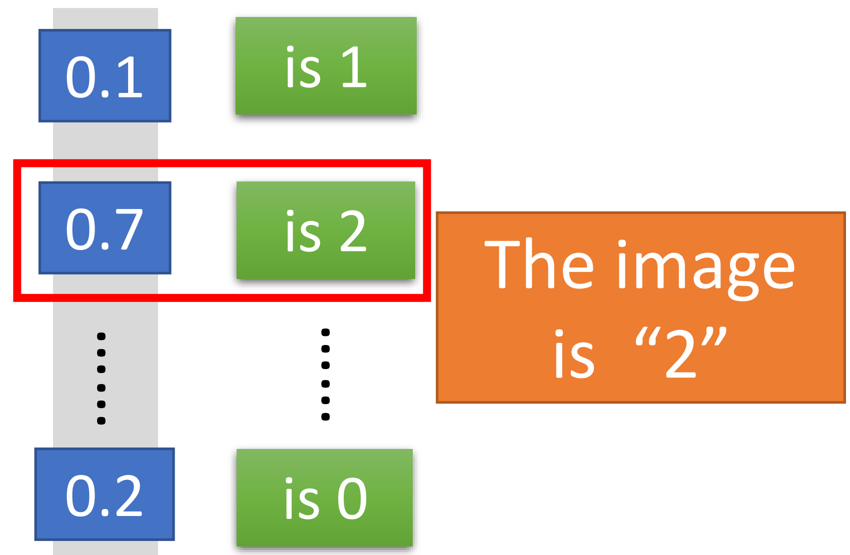
Input



16 x 16 = 256

with Ink → 1
w/o ink → 0

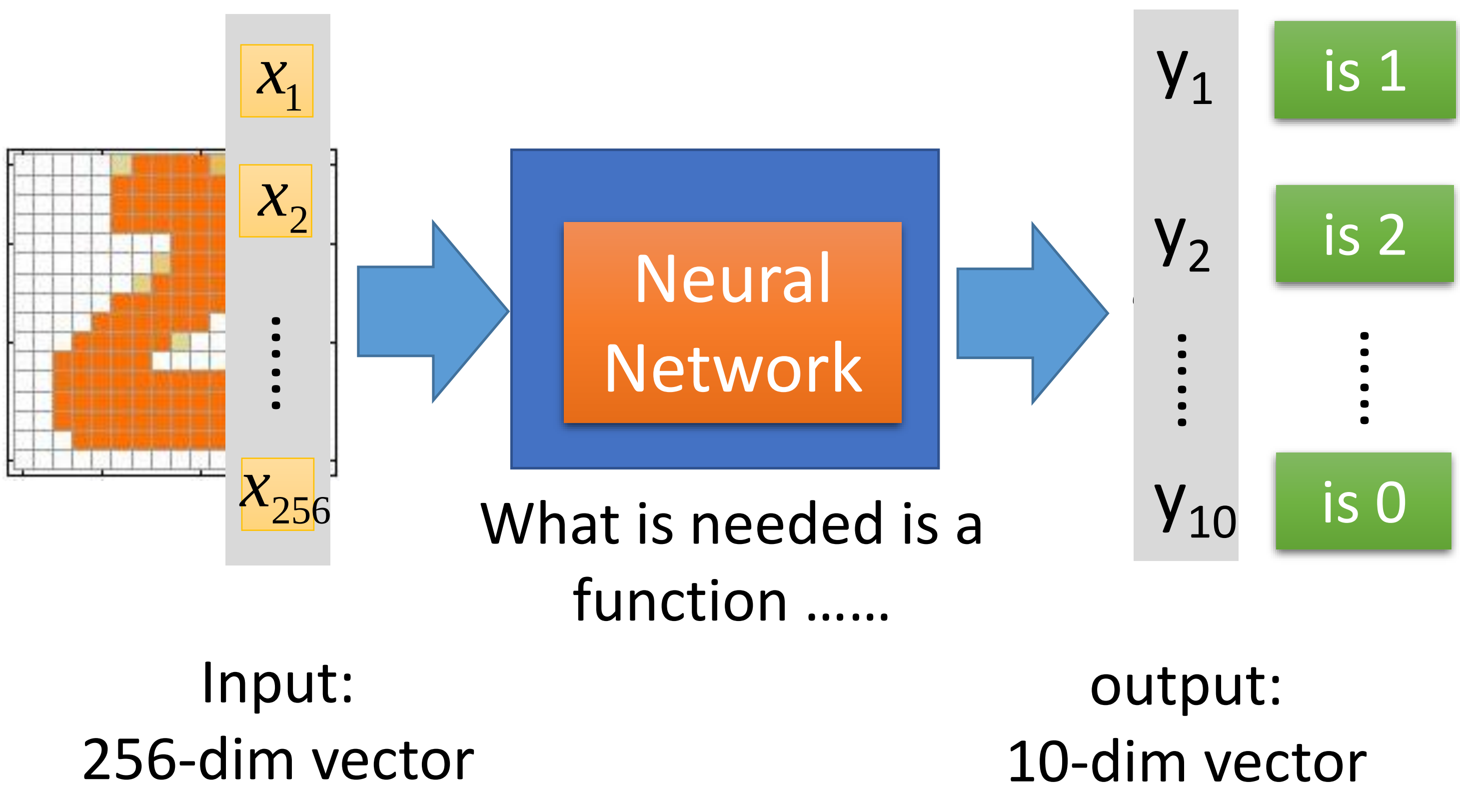
Output



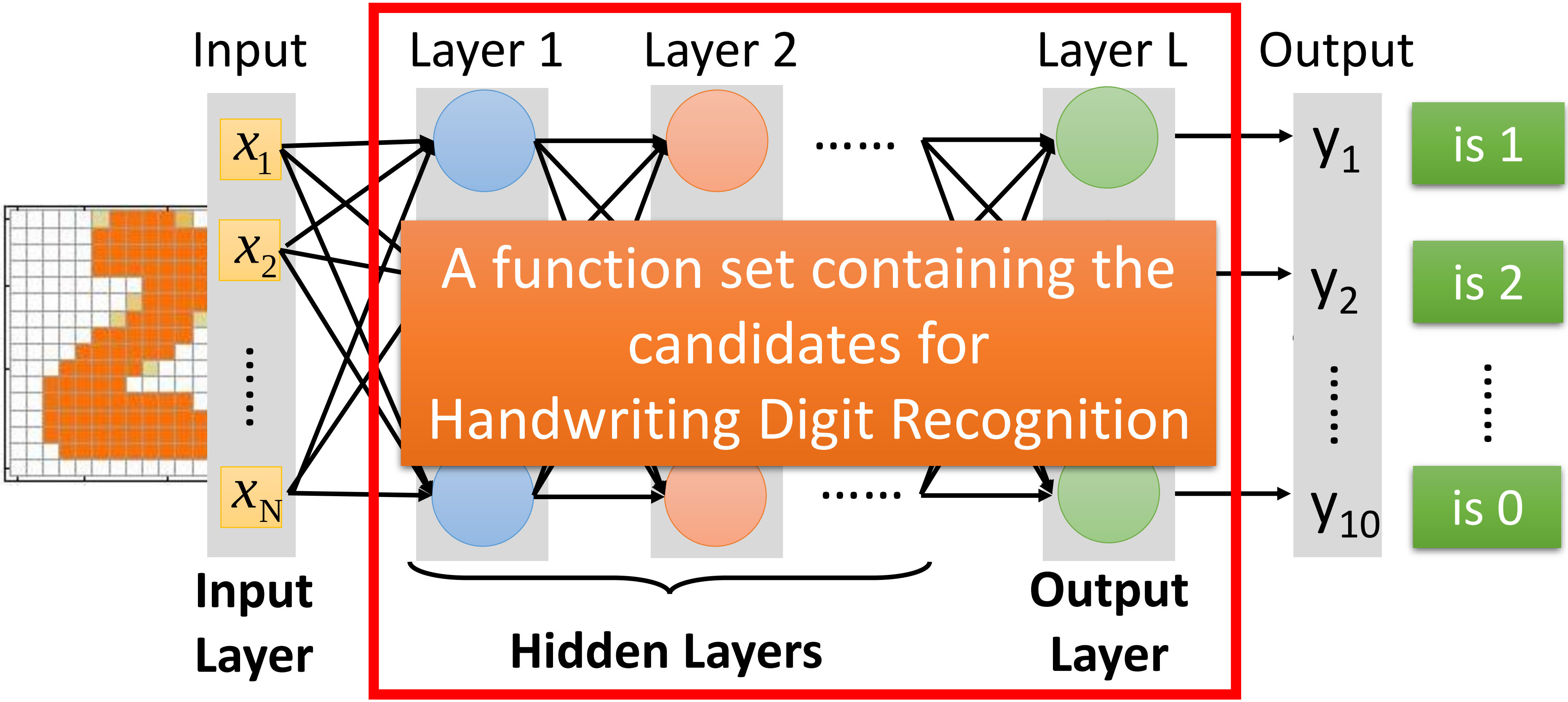
Each dimension represents the confidence of a digit.

Example Application

- Handwriting Digit Recognition



Example Application



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

小结：损失函数

- 损失函数（Loss Function）
  - 评估参数好坏的准则
  - 为什么不直接使用准确率等指标进行评估？
- 两种常用的损失函数
  - 均方误差（Mean Squared Error，MSE）

$$\mathrm{MSE} = \frac{1}{m}\sum_{i=1}^{m}(\hat{y}^{(i)} - y^{(i)})^2$$

  - 交叉熵（Cross-Entropy，CE）

$$\mathrm{CE} = -\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{c} y_j^{(i)} \log \hat{y}_j^{(i)} \Rightarrow \mathrm{CE} = -\frac{1}{m}\sum_{i=1}^{m} \log \hat{y}_t^{(i)}$$

负对数似然损失
Negative Log
Likelihood，NLL

Three Steps for Deep Learning

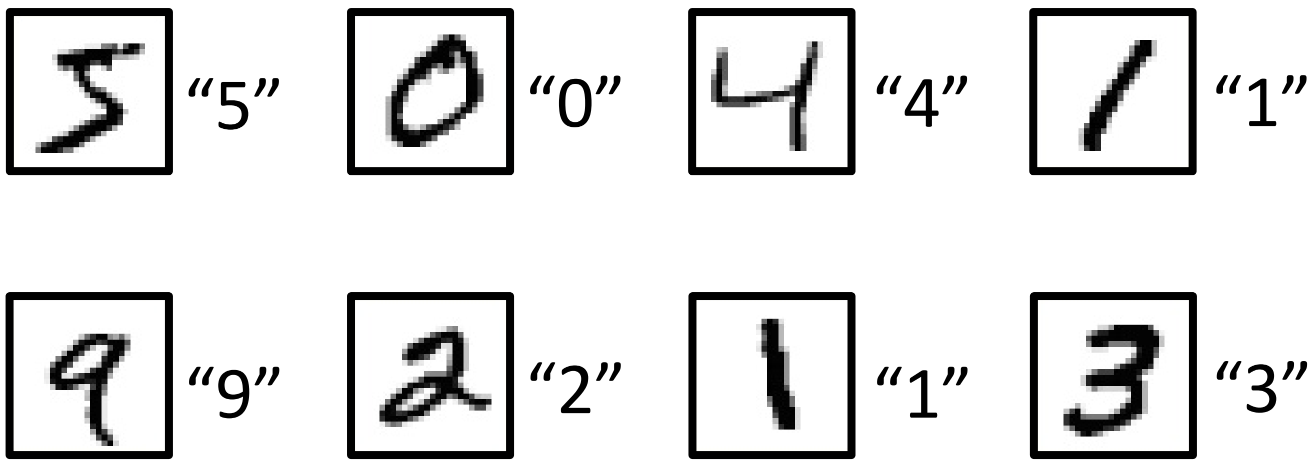
Step 1: define a set of function

Step 2: goodness of function

Step 3: pick the best function

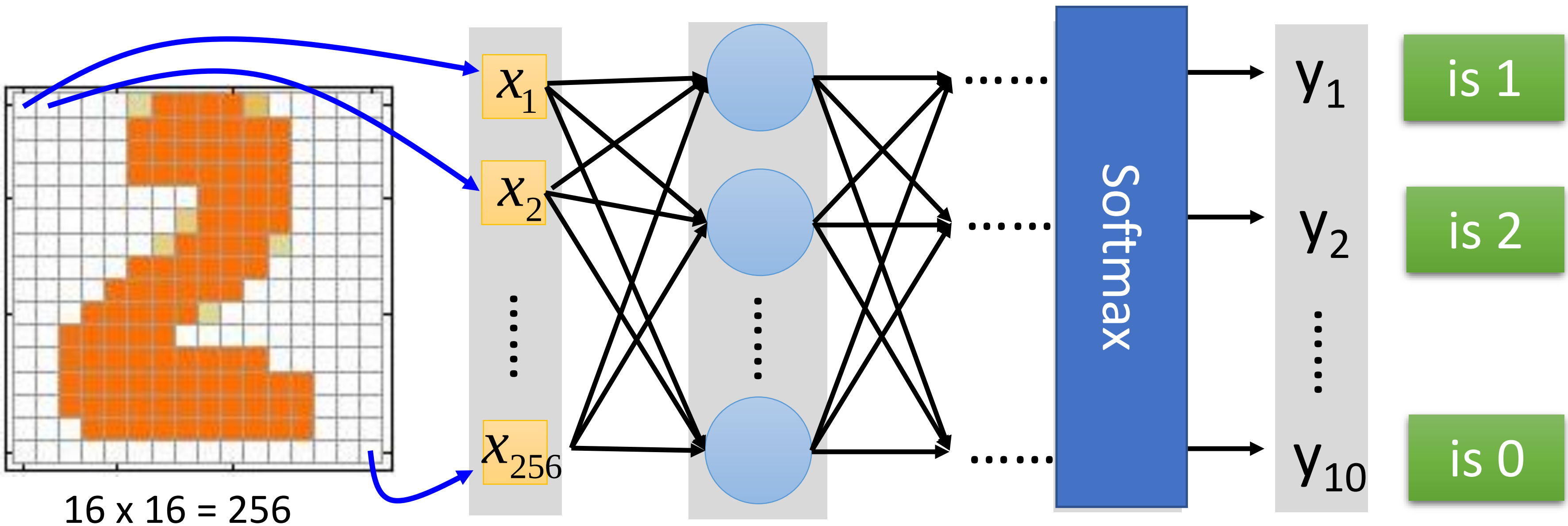
Training Data

- Preparing training data: images and their labels



The learning target is defined on the training data.

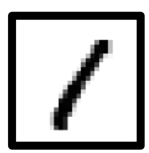
Learning Target



Ink → 1

No ink → 0

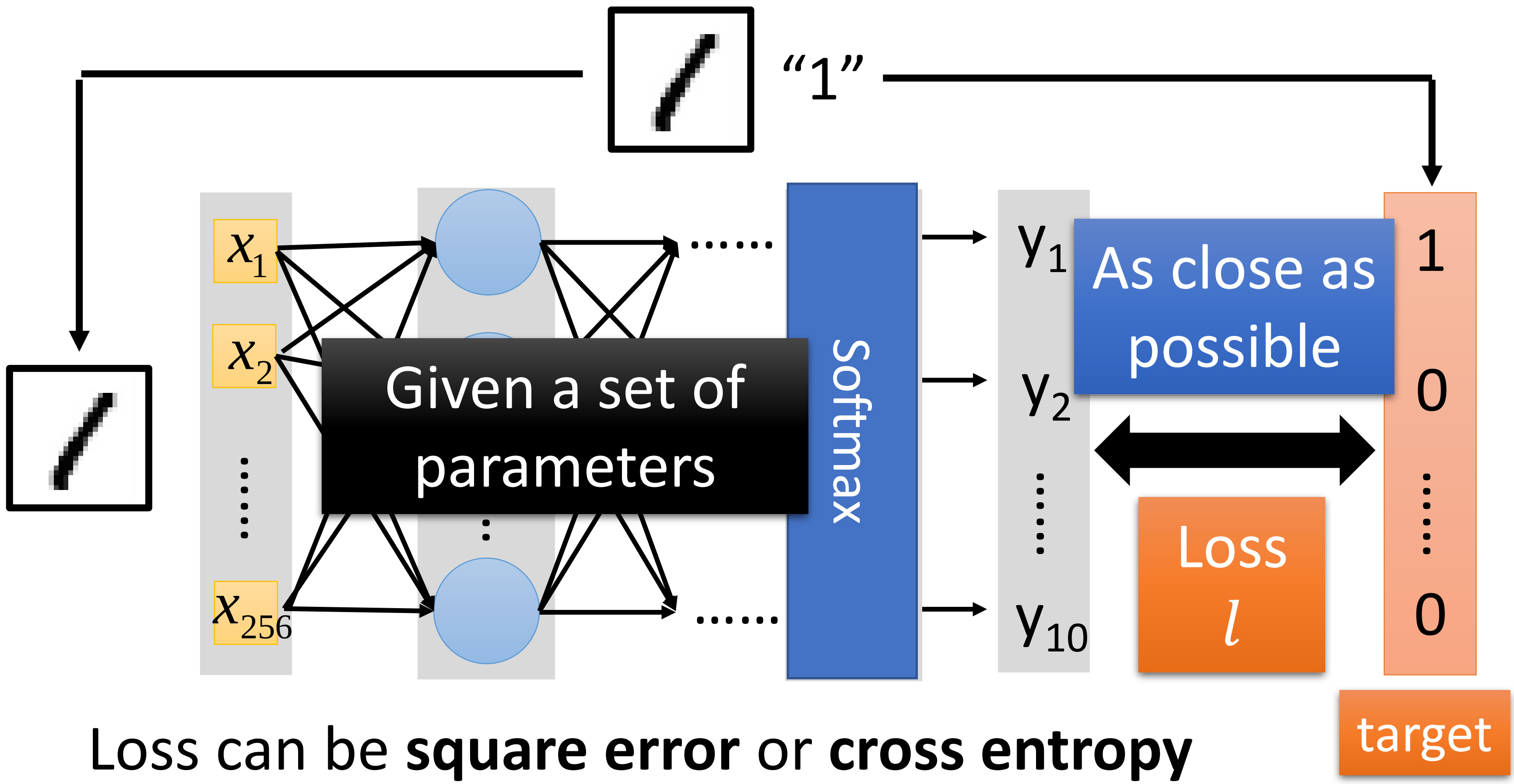
The learning target is ……

Input:  y_1 has the maximum value

Input:  y_2 has the maximum value

Loss

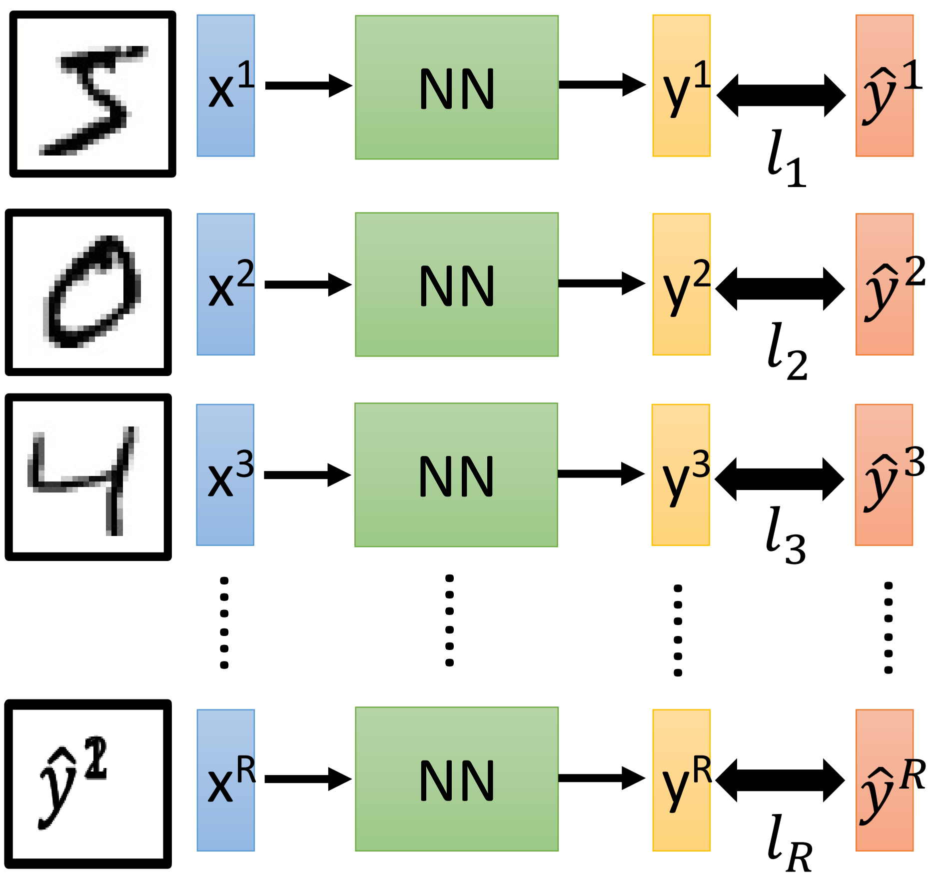
A good function should make the loss of all examples as small as possible.



Loss can be **square error** or **cross entropy** 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

Step 1: define a set of function

Step 2: goodness of function

Step 3: pick the best function

How to pick the best function

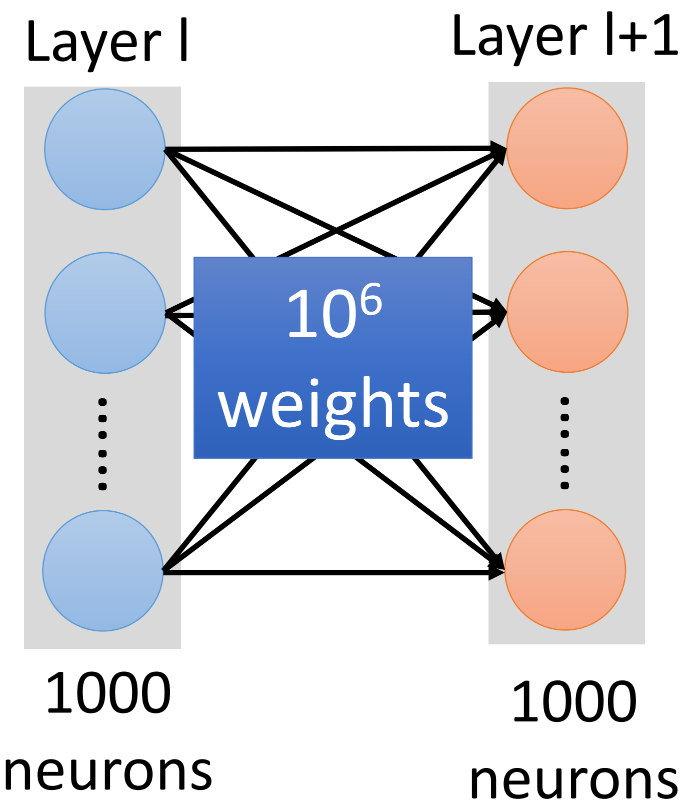
Find ***network parameters θ^**** that minimize total loss L

Enumerate all possible values

Network parameters $\theta =$

$$\underbrace{\{w_1, w_2, w_3, \cdots, b_1, b_2, b_3, \cdots\}}_{\text{Millions of parameters}}$$

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



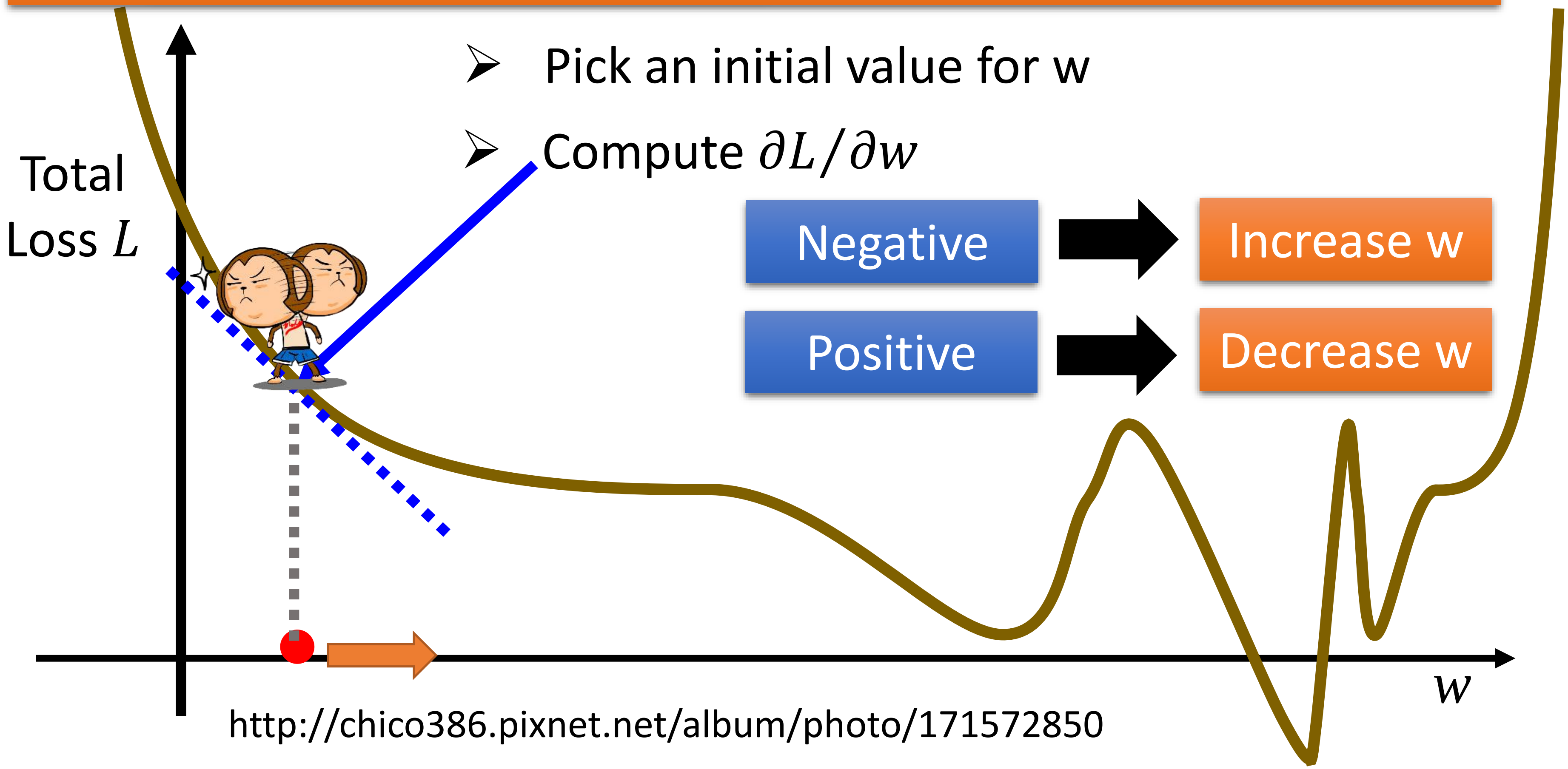
Gradient Descent

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

Find *network parameters θ^** 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

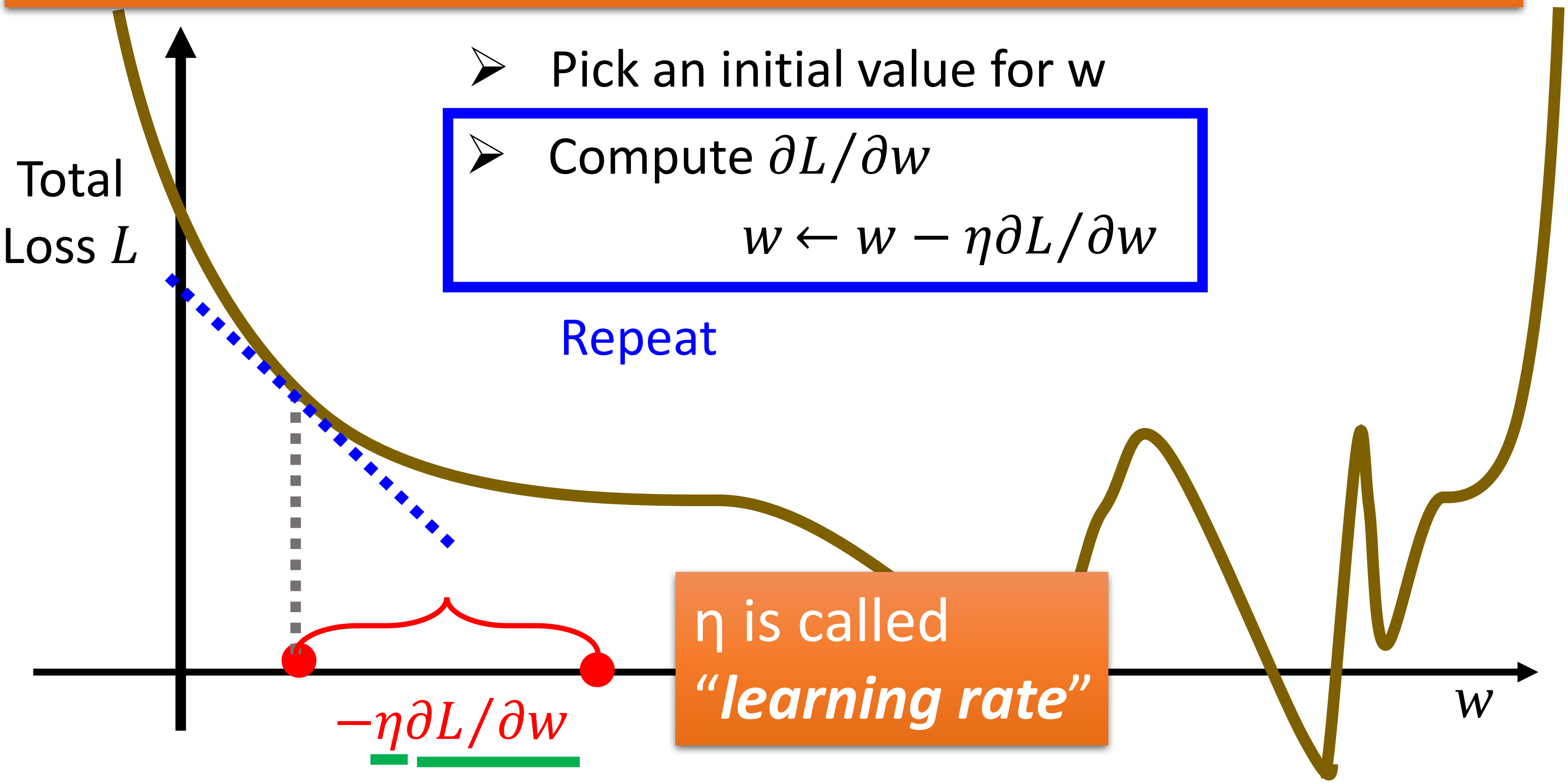
Network parameters $\theta = \{w_1, w_2, \cdots, b_1, b_2, \cdots\}$

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

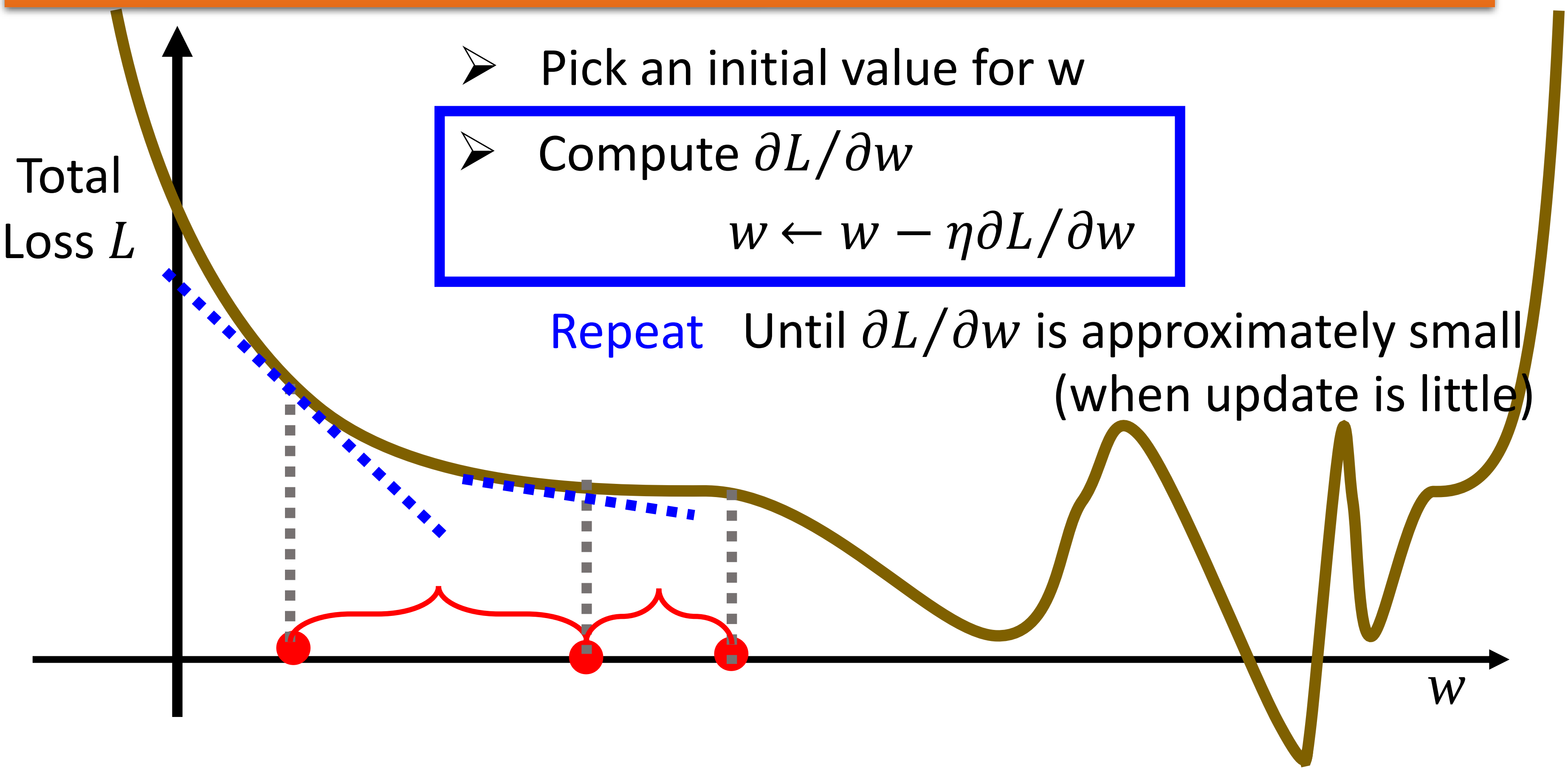
Network parameters $\theta = \{w_1, w_2, \cdots, b_1, b_2, \cdots\}$

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

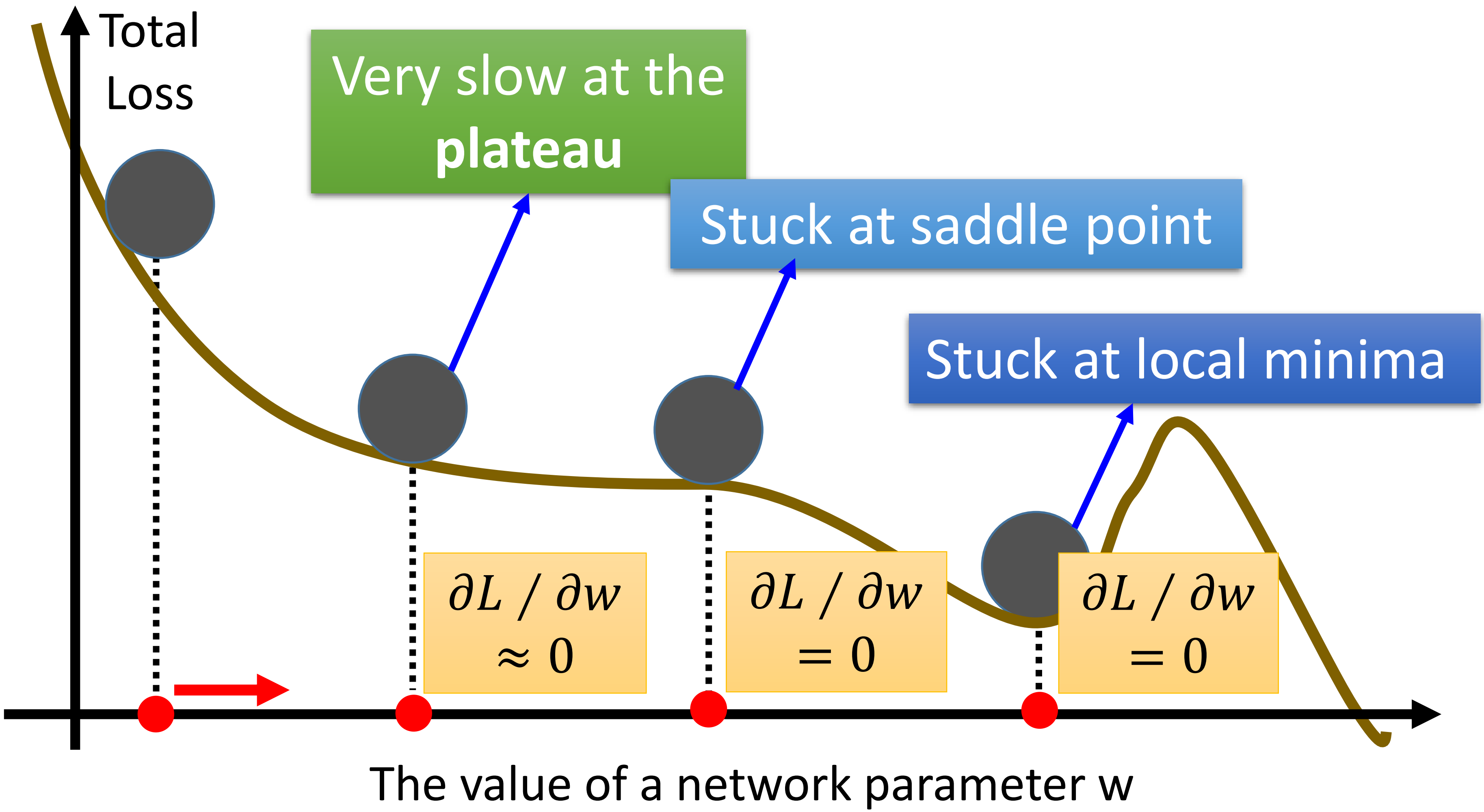
- Pick an initial value for w
- Compute $\partial L / \partial w$

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

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

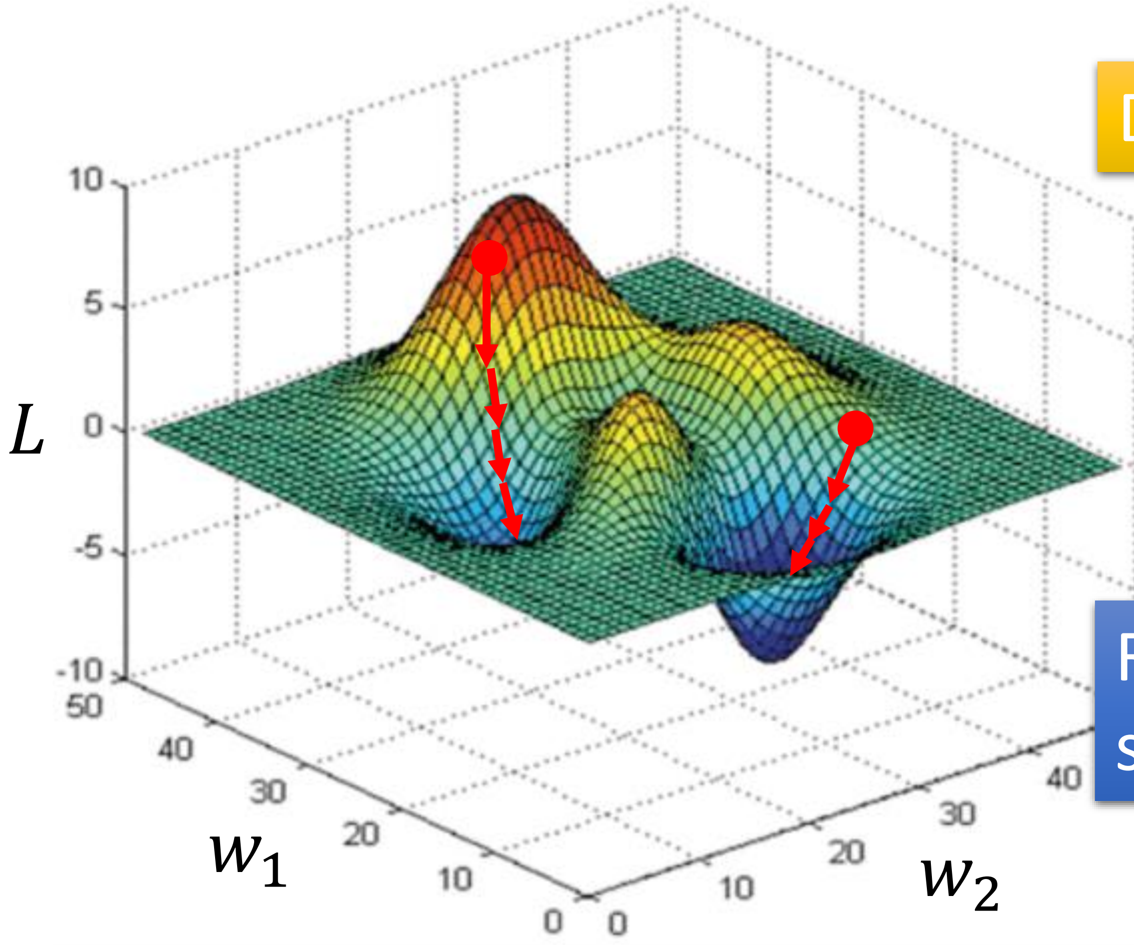


Local Minima



Local Minima

- Gradient descent never guarantee global minima



Different initial point

Reach different minima, so different results

小结：梯度下降

- 梯度（Gradient）
  - 以向量的形式写出的对多元函数各个参数求得的偏导数
  - 是函数值增加最快的方向
  - 沿着梯度相反的方向，更加容易找到函数的极小值
- 梯度下降算法（Gradient Descent，GD）

算法 4.1 梯度下降算法

Input: 学习率 α；含有 m 个样本的训练数据

Output: 优化参数 $\boldsymbol{\theta}$

设置损失函数为 $L(f(\boldsymbol{x};\boldsymbol{\theta}),y)$;

初始化参数 $\boldsymbol{\theta}$。

while 未达到终止条件 **do**

  计算梯度 $\boldsymbol{g}=\frac{1}{m}\nabla_{\boldsymbol{\theta}}\sum_i^m L(f(\boldsymbol{x}^{(i)};\boldsymbol{\theta}),y^{(i)})$;

  $\boldsymbol{\theta}=\boldsymbol{\theta}-\alpha\boldsymbol{g}$。

end

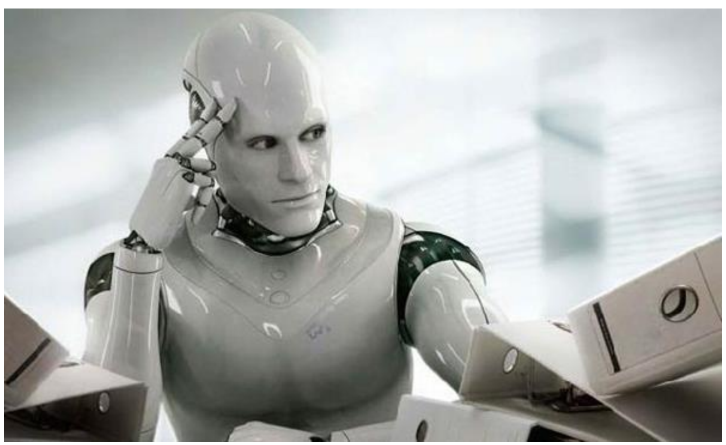
- 小批次梯度下降法（Mini-batch Gradient Descent）
  - 每次随机采样小规模的训练数据来估计梯度
  - 提高算法的运行速度

Gradient Descent

This is the “learning” of machines in deep learning ……

➡ Even alpha go using this approach.

People imagine ……



Actually …..

I hope you are not too disappointed :p

Three Steps for Deep Learning

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

Deep Learning is so simple ……

Now If you want to find a function

If you have lots of function input/output (?) as training data

→ You can use deep learning

Next

Network structures!