

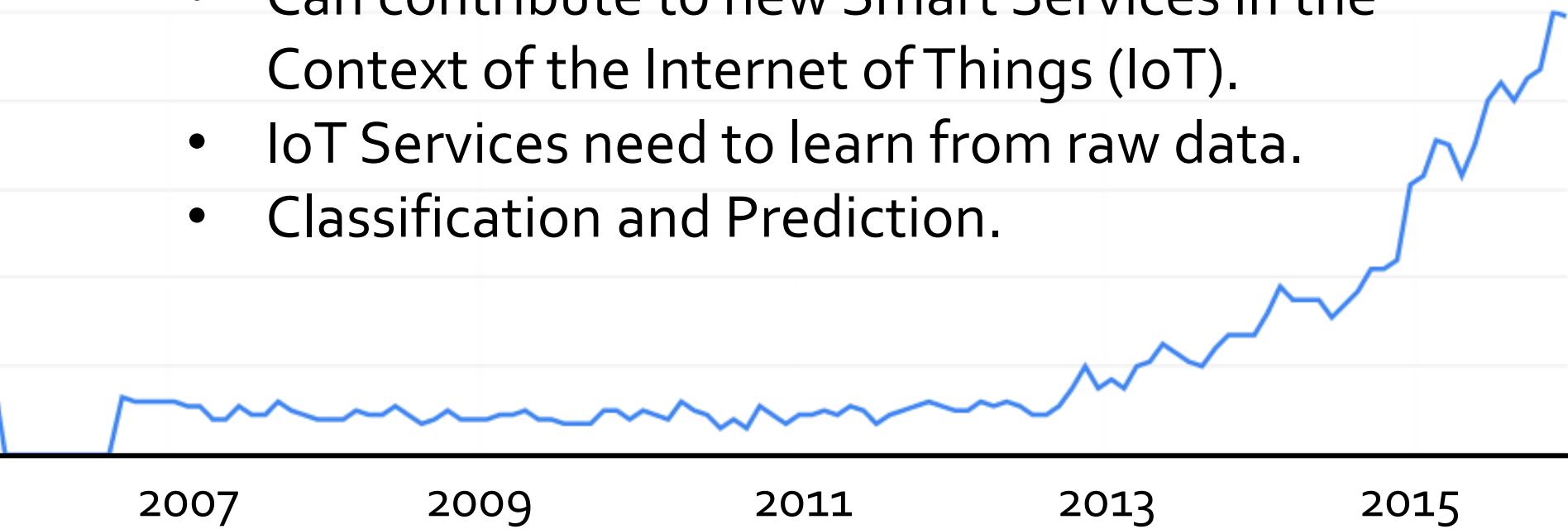
Deep Learning

Based on the original slides of Hung-yi Lee

Deep learning attracts lots of attention.

■ Google Trends

- Deep learning obtains many exciting results.
- Can contribute to new Smart Services in the Context of the Internet of Things (IoT).
- IoT Services need to learn from raw data.
- Classification and Prediction.



Outline

Part I: Introduction of Deep Learning



Part II: Why Deep?



Part III: Tips for Training Deep Neural Network



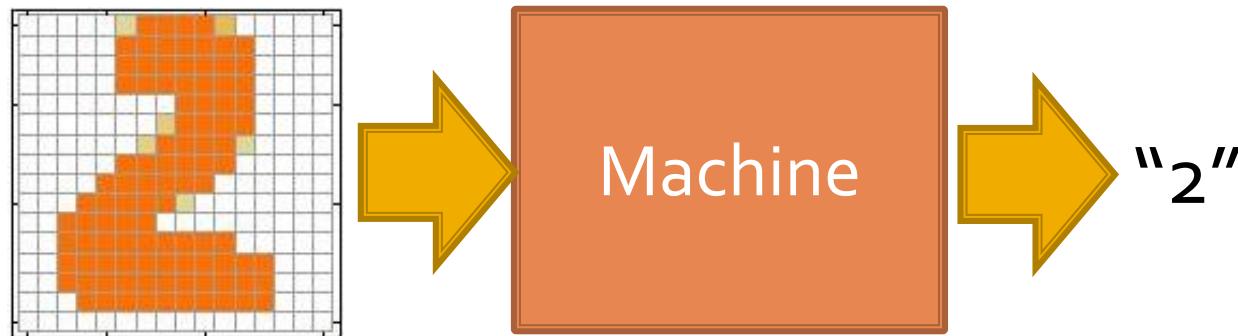
Part IV: Neural Network with Memory

Part I: Introduction of Deep Learning

What people already knew in 1980s

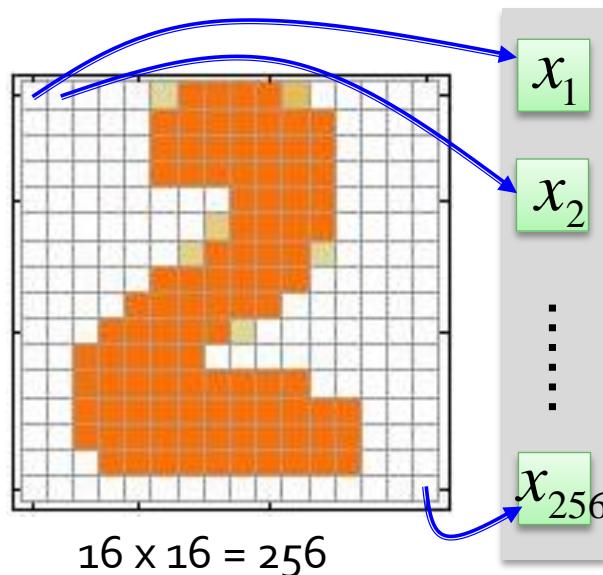
Example Application

- Handwriting Digit Recognition



Handwriting Digit Recognition

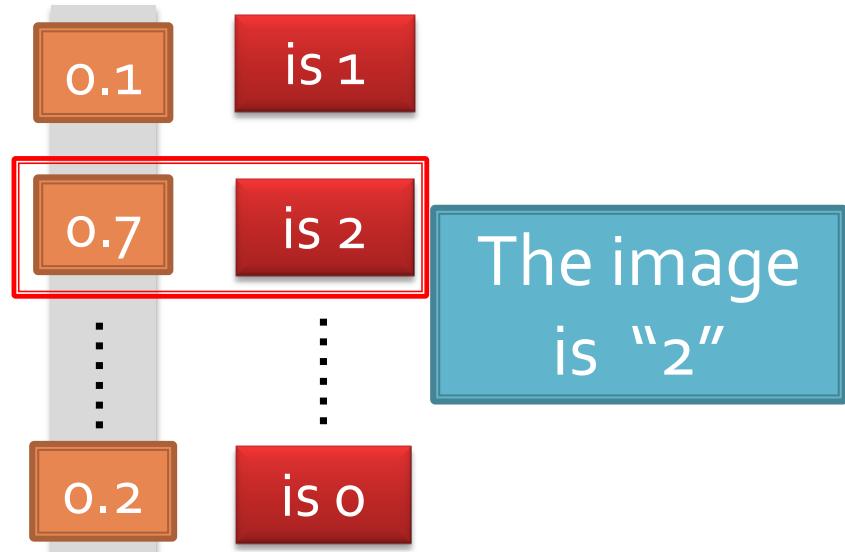
INPUT



Ink $\rightarrow 1$

No ink $\rightarrow 0$

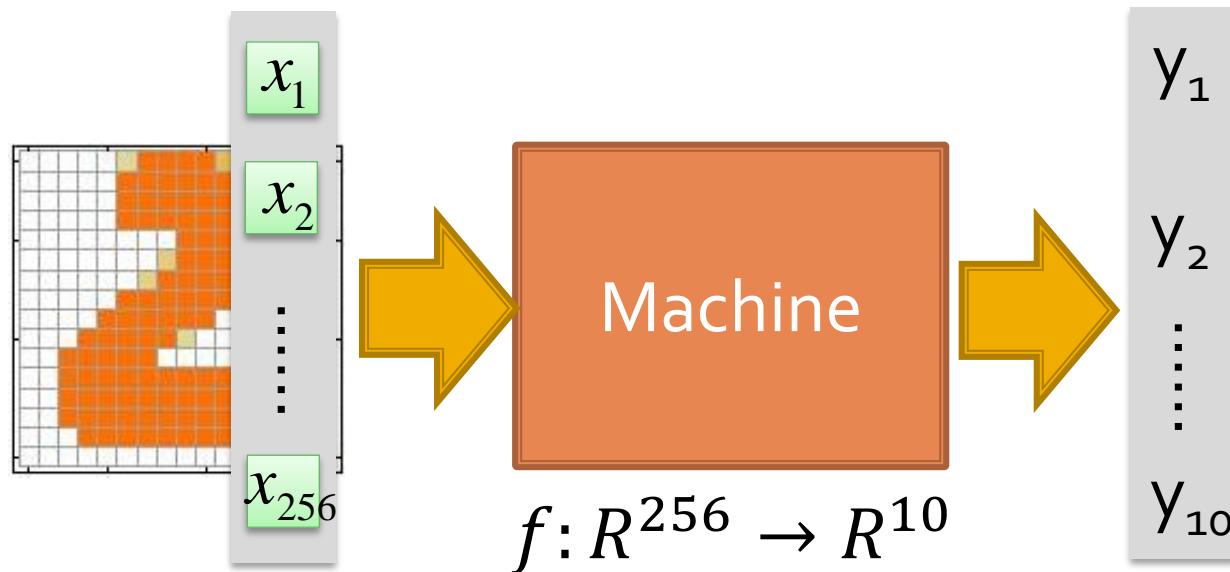
OUTPUT



Each dimension represents the confidence of a digit.

Example Application

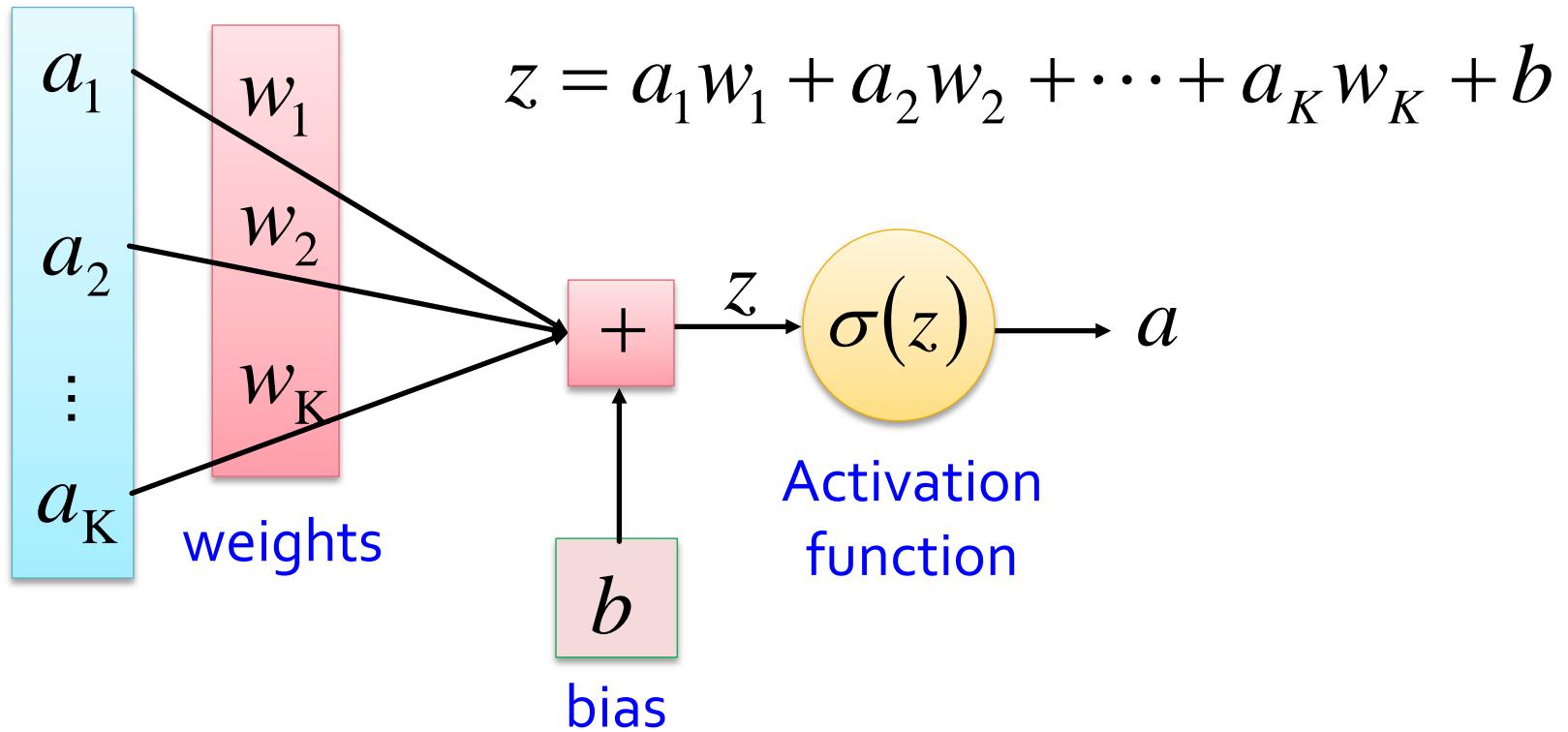
■ Handwriting Digit Recognition



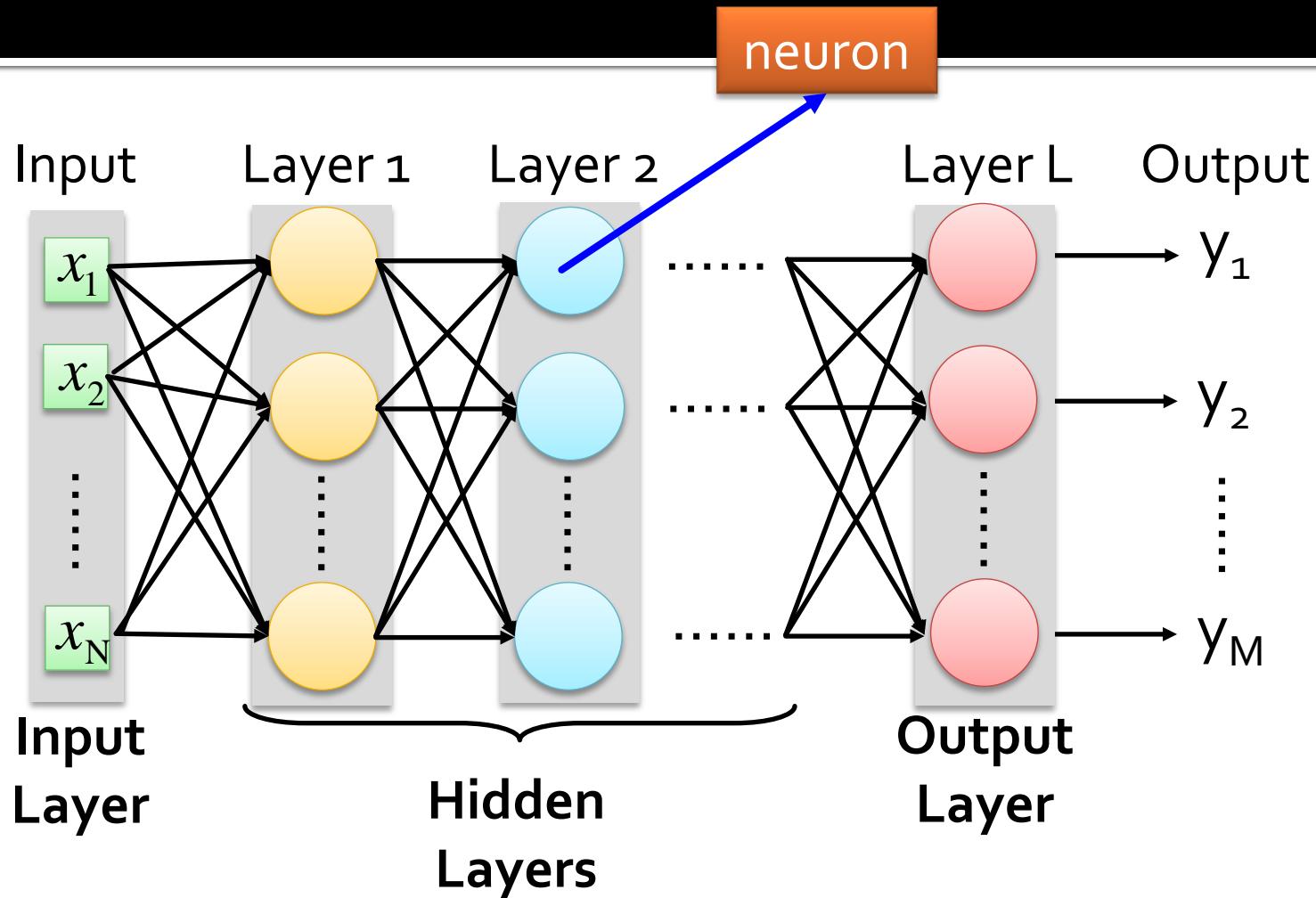
In deep learning, the function f is represented by neural network

Element of Neural Network

Neuron $f: R^K \rightarrow R$

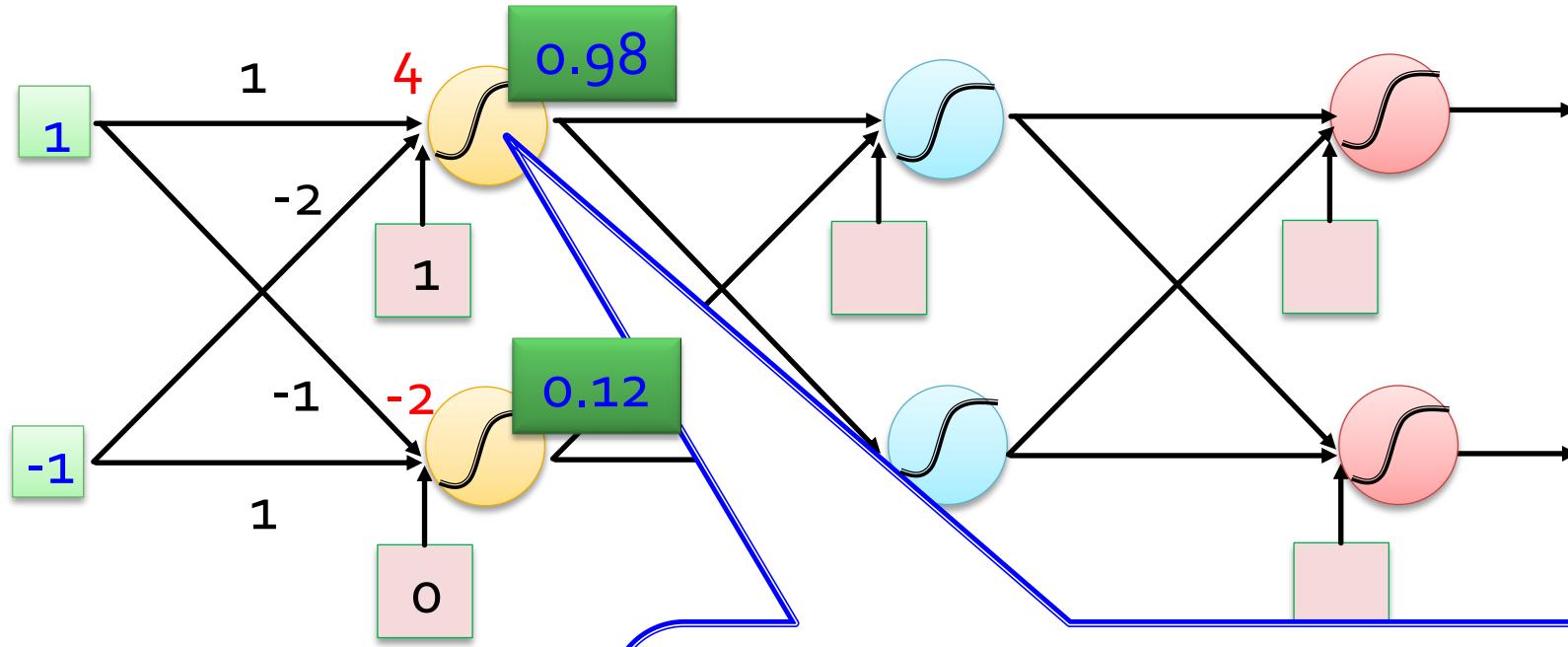


Neural Network



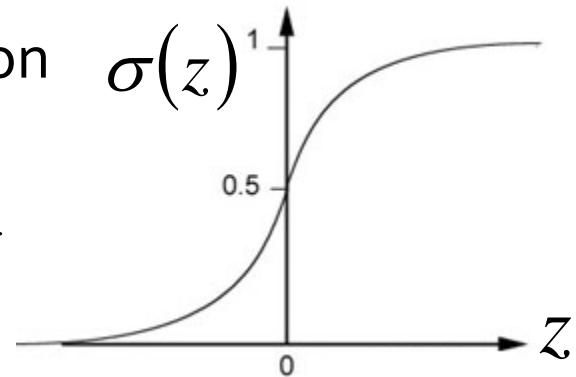
Deep means many hidden layers

Example of Neural Network

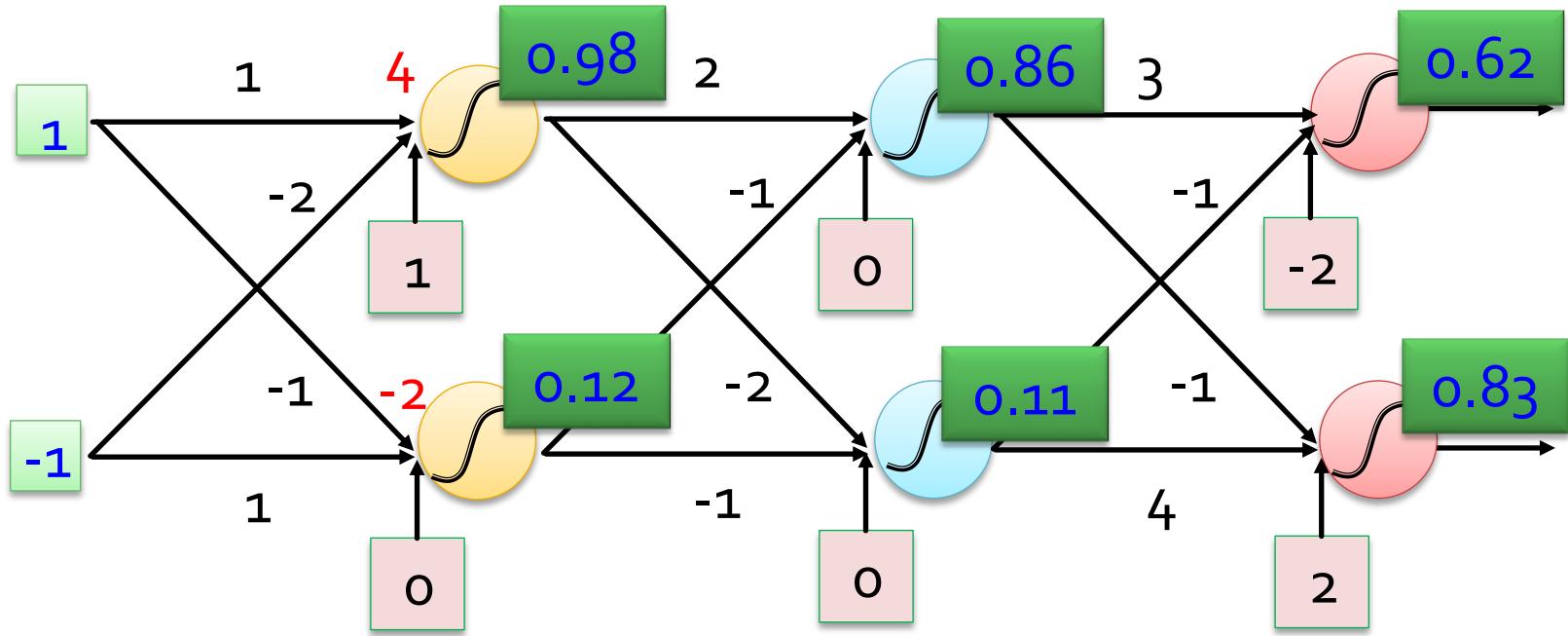


Sigmoid Function

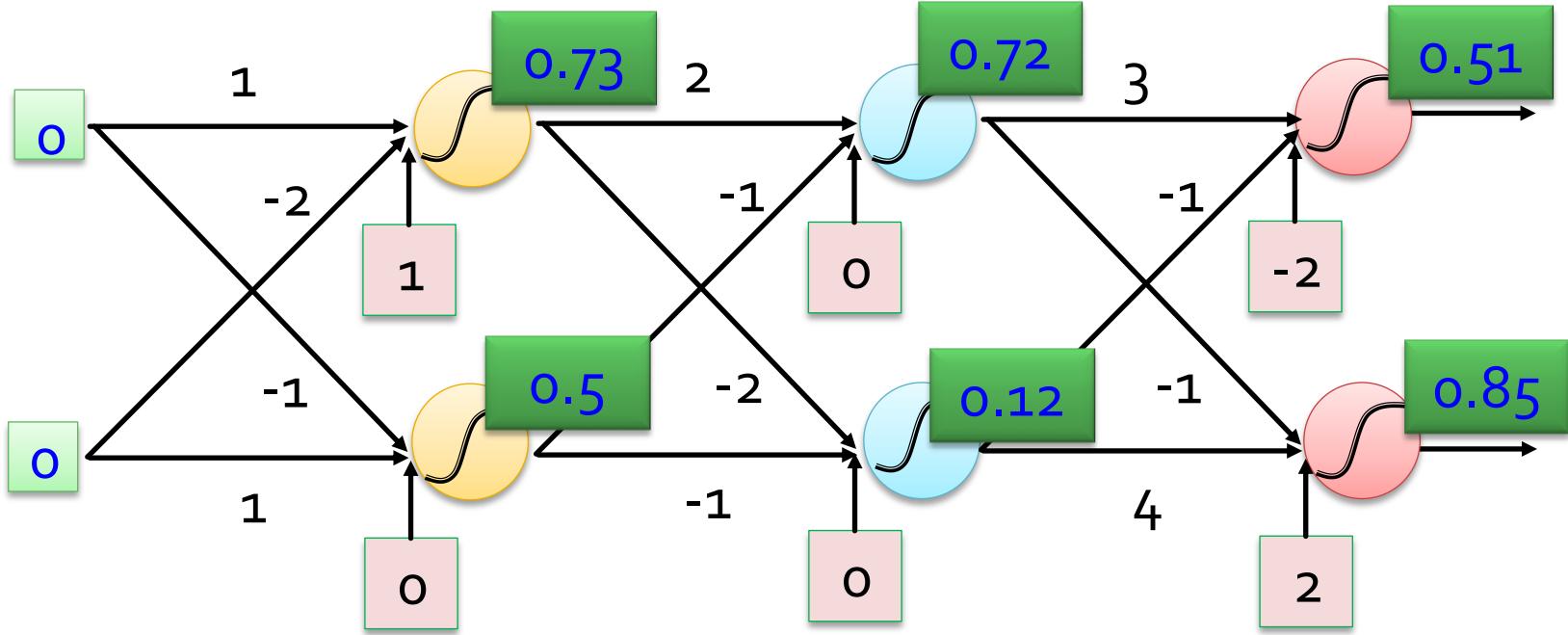
$$\sigma(z) = \frac{1}{1 + e^{-z}}$$



Example of Neural Network



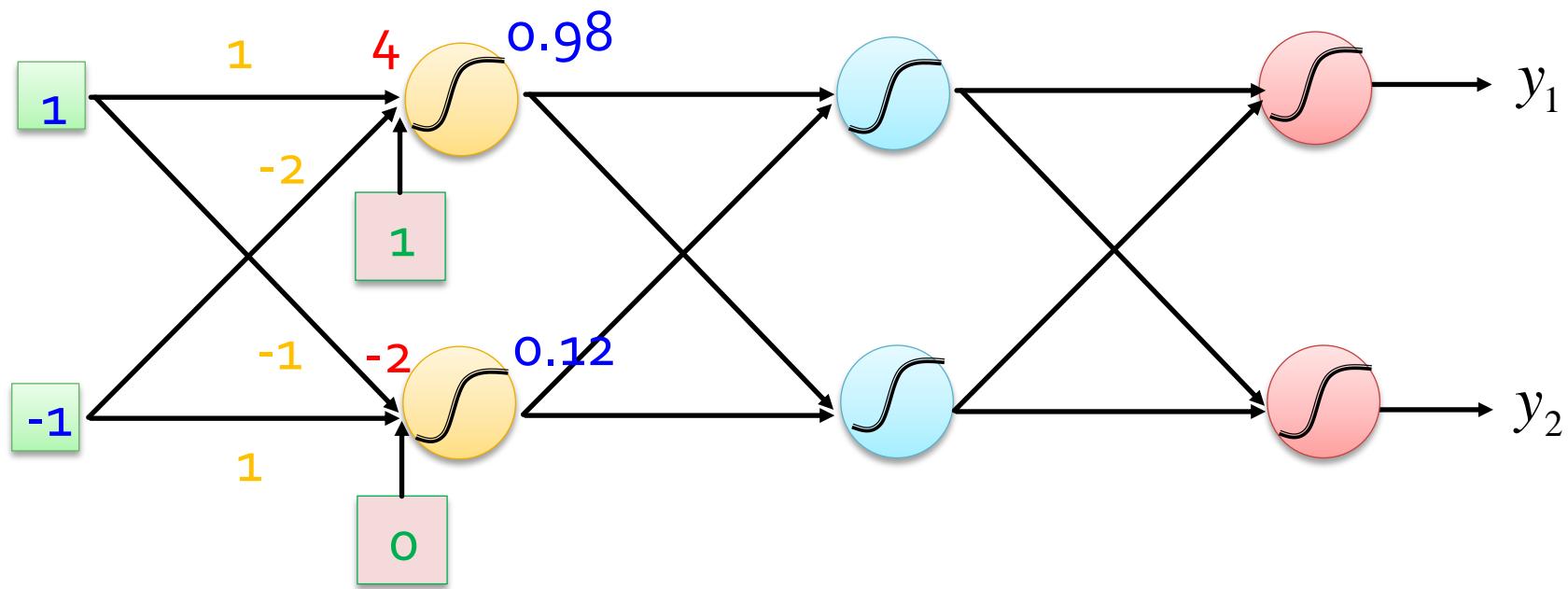
Example of Neural Network



$$f: R^2 \rightarrow R^2 \quad f \left(\begin{bmatrix} 1 \\ -1 \end{bmatrix} \right) = \begin{bmatrix} 0.62 \\ 0.83 \end{bmatrix} \quad f \left(\begin{bmatrix} 0 \\ 0 \end{bmatrix} \right) = \begin{bmatrix} 0.51 \\ 0.85 \end{bmatrix}$$

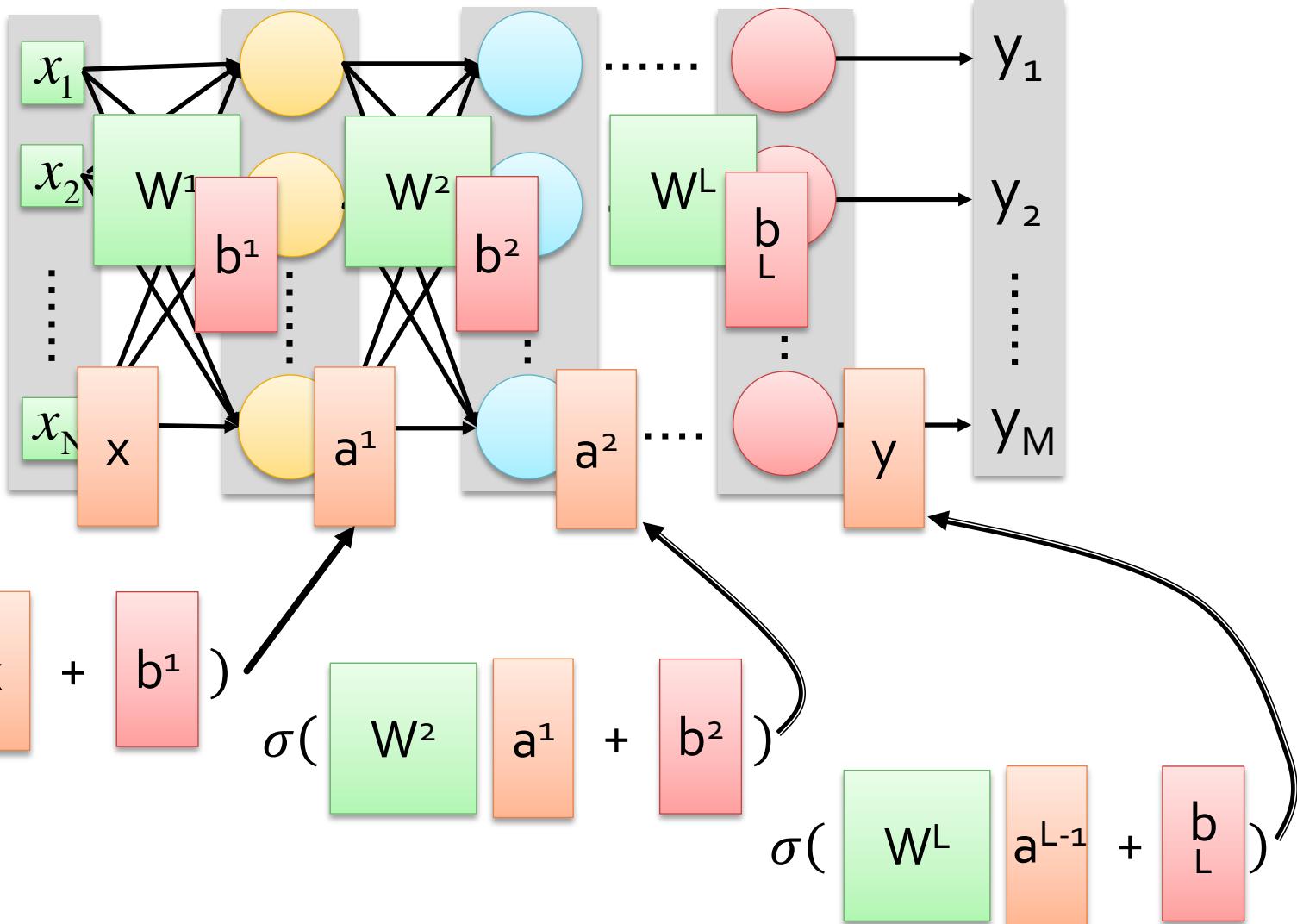
Different parameters define different function

Matrix Operation

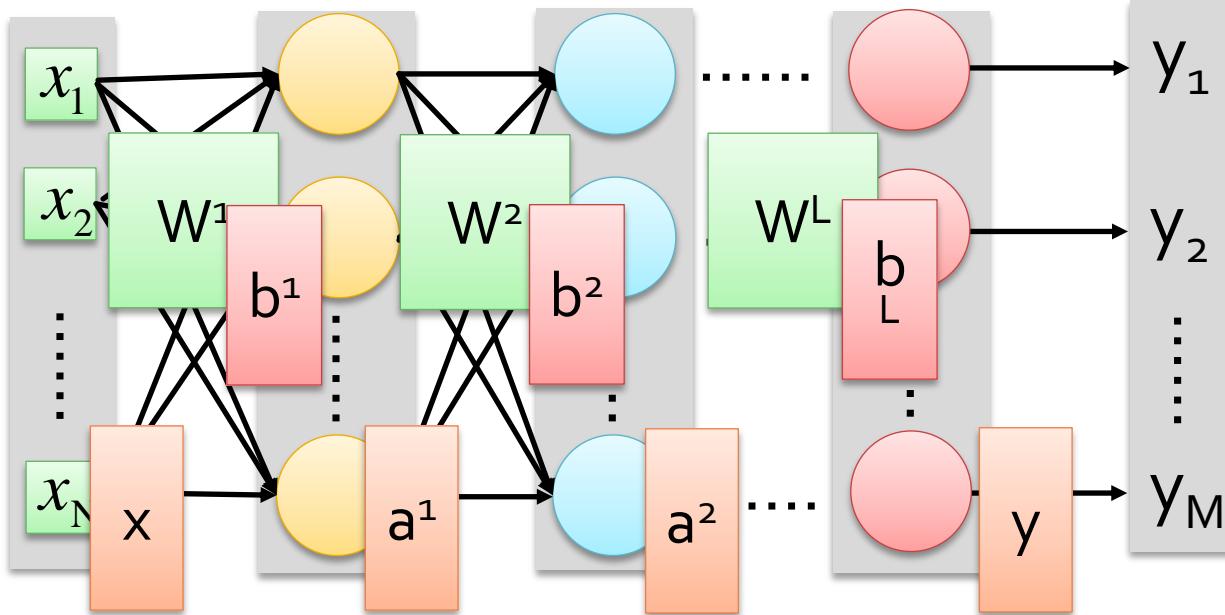


$$\sigma \left(\underbrace{\begin{bmatrix} 1 & -2 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix}}_{\begin{bmatrix} 4 \\ -2 \end{bmatrix}} \right) = \begin{bmatrix} 0.98 \\ 0.12 \end{bmatrix}$$

Neural Network



Neural Network



$$y = f(x)$$

Using parallel computing techniques
to speed up matrix operation

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

Softmax

- Softmax layer as the output layer

Ordinary Layer

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

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

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

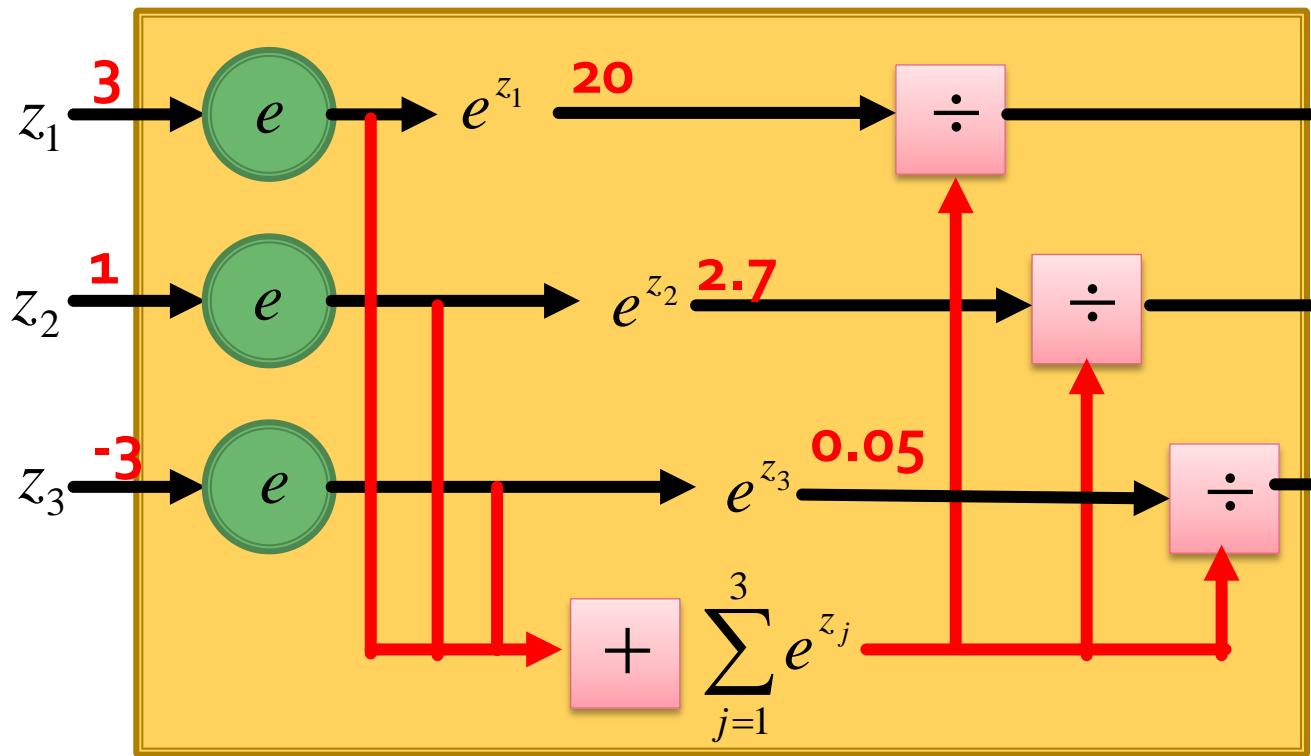
In general, the output of network can be any value.

May not be easy to interpret

Softmax

- Softmax layer as the output layer

Softmax Layer



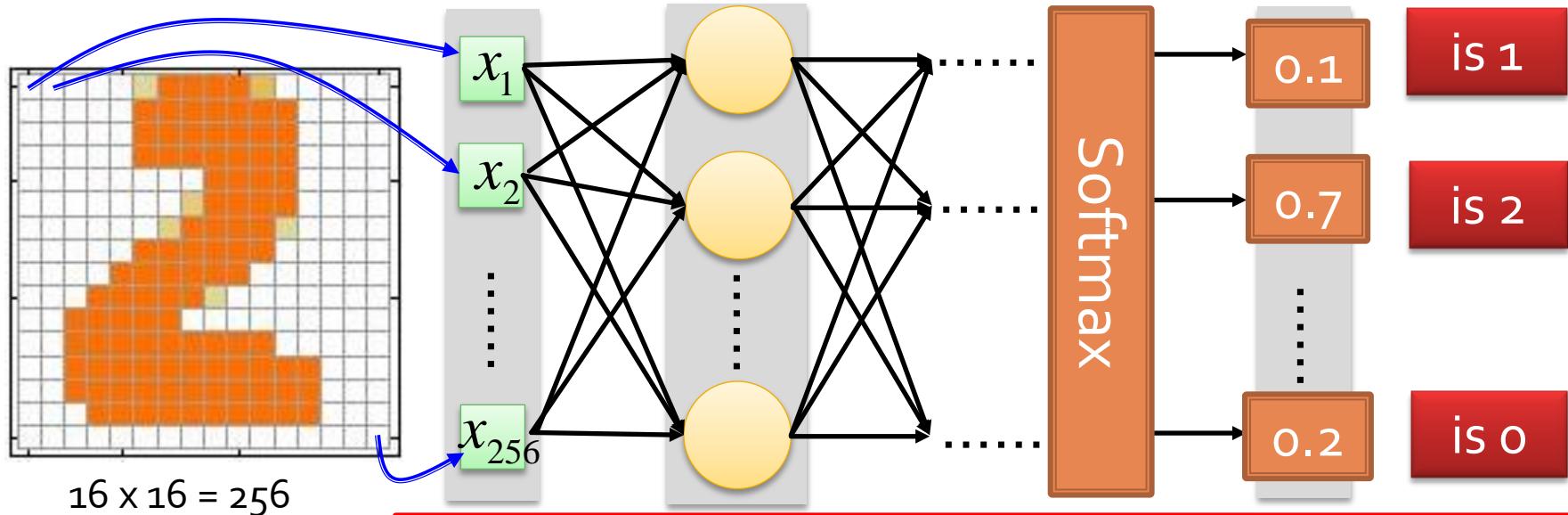
Probability:

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

$$y_1 = e^{z_1} \Big/ \sum_{j=1}^3 e^{z_j}$$
$$y_2 = e^{z_2} \Big/ \sum_{j=1}^3 e^{z_j}$$
$$y_3 = e^{z_3} \Big/ \sum_{j=1}^3 e^{z_j}$$

How to set network parameters

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



Ink $\rightarrow 1$

No ink $\rightarrow 0$

Set the network parameters θ such that

Input: How to let the neural network achieve this maximum value

Input: y_2 has the maximum value

Training Data

- Preparing training data: images and their labels



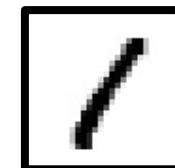
“5”



“0”



“4”



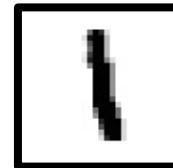
“1”



“9”



“2”



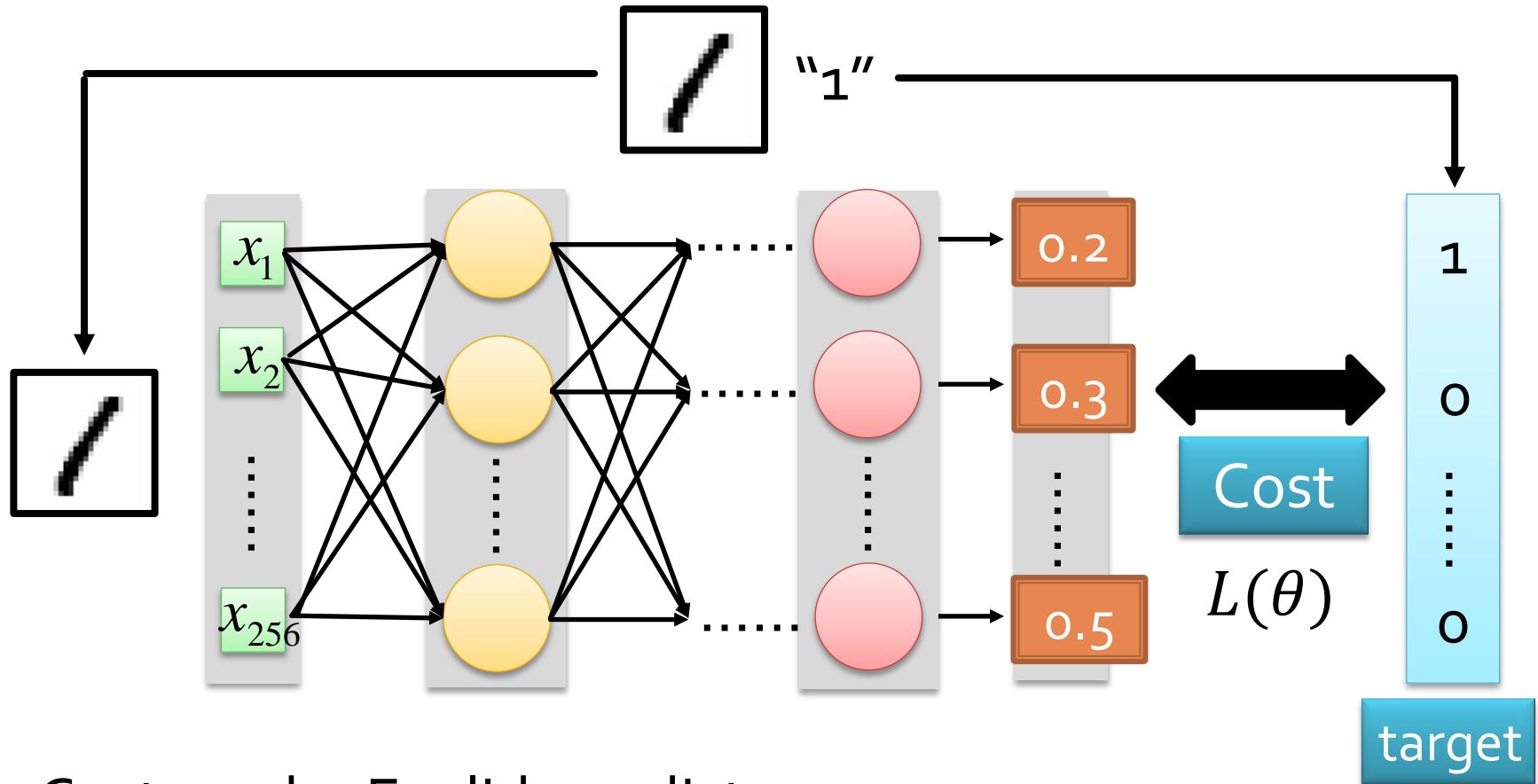
“1”



“3”

Using the training data to find
the network parameters.

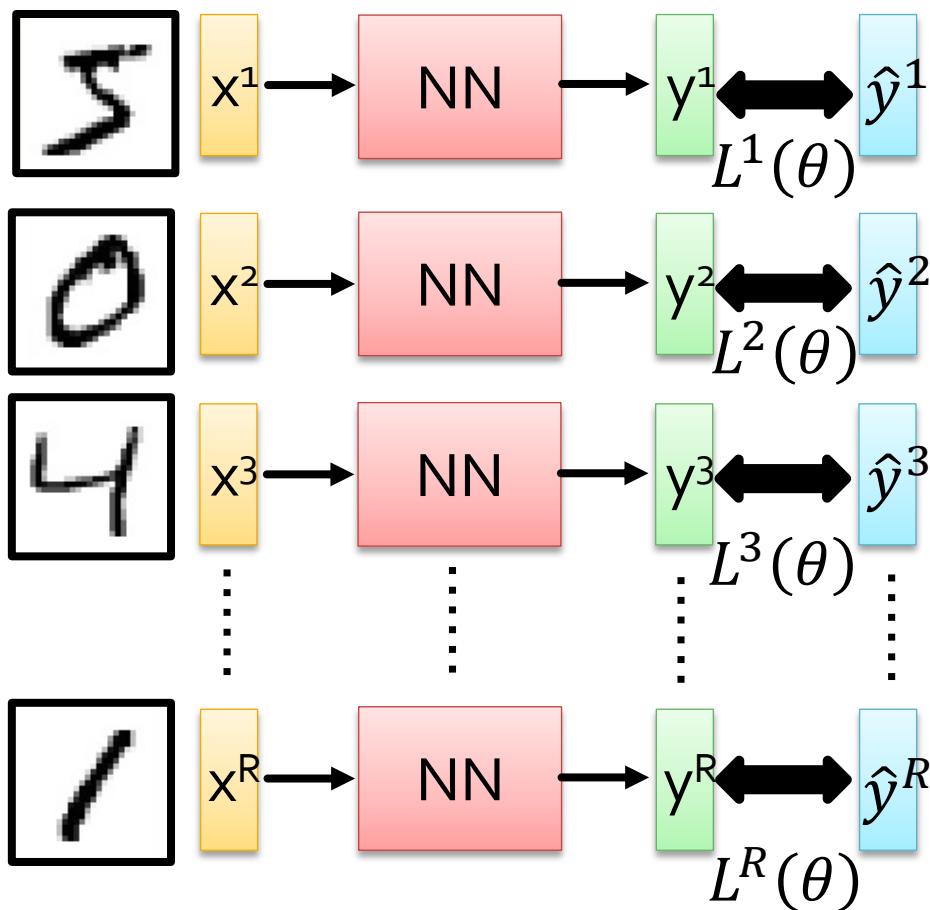
Cost



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

Total Cost

For all training data ...



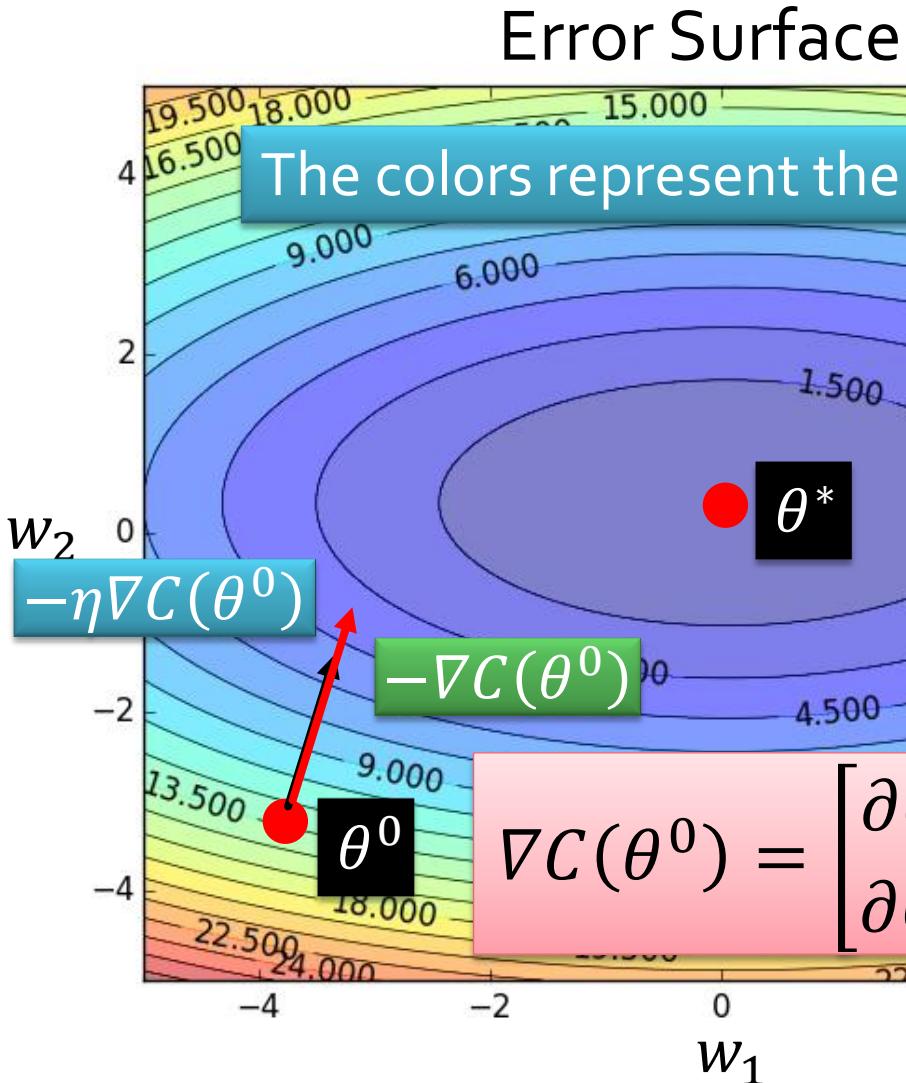
Total Cost:

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

How bad the network parameters θ is on this task

Find the network parameters θ^* that minimize this value

Gradient Descent



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

Randomly pick a starting point θ^0

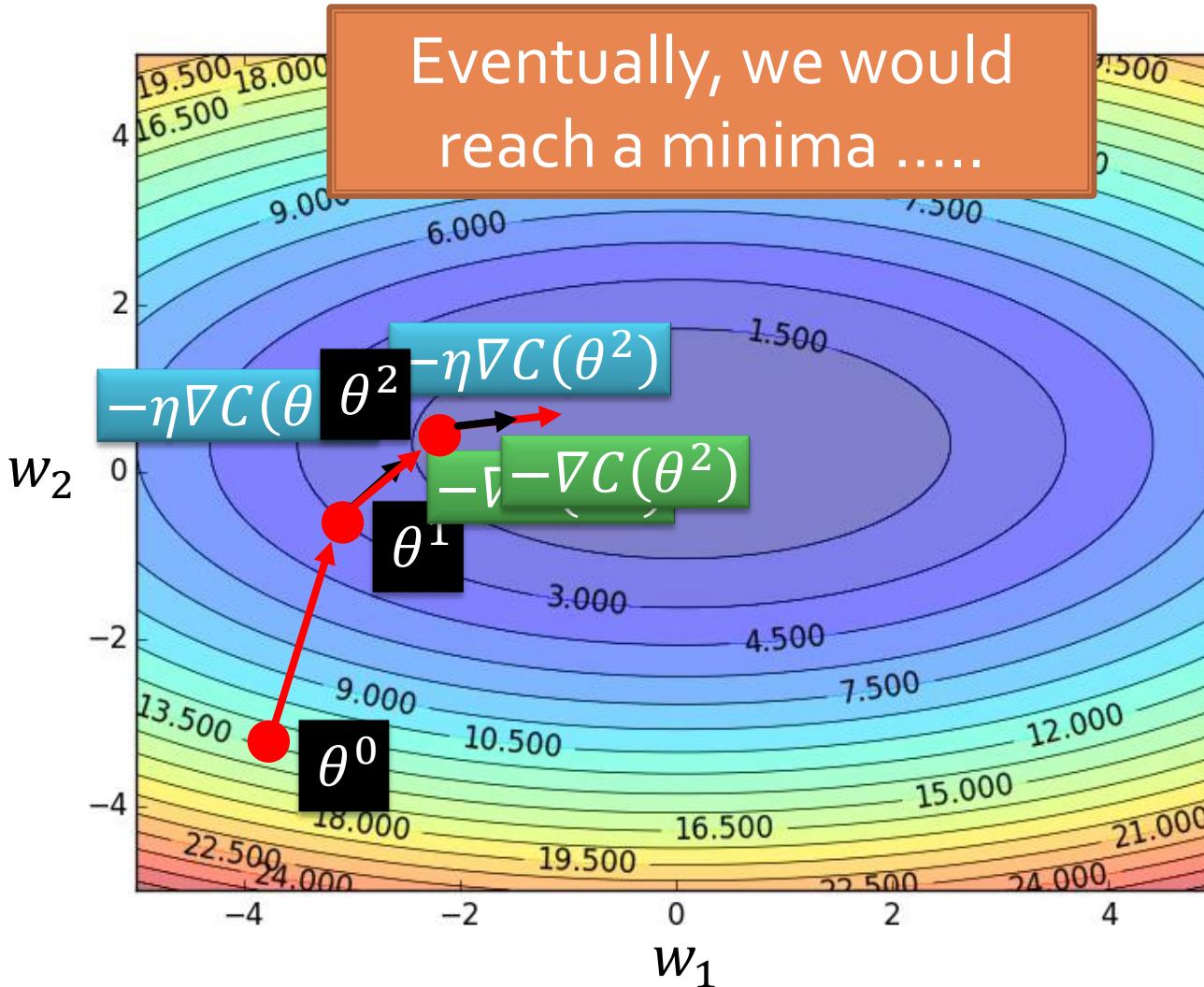
Compute the negative gradient at θ^0

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

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

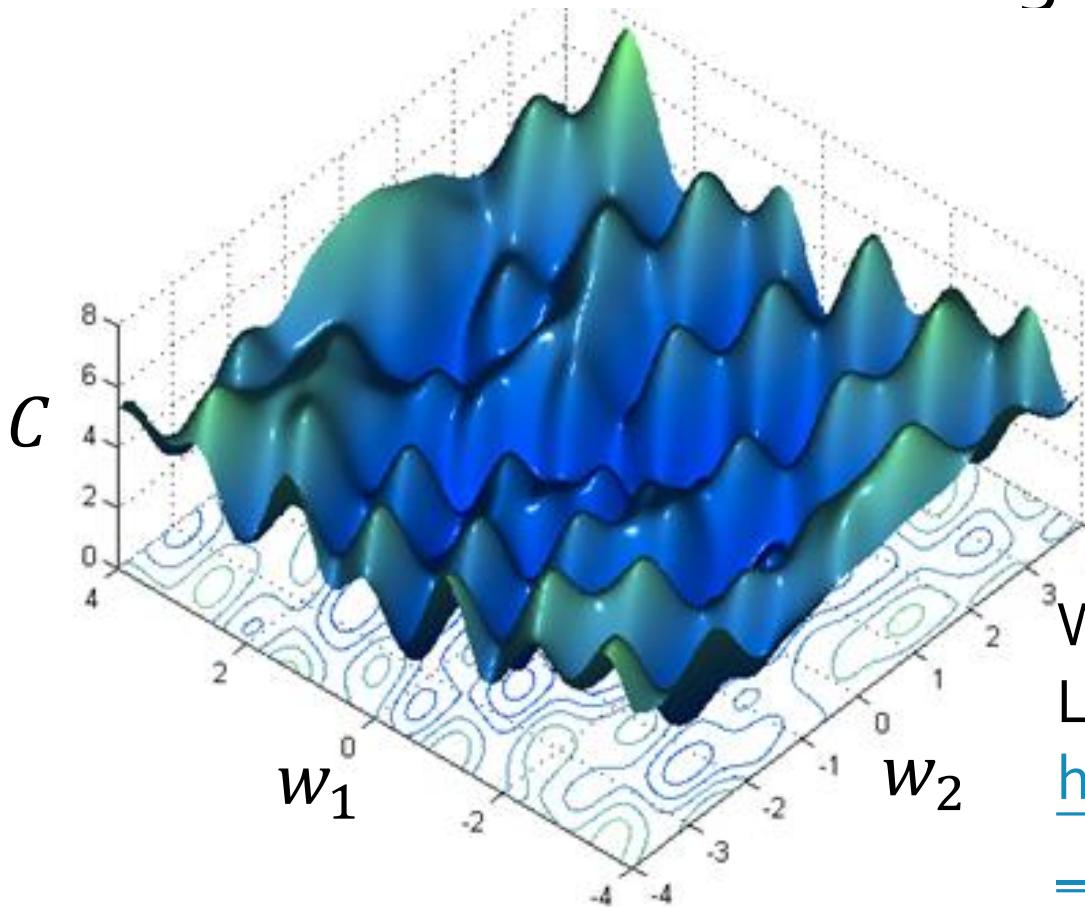
$$\rightarrow -\nabla C(\theta^0)$$

Times the learning rate η

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

Local Minima

- Gradient descent never guarantee global



Different initial point θ^0

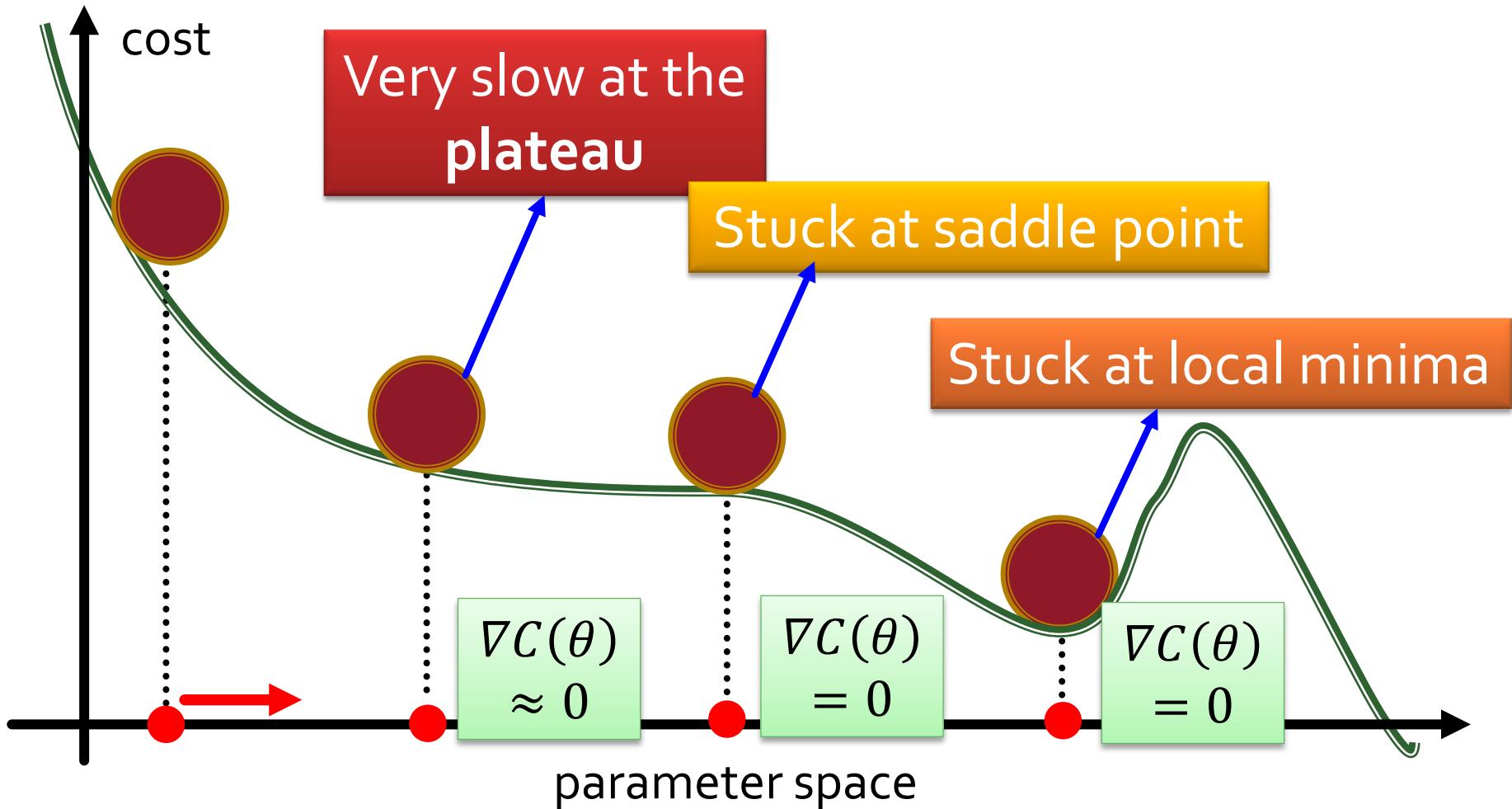


Reach different minima, so different results

Who is Afraid of Non-Convex Loss Functions?

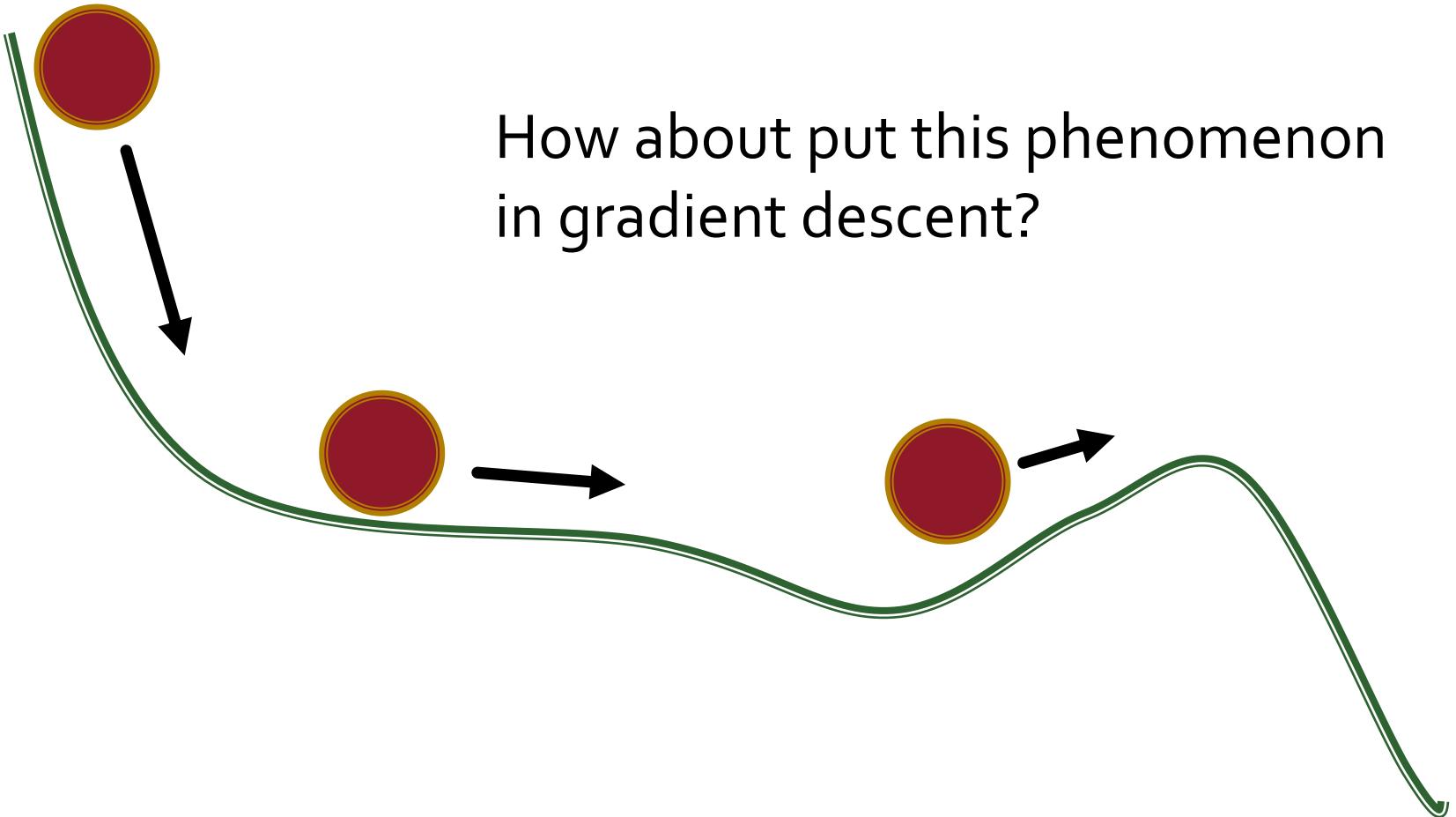
http://videolectures.net/emlo7_lecun_wia/

Besides local minima



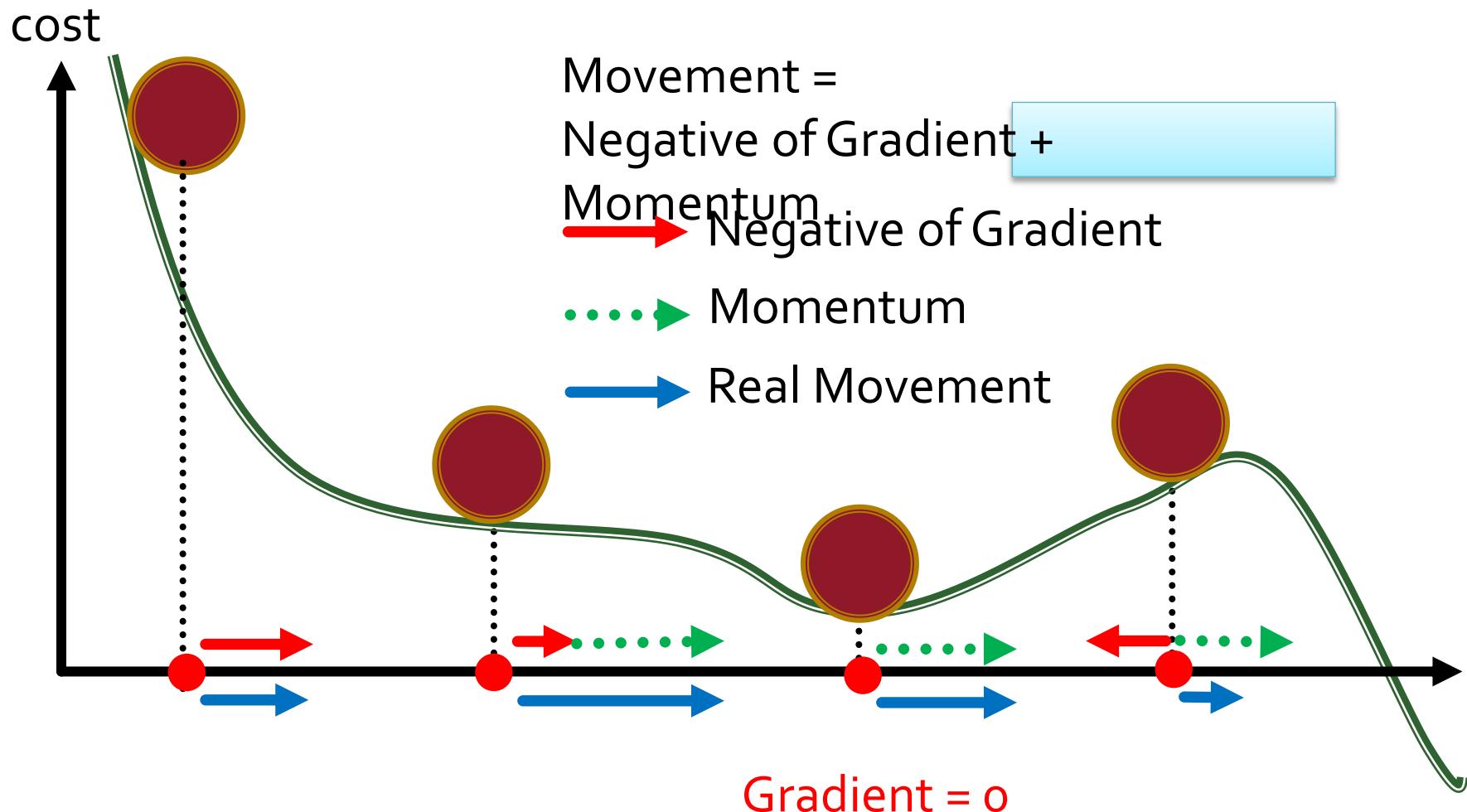
In physical world

- Momentum



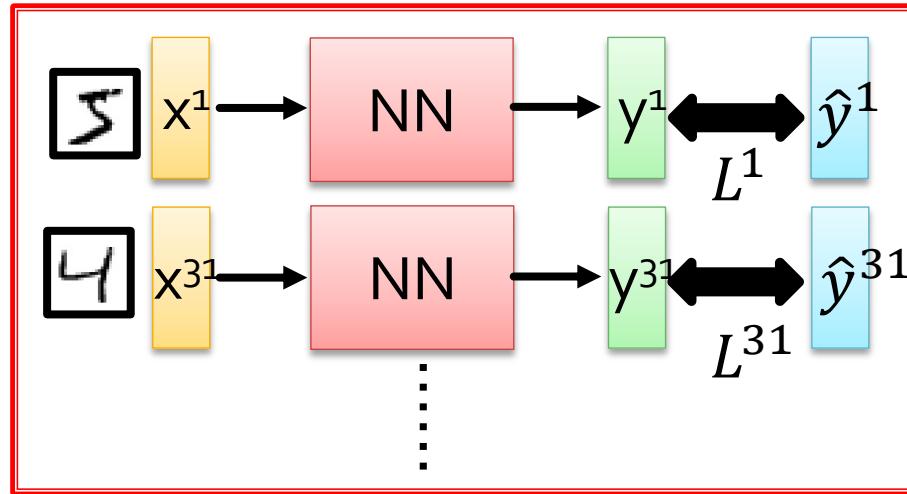
Momentum

Still not guarantee reaching global minima, but give some hope

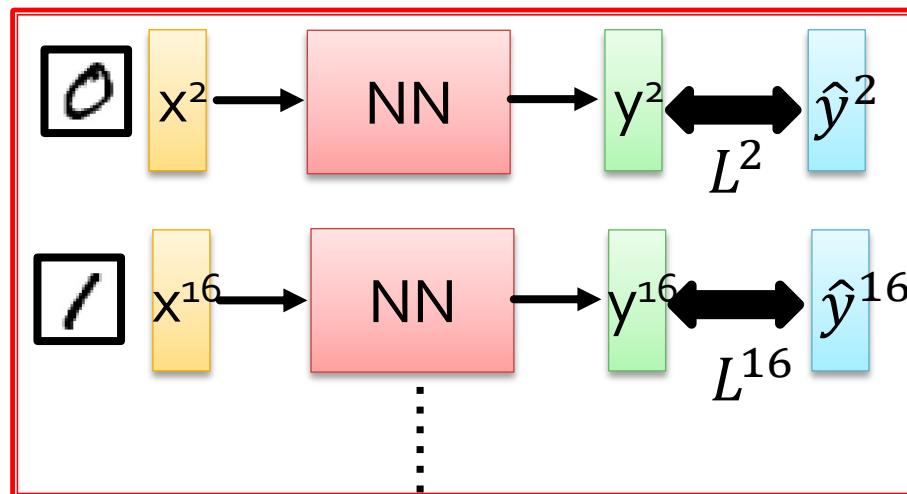


Mini-batch

Mini-batch



Mini-batch

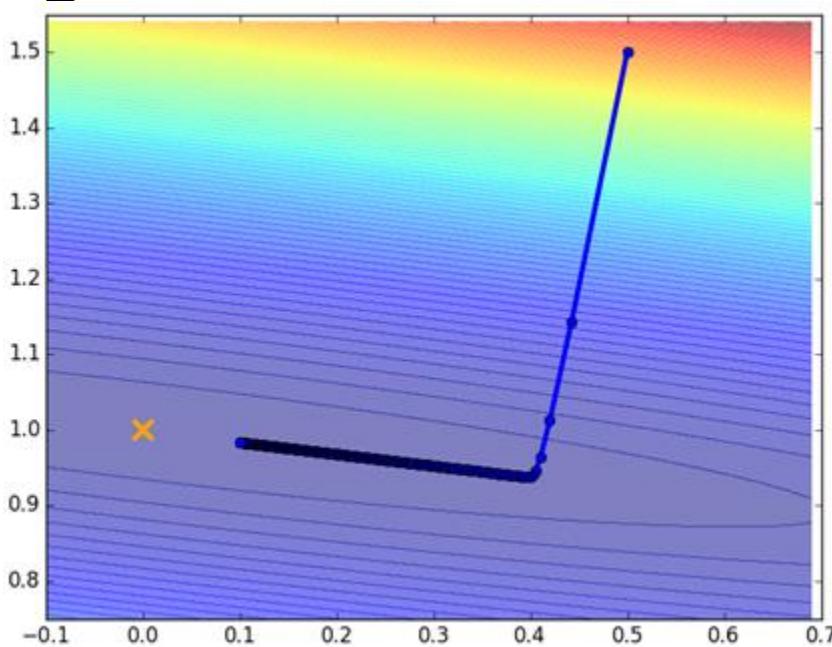


- Randomly initialize θ^0
- Pick the 1st batch
 $C = L^1 + L^{31} + \dots$
 $\theta^1 \leftarrow \theta^0 - \eta \nabla C(\theta^0)$
- Pick the 2nd batch
 $C = L^2 + L^{16} + \dots$
 $\theta^2 \leftarrow \theta^1 - \eta \nabla C(\theta^1)$
⋮

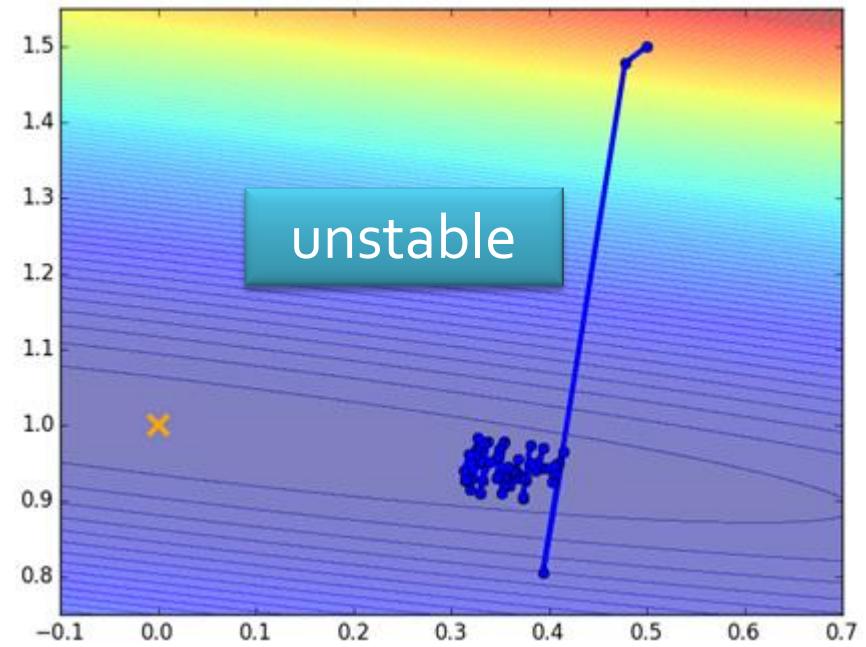
C is different each time
when we update
parameters!

Mini-batch

Original Gradient



With Mini-batch



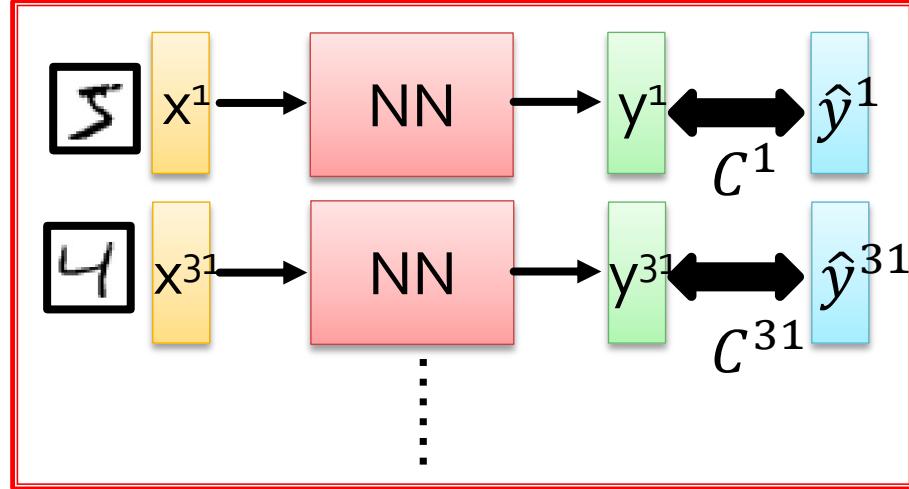
The colors represent the total C on all training data.

Mini-batch

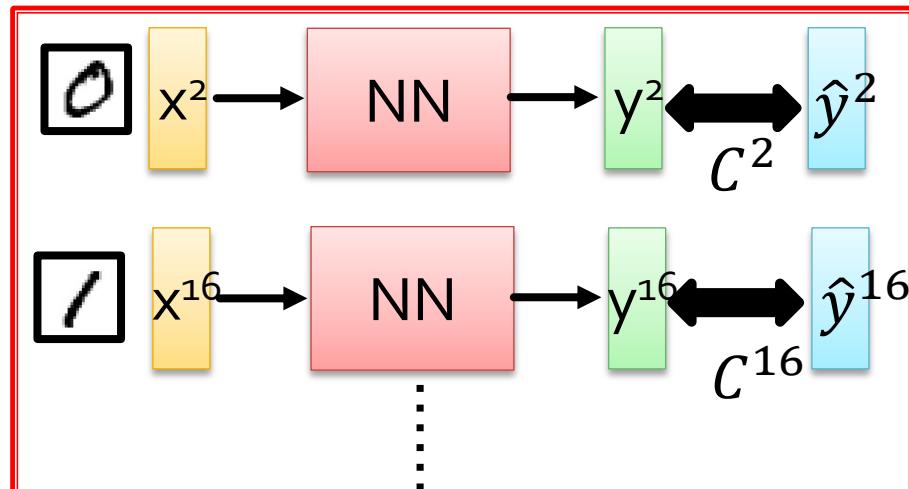
Faster

Better!

Mini-batch



Mini-batch



➤ Randomly initialize θ^0

➤ Pick the 1st batch

$$C = C^1 + C^{31} + \dots$$

$$\theta^1 \leftarrow \theta^0 - \eta \nabla C(\theta^0)$$

➤ Pick the 2nd batch

$$C = C^2 + C^{16} + \dots$$

$$\theta^2 \leftarrow \theta^1 - \eta \nabla C(\theta^1)$$

:

➤ Until all mini-batches have been picked

one epoch

Repeat the above process

Backpropagation

- A network can have millions of parameters.
 - Backpropagation is the way to compute the gradients efficiently
 - Ref:
http://speech.ee.ntu.edu.tw/~tlkagk/courses/MLDS_2015_2/Lecture/DNN%20backprop.ecm.mp4/index.html
- Many toolkits can compute the gradients automatically

theano

Ref:

http://speech.ee.ntu.edu.tw/~tlkagk/courses/MLDS_2015_2/Lecture/Theano%20DNN.ecm.mp4/index.html



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

Not surprised, more parameters, better performance

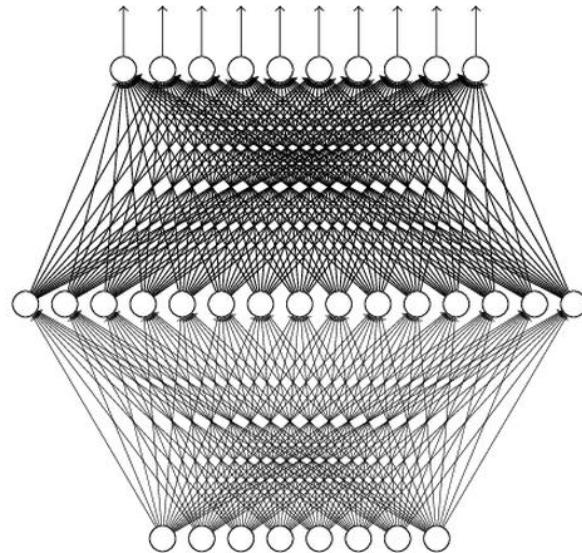
Seide, Frank, Gang Li, and Dong Yu. "Conversational Speech Transcription Using Context-Dependent Deep Neural Networks." *Interspeech*. 2011.

Universality Theorem

Any continuous function f

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

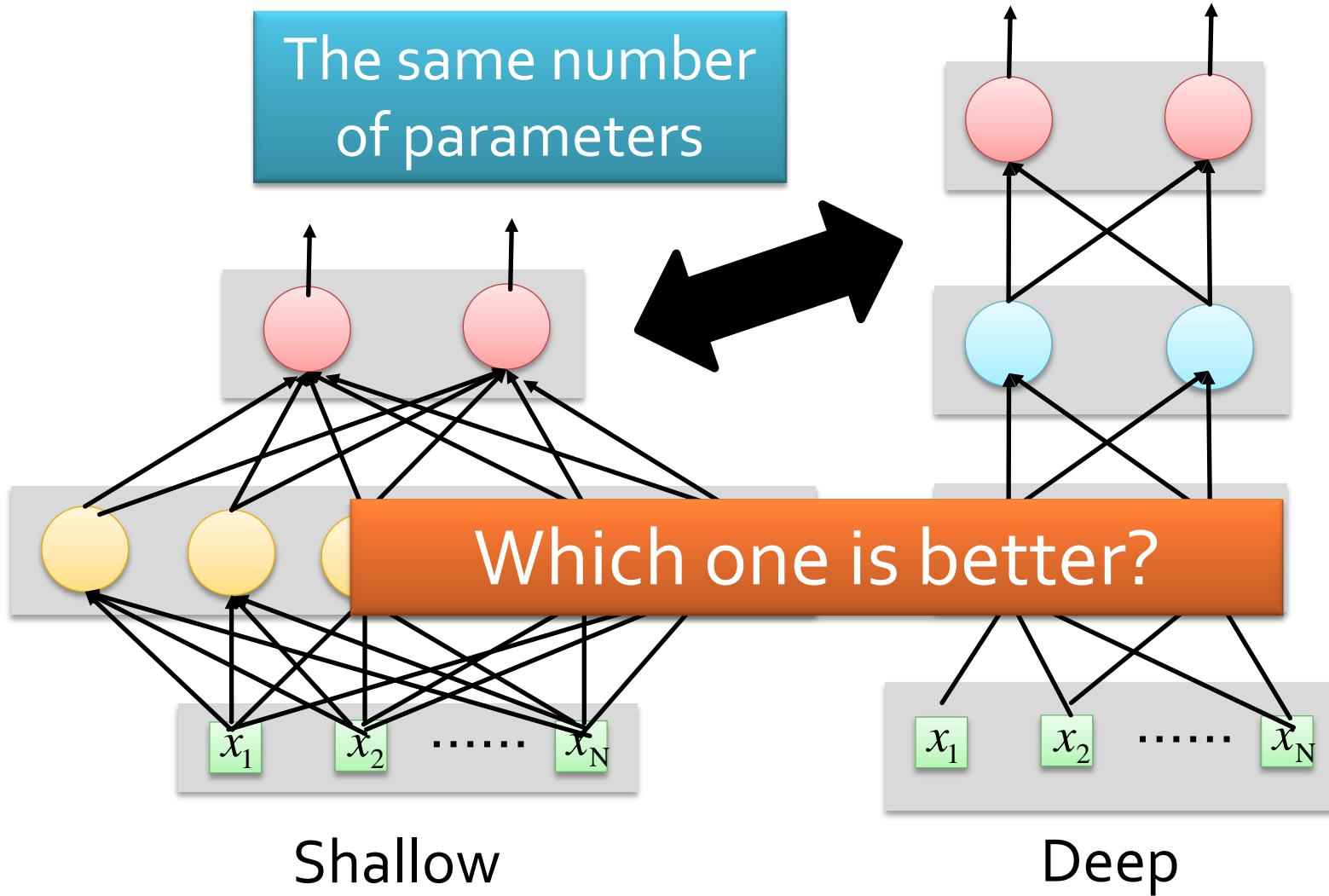
Can be realized by a network with one hidden layer
(given **enough** hidden neurons)



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

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

Fat + Short v.s. Thin + Tall



Fat + Short v.s. Thin + Tall

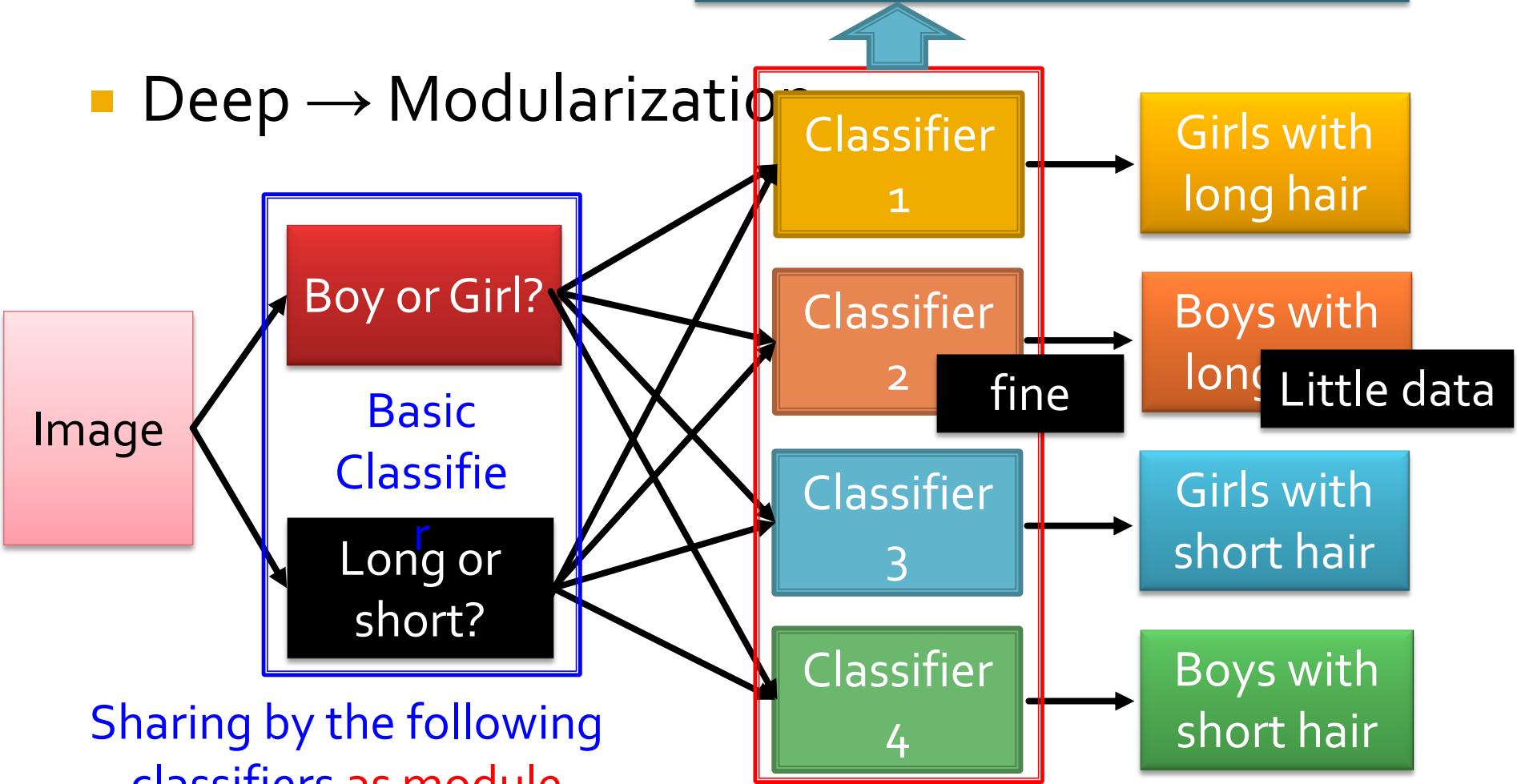
Layer X Size	Word Error Rate (%)	Layer X Size	Word Error Rate (%)
1 X 2k	24.2		
2 X 2k	20.4		
3 X 2k	18.4		
4 X 2k	17.8		
5 X 2k	17.2	↔↔ 1 X 3772	22.5
7 X 2k	17.1	↔↔ 1 X 4634	22.6
		1 X 16k	22.1

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

Why Deep?

can be trained by little data

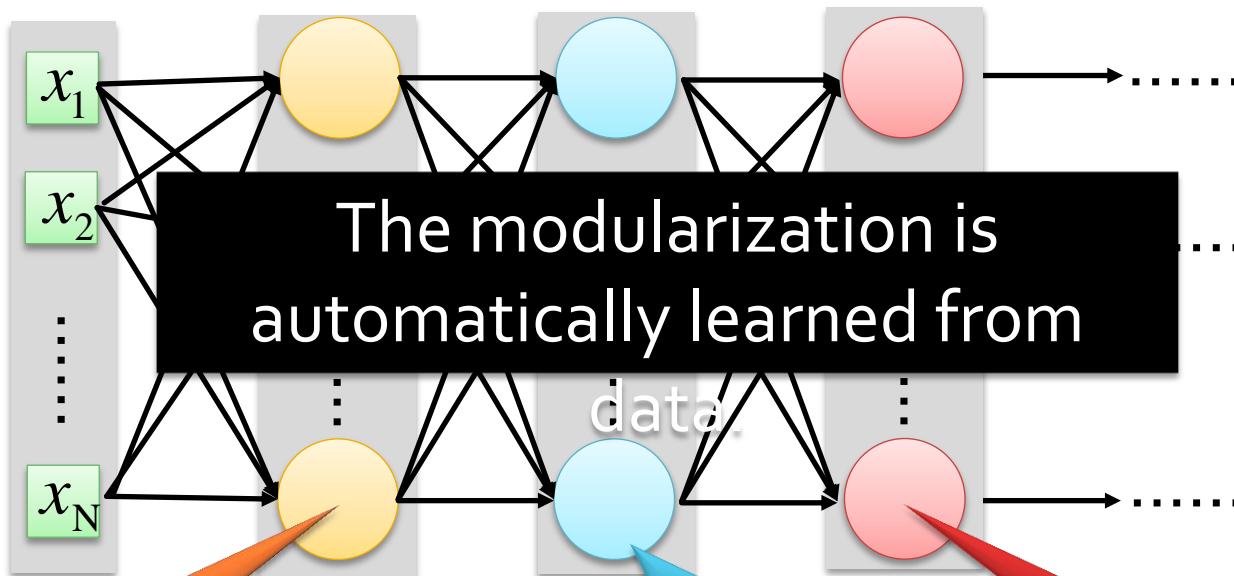
- Deep → Modularization



Why Deep?

Deep Learning also works on small data set like TIMIT.

- Deep → Modularizat → Less training data?

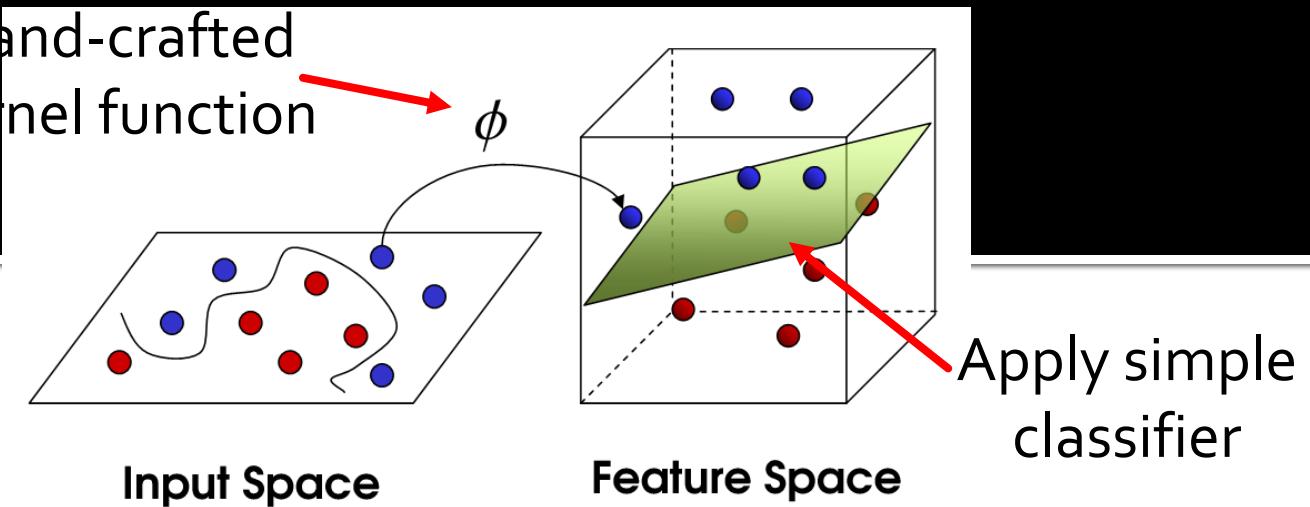


The most basic classifiers

Use 1st layer as module to build classifiers

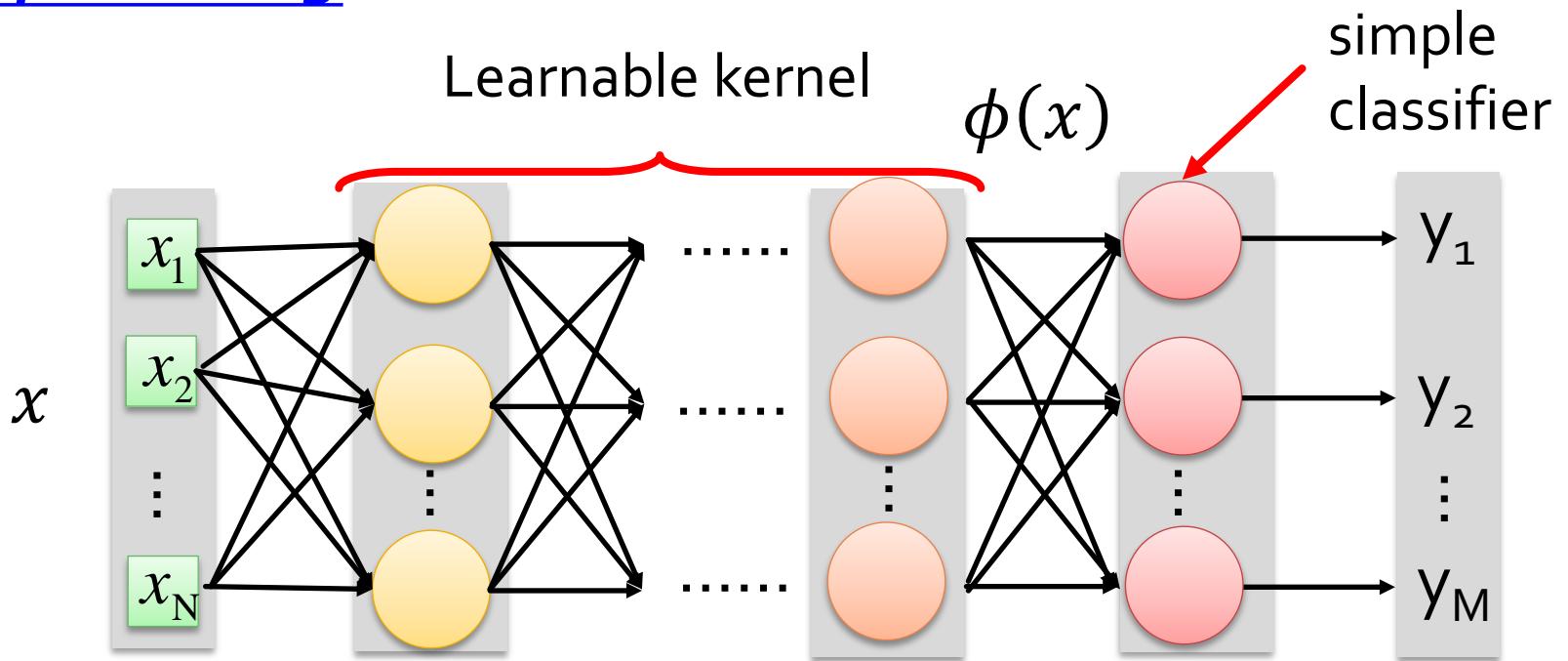
Use 2nd layer as module

SVM



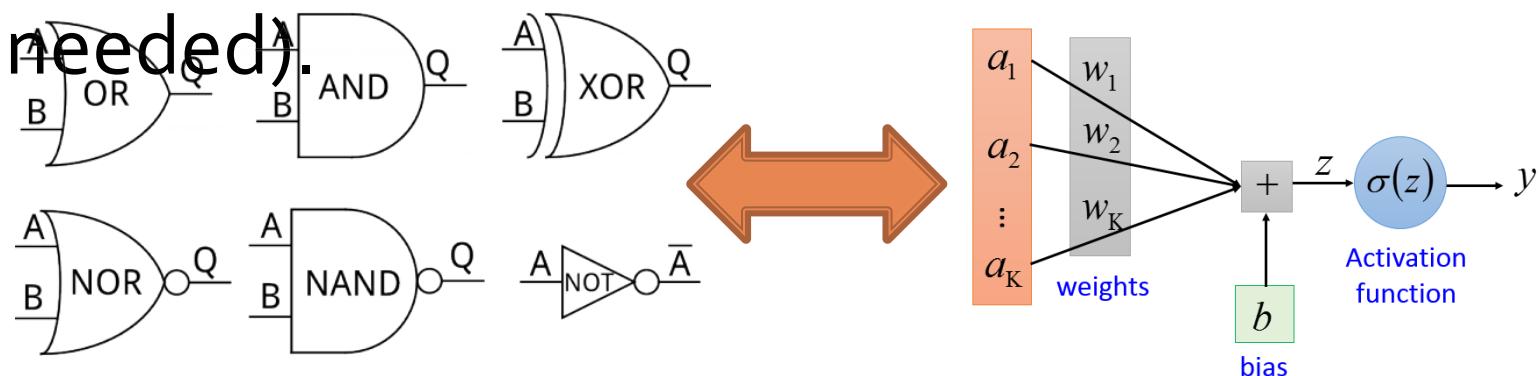
Deep Learning

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

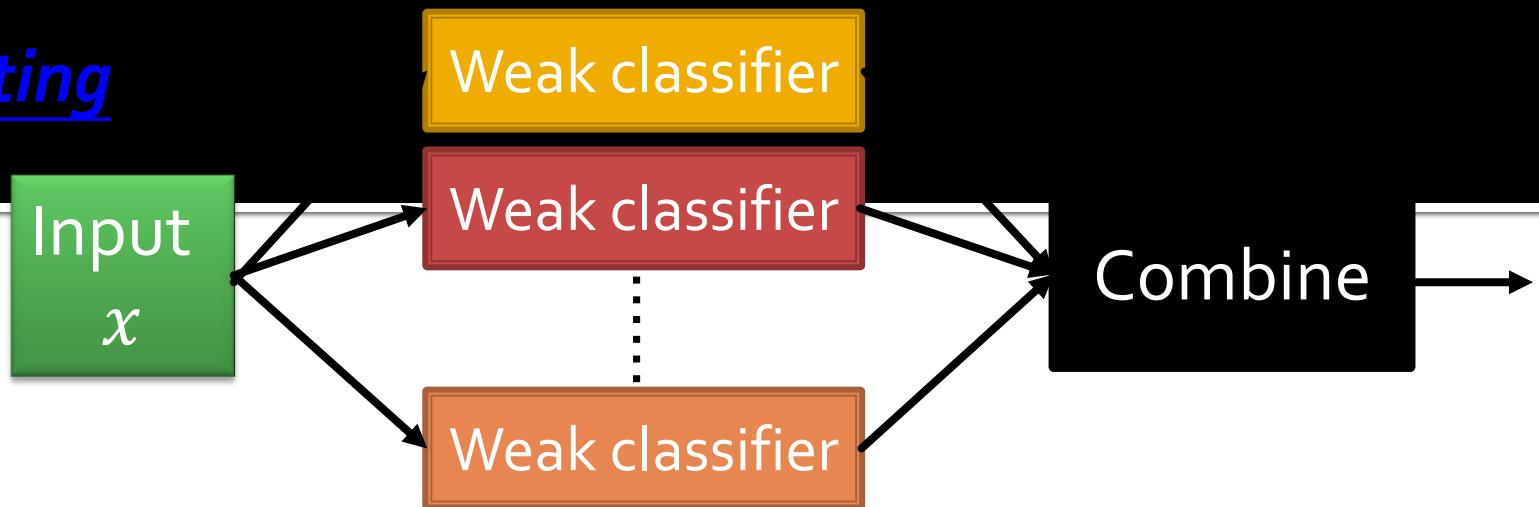


Why Deep? – Logic Circuits

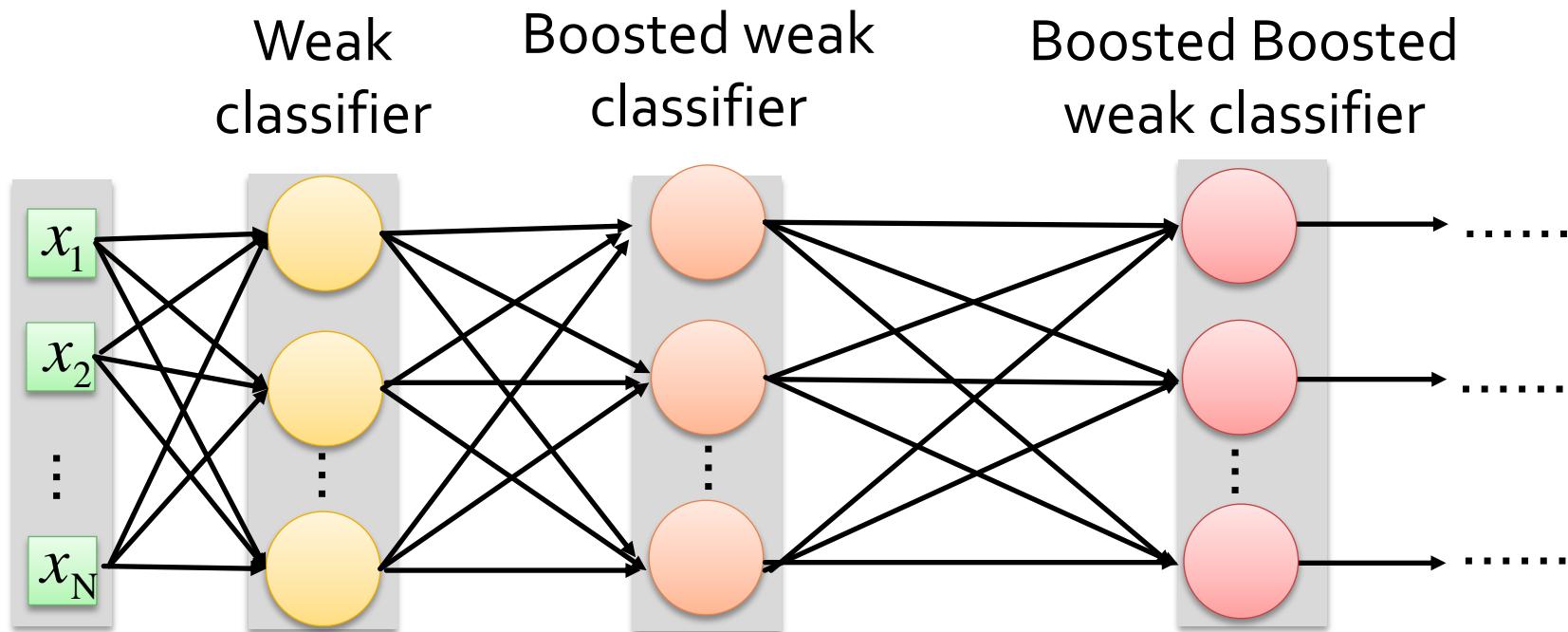
- A two levels of basic logic gates can represent any Boolean function.
- However, no one uses two levels of logic gates to build computers
- Using multiple layers of logic gates to build some functions are much simpler (less gates needed).



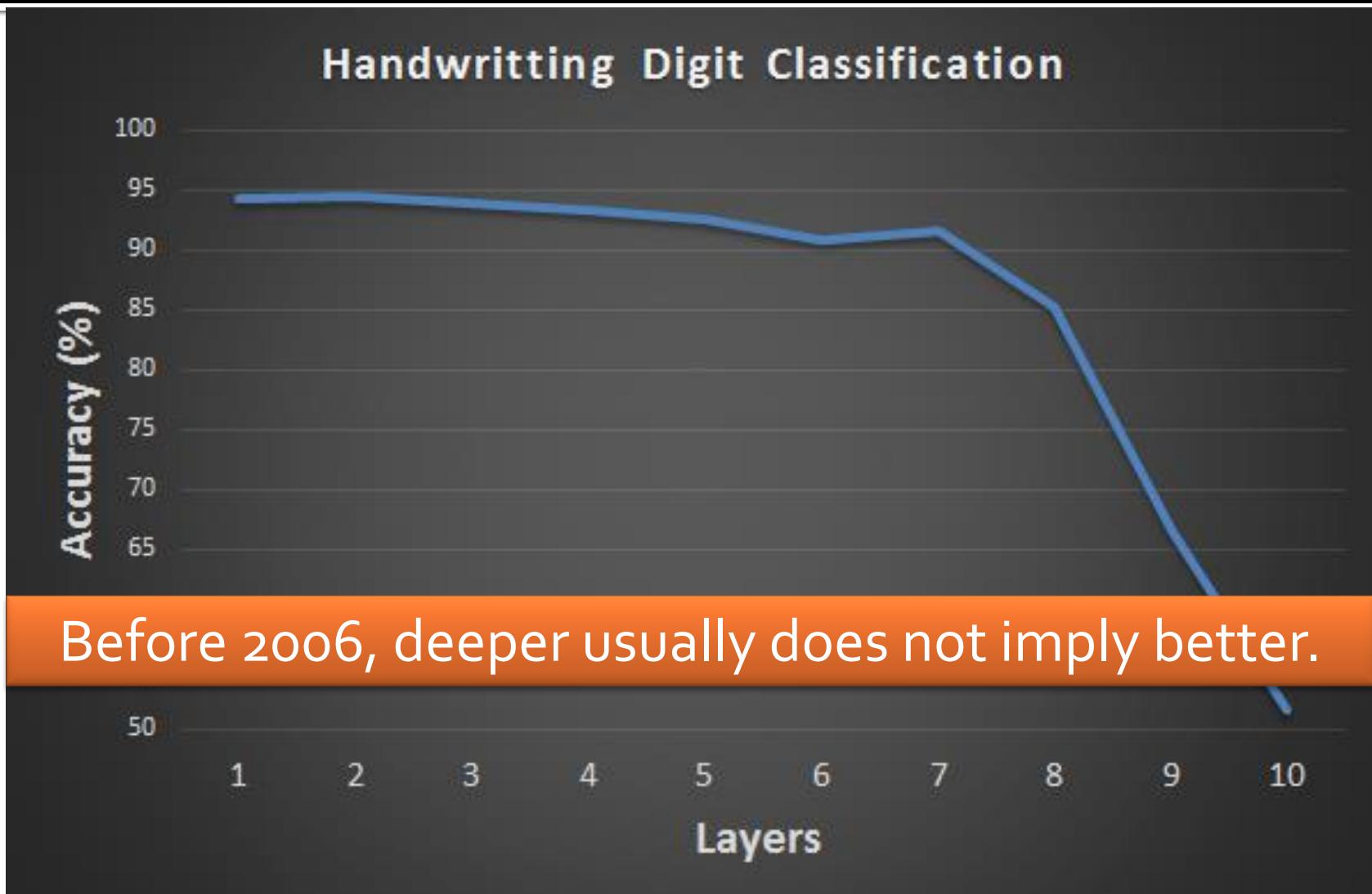
Boosting



Deep Learning

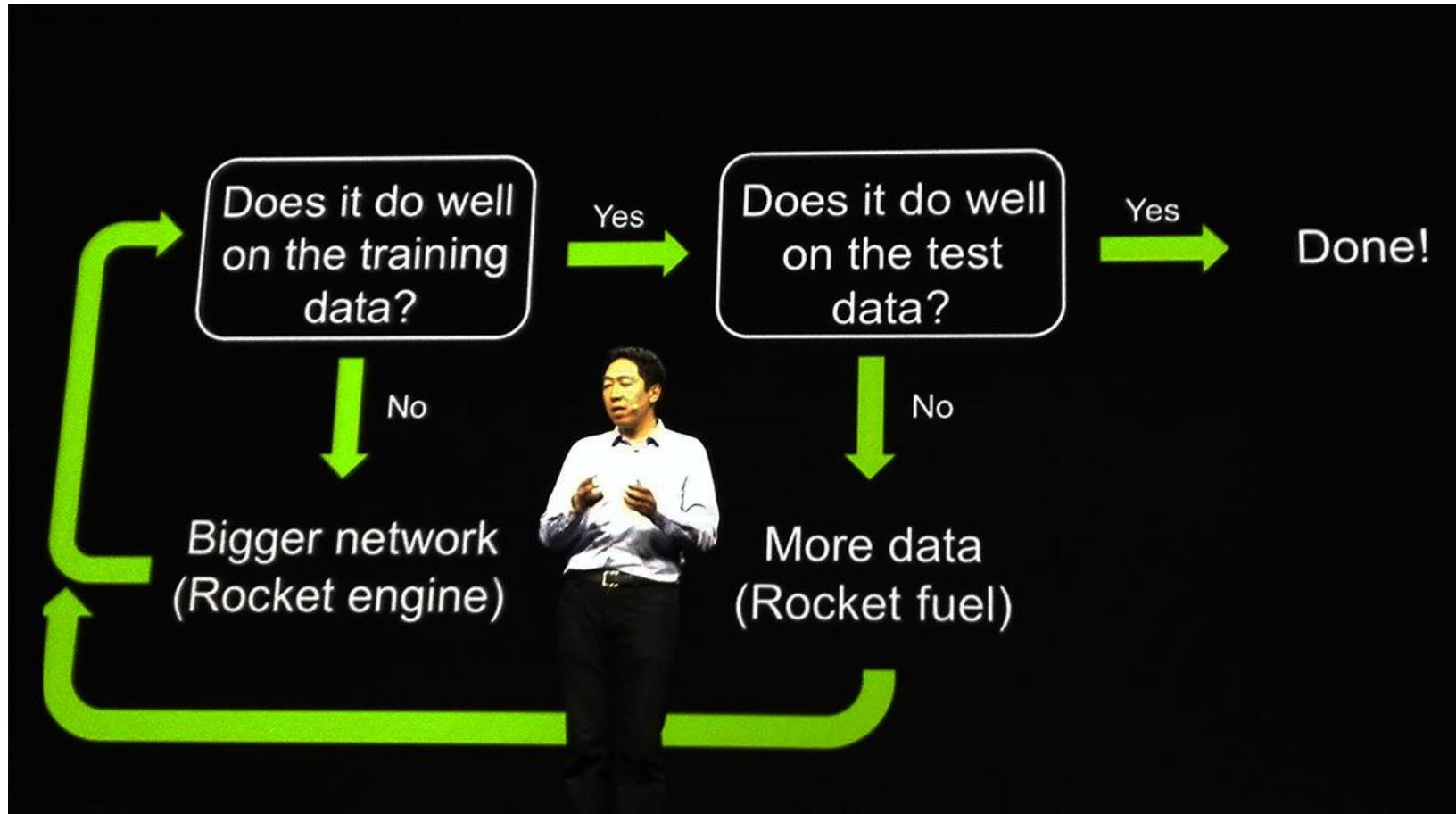


Hard to get the power of Deep ...



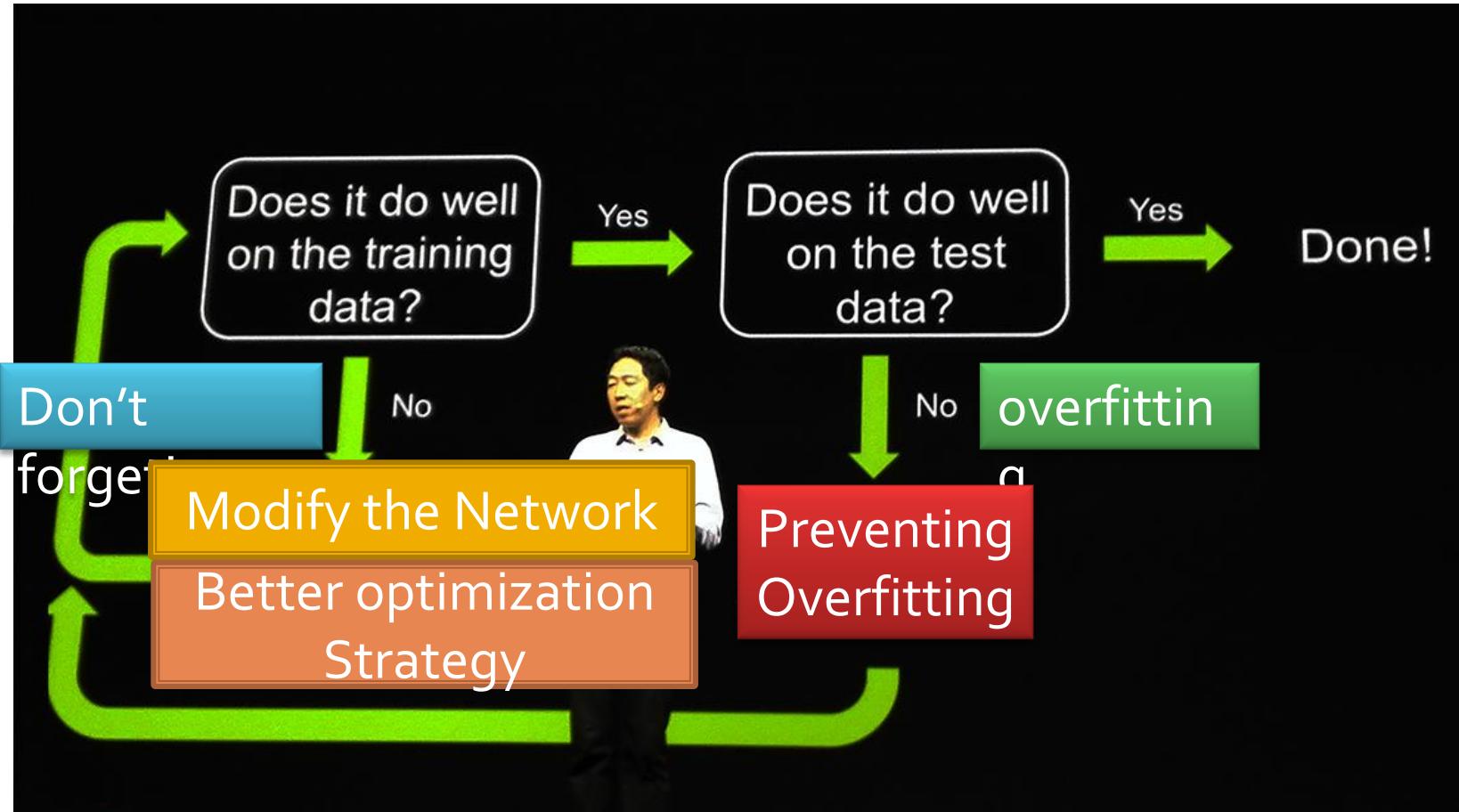
Part III: Tips for Training DNN

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

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.

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/>