Deep Learning Tutorial

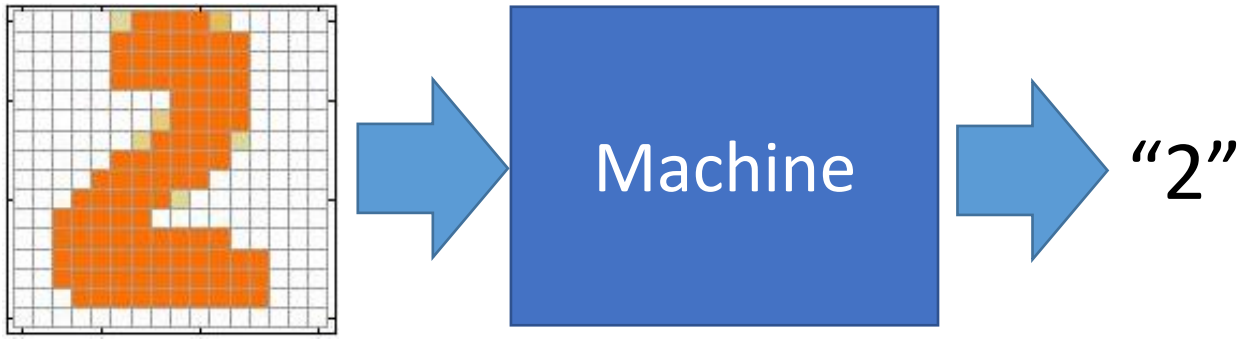
[Slides from Hung-yi Lee and Kai Yu]

Part I:
Introduction of Deep Learning

What people already knew in 1980s

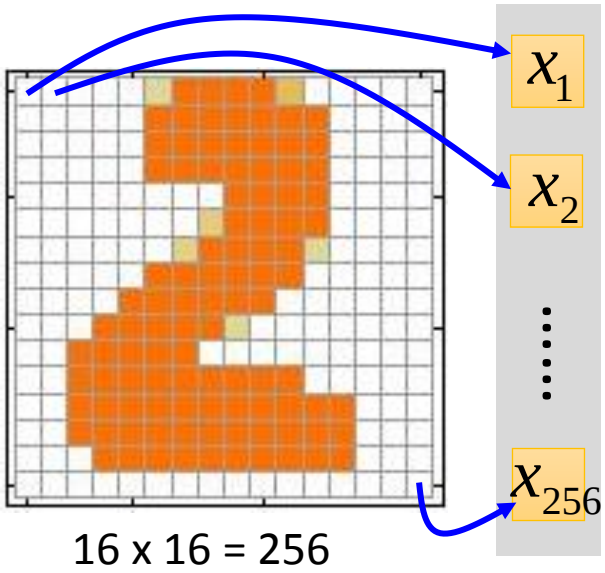
Example Application

- Handwriting Digit Recognition



Handwriting Digit Recognition

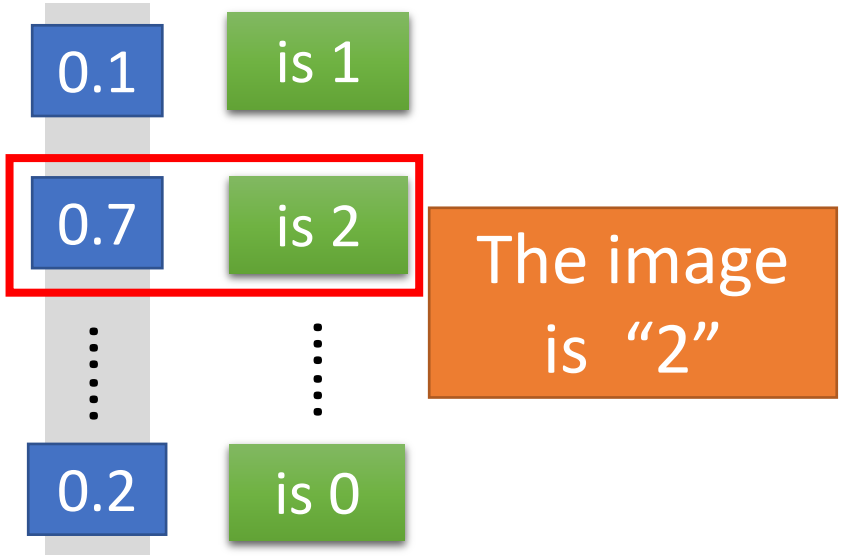
Input



Ink → 1

No ink → 0

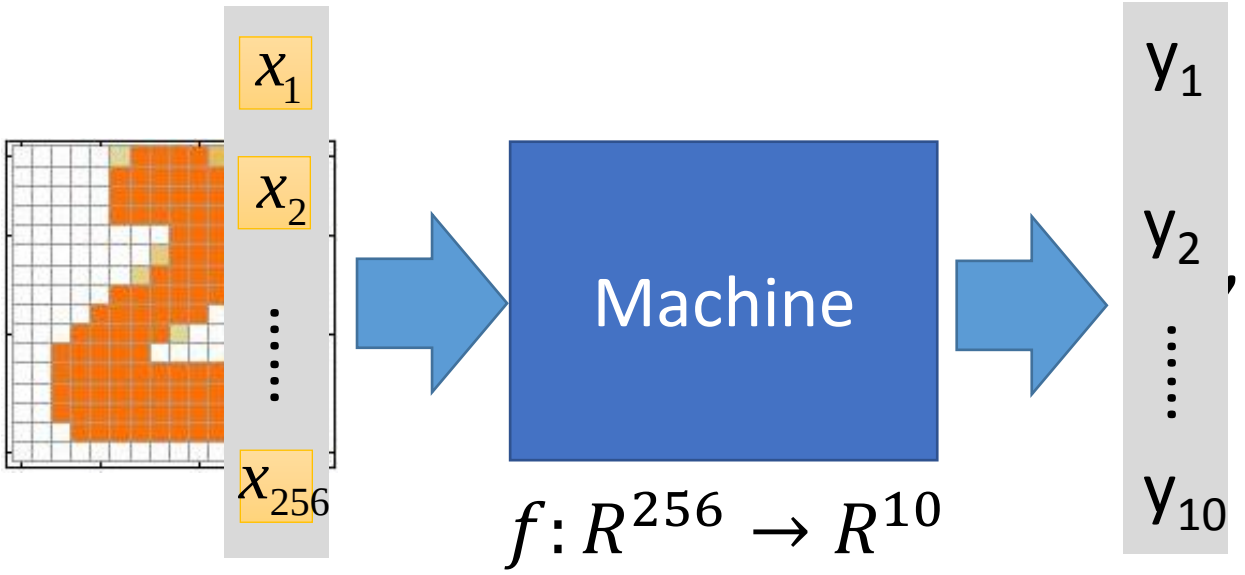
Output



Each dimension represents the confidence of a digit.

Example Application

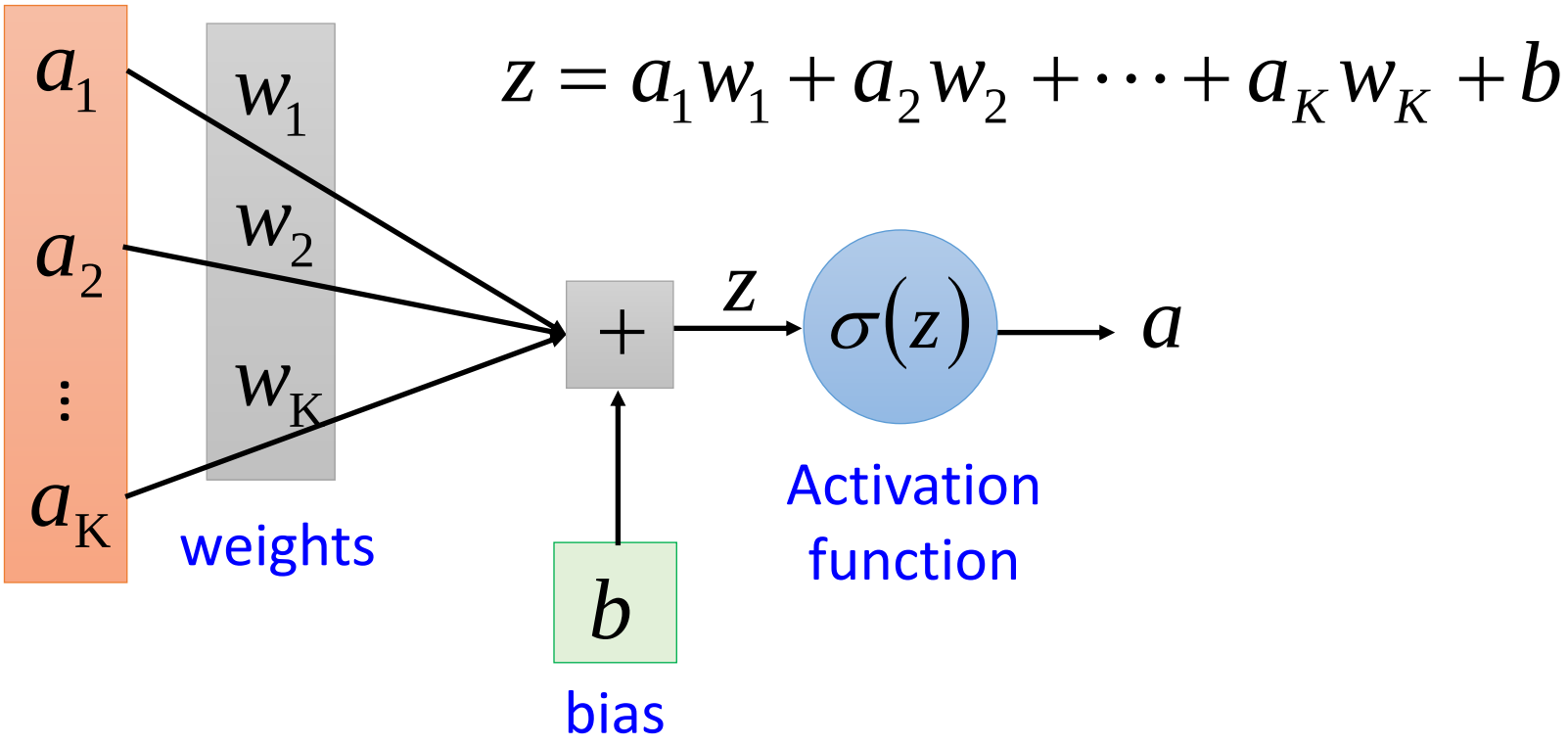
- Handwriting Digit Recognition



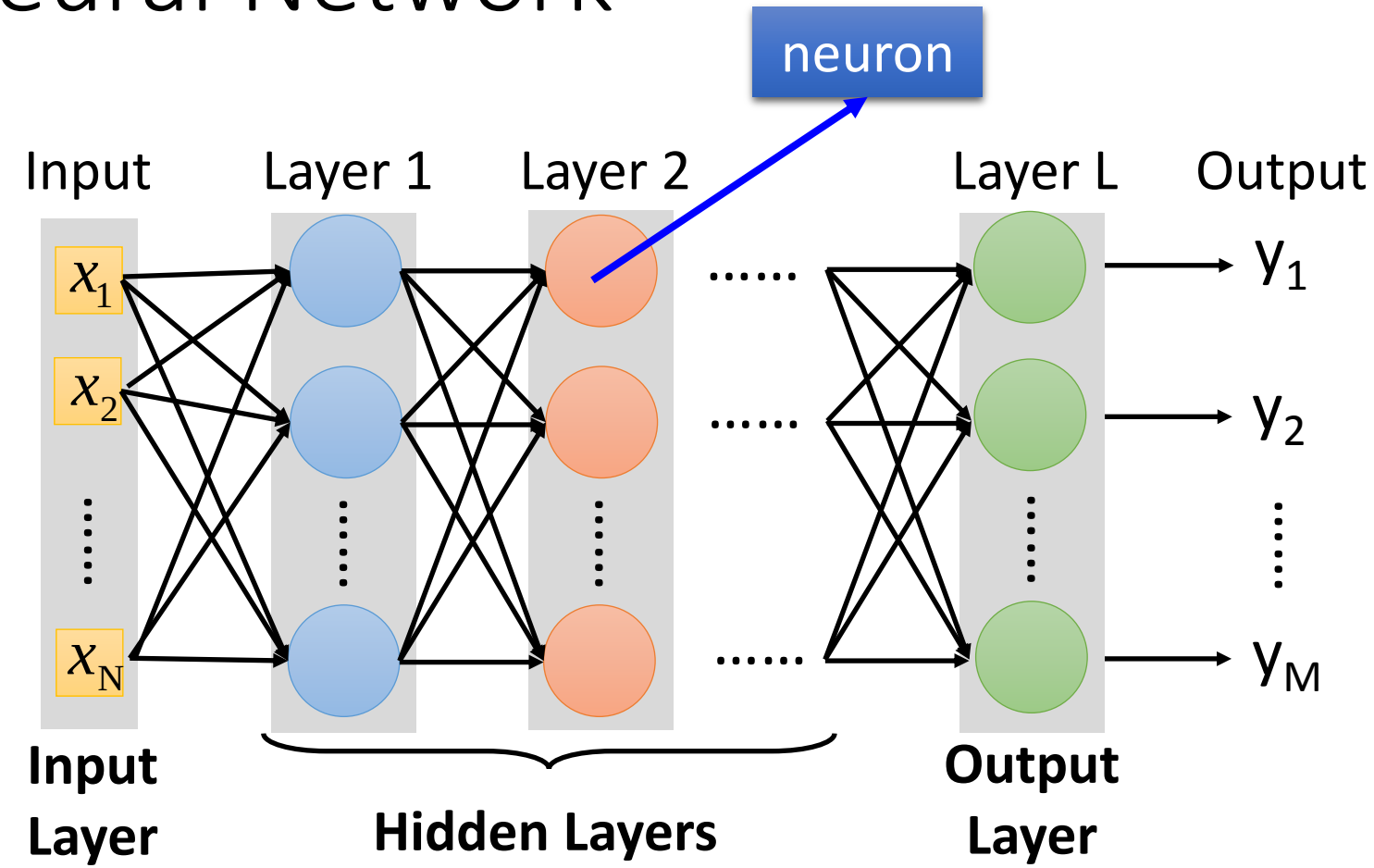
In deep learning, the function f is represented by neural network

Element of Neural Network

Neuron $f: R^K \rightarrow R$

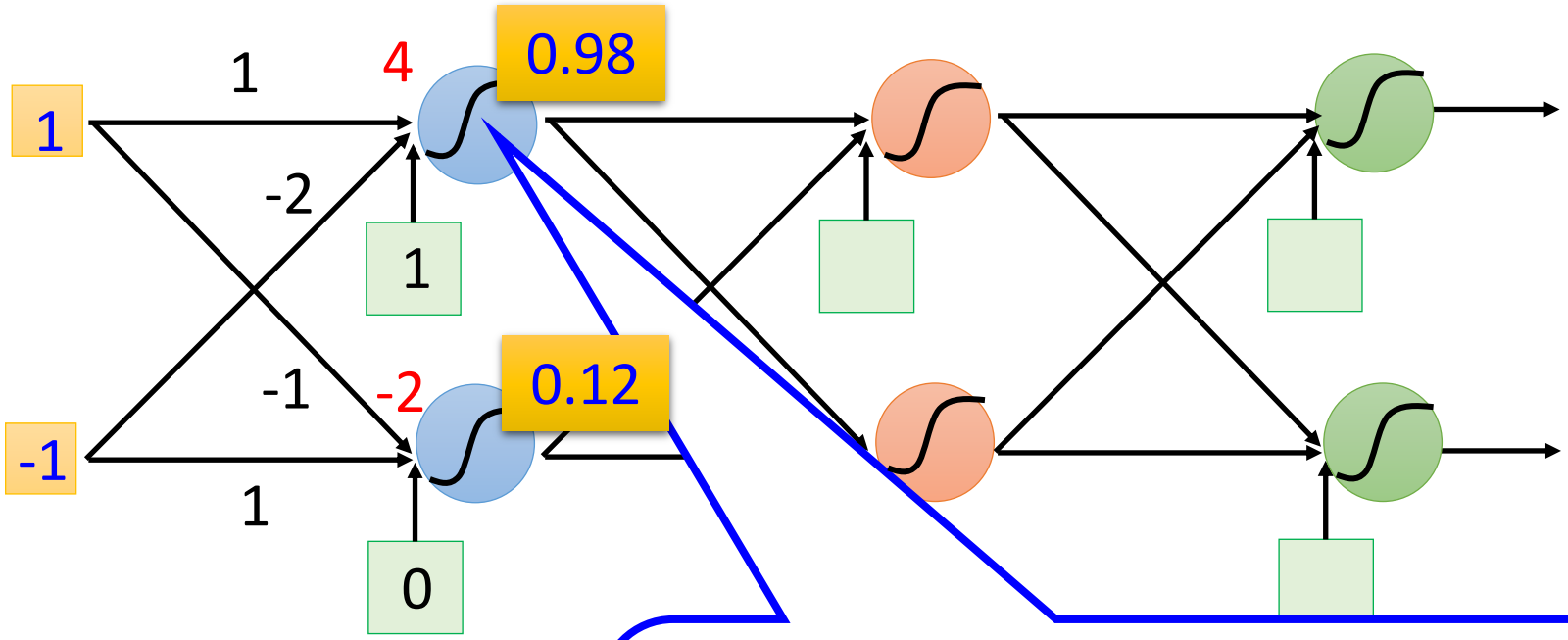


Neural Network



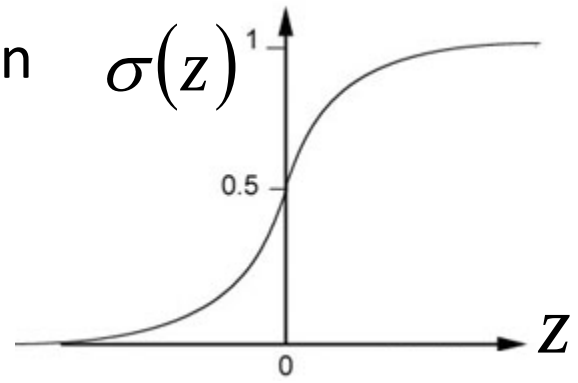
Deep means many hidden layers

Example of Neural Network

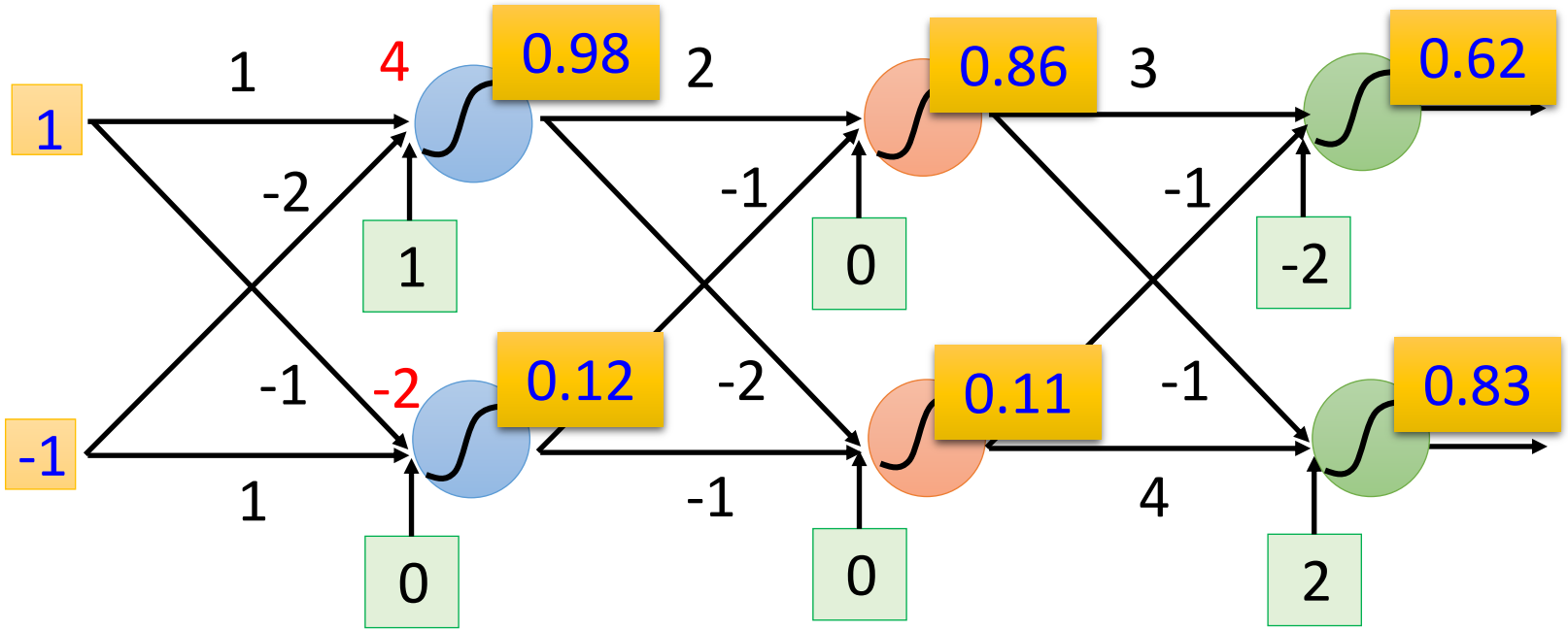


Sigmoid Function

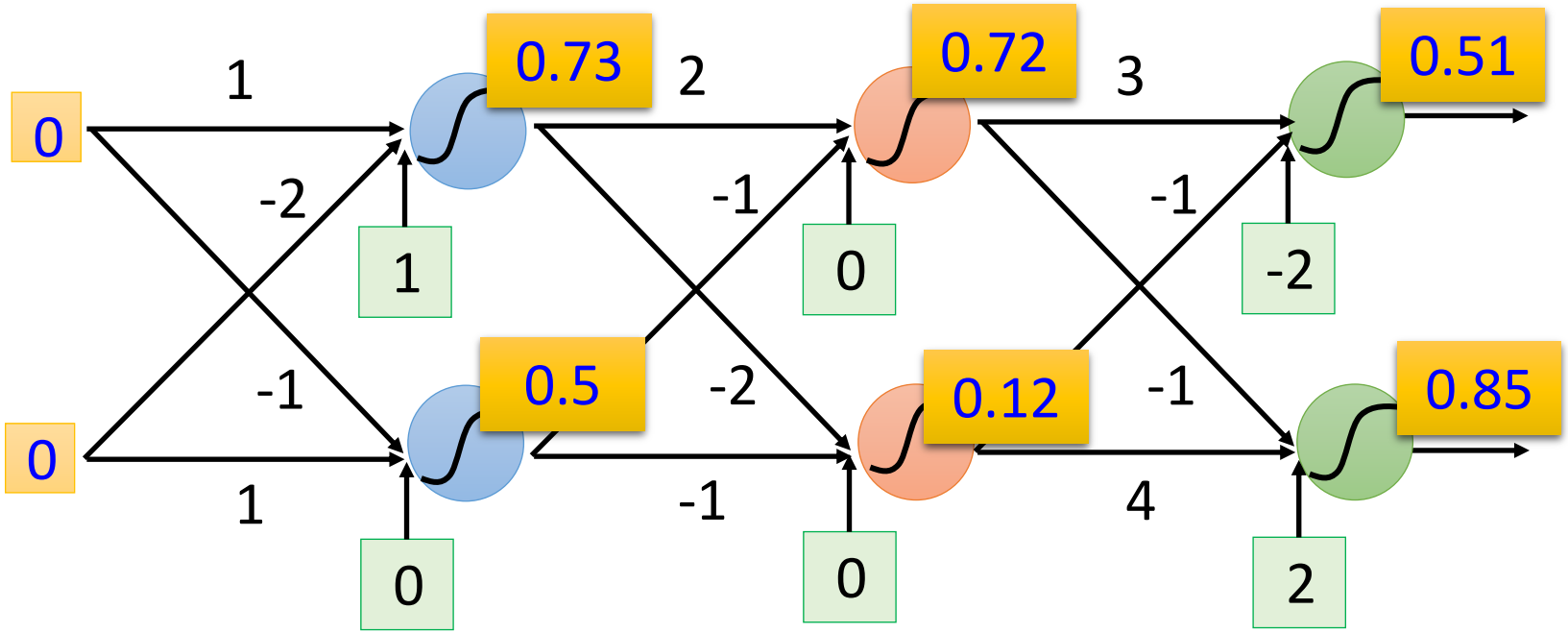
$$\sigma(z) = \frac{1}{1+e^{-z}}$$



Example of Neural Network



Example of Neural Network

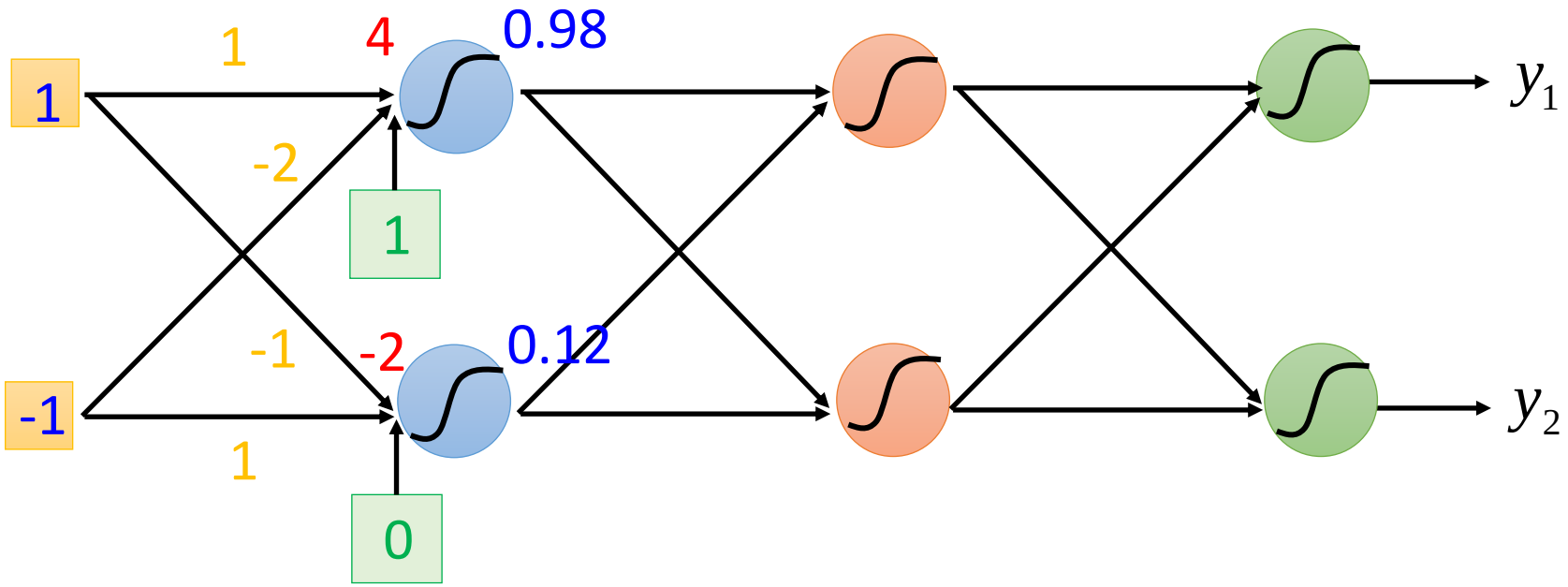


$f: R^2 \rightarrow R^2$

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

Different parameters define different function

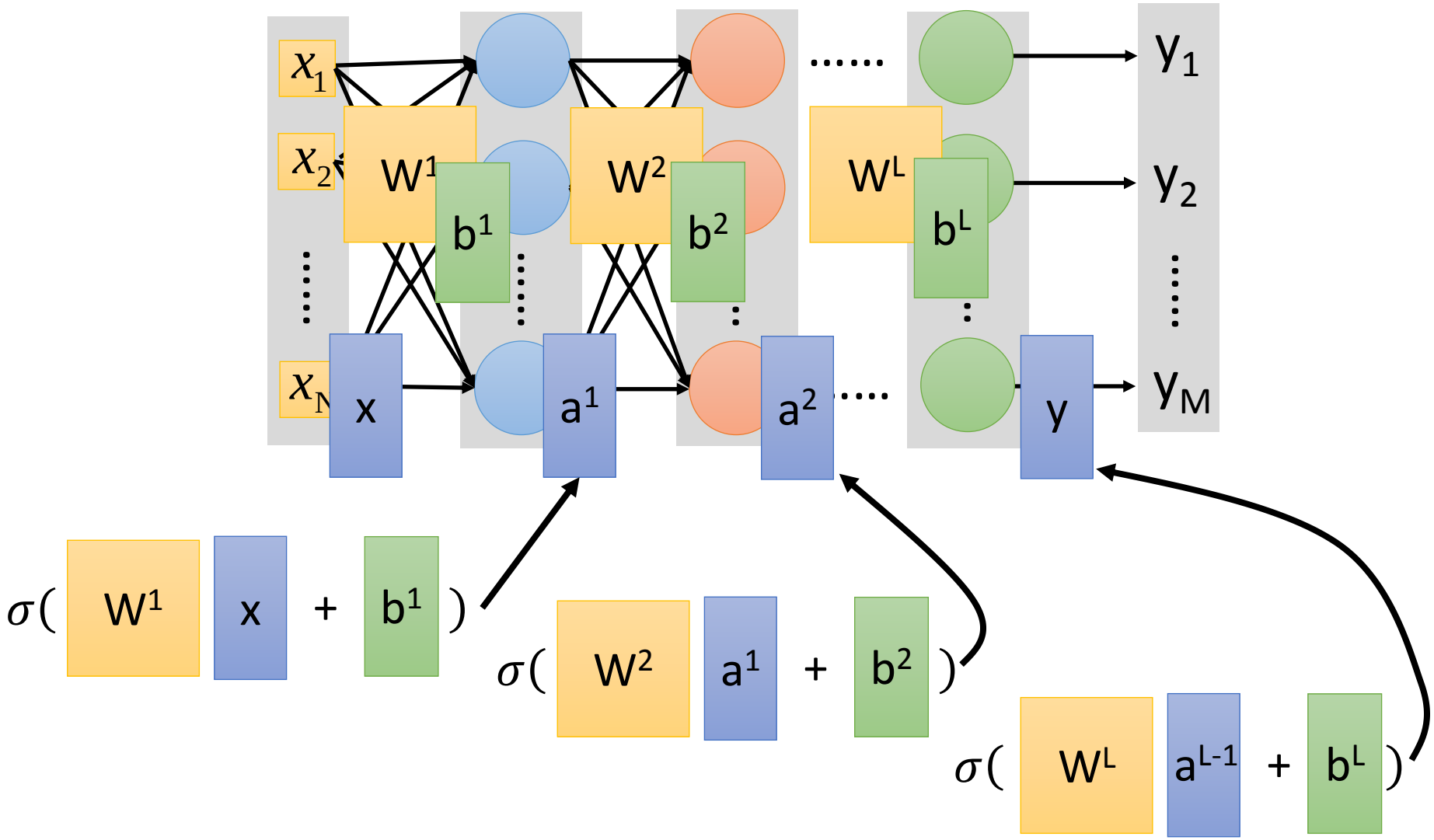
Matrix Operation



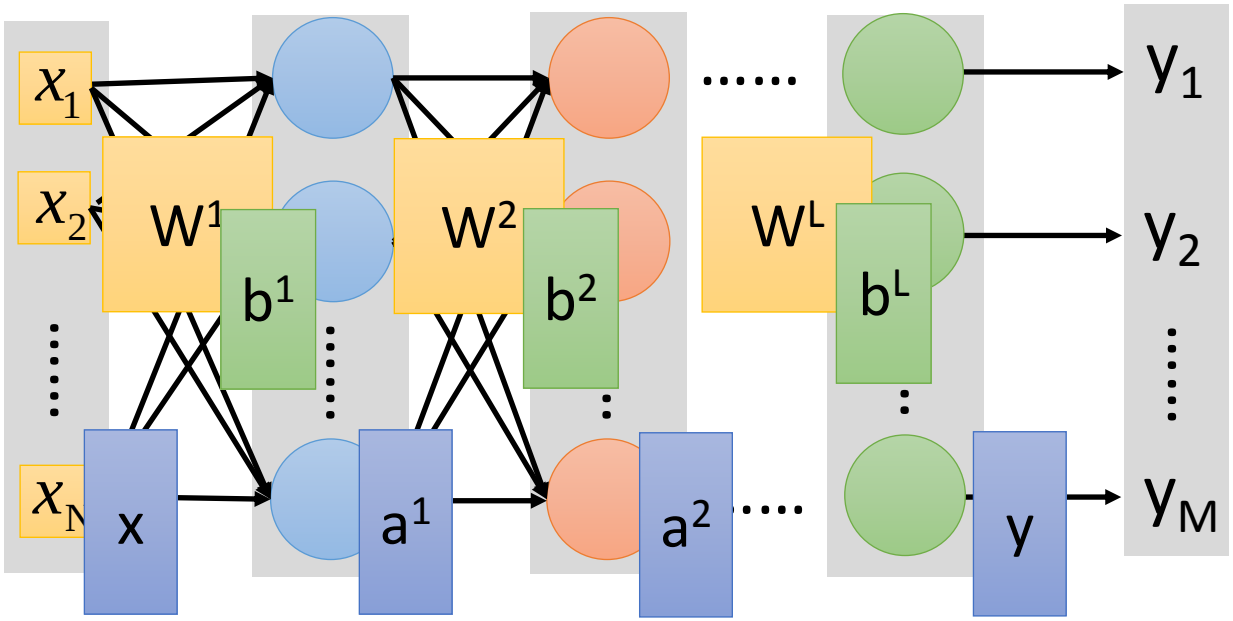
$$\sigma\left(\begin{bmatrix}1 & -2\\ -1 & 1\end{bmatrix}\begin{bmatrix}1\\ -1\end{bmatrix}+\begin{bmatrix}1\\ 0\end{bmatrix}\right)=\begin{bmatrix}0.98\\ 0.12\end{bmatrix}$$

$$\begin{bmatrix}4\\ -2\end{bmatrix}$$

Neural Network



Neural Network



$$\mathbf{y} = f(\mathbf{x})$$

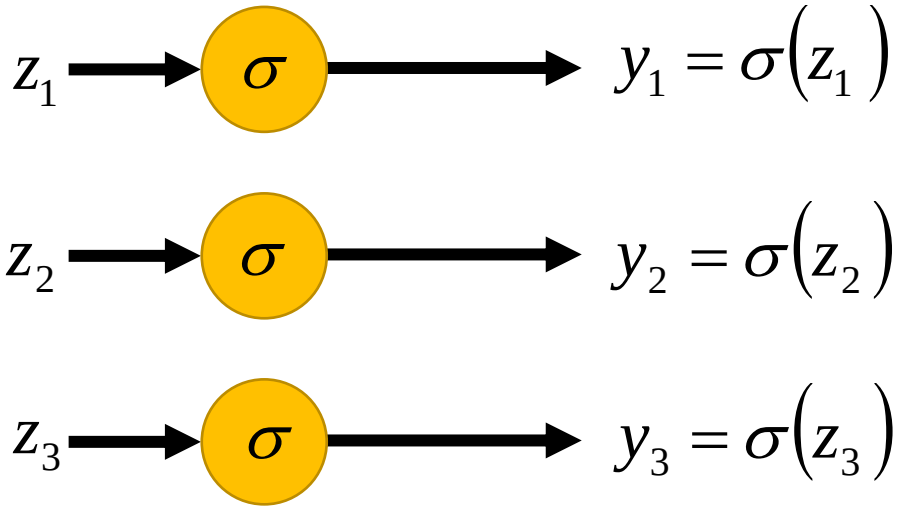
Using parallel computing techniques to speed up matrix operation

$$= \sigma(\mathbf{W}^L \cdots \sigma(\mathbf{W}^2 \sigma(\mathbf{W}^1 \mathbf{x} + \mathbf{b}^1) + \mathbf{b}^2) \cdots + \mathbf{b}^L)$$

Softmax

- Softmax layer as the output layer

Ordinary Layer



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

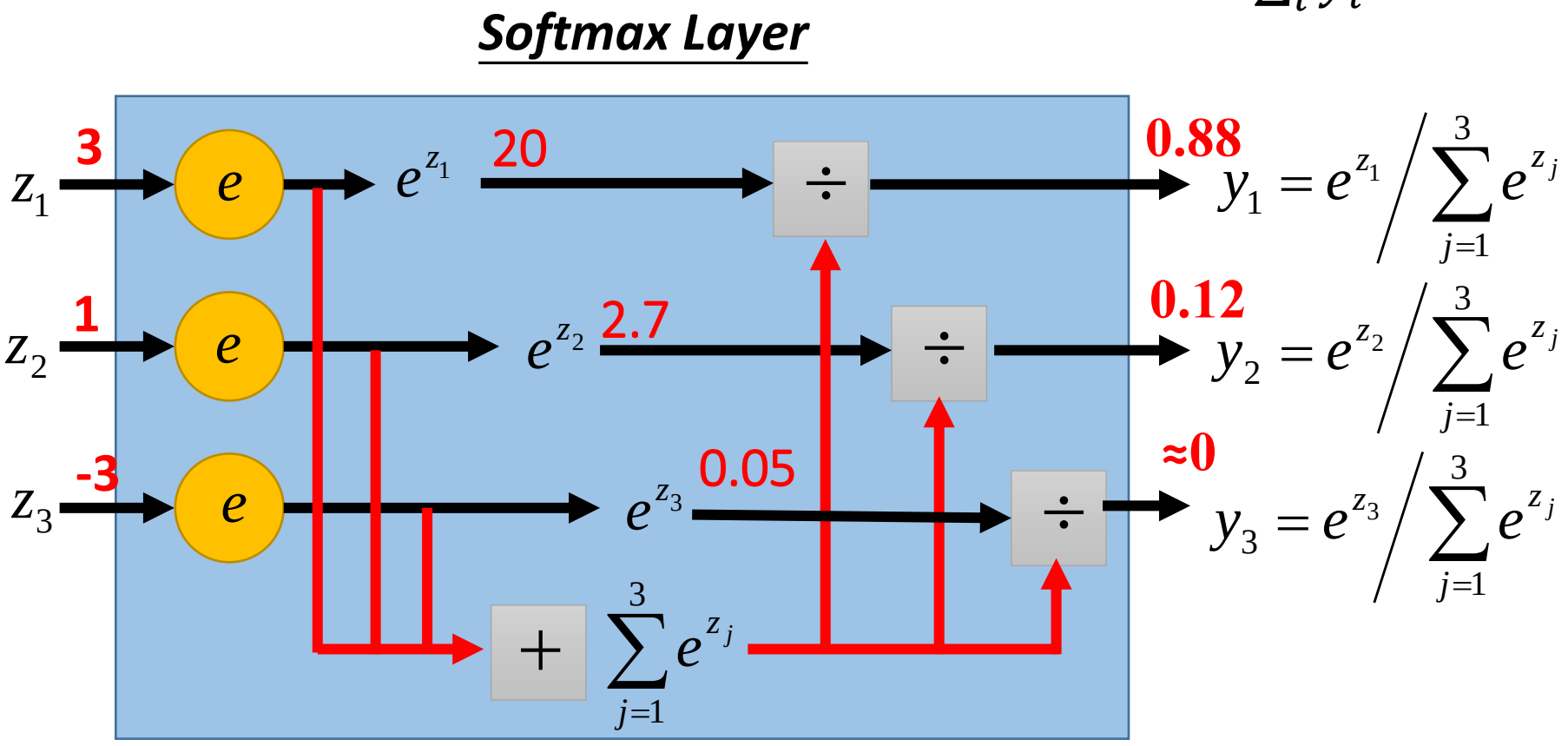
May not be easy to interpret

Softmax

- Softmax layer as the output layer

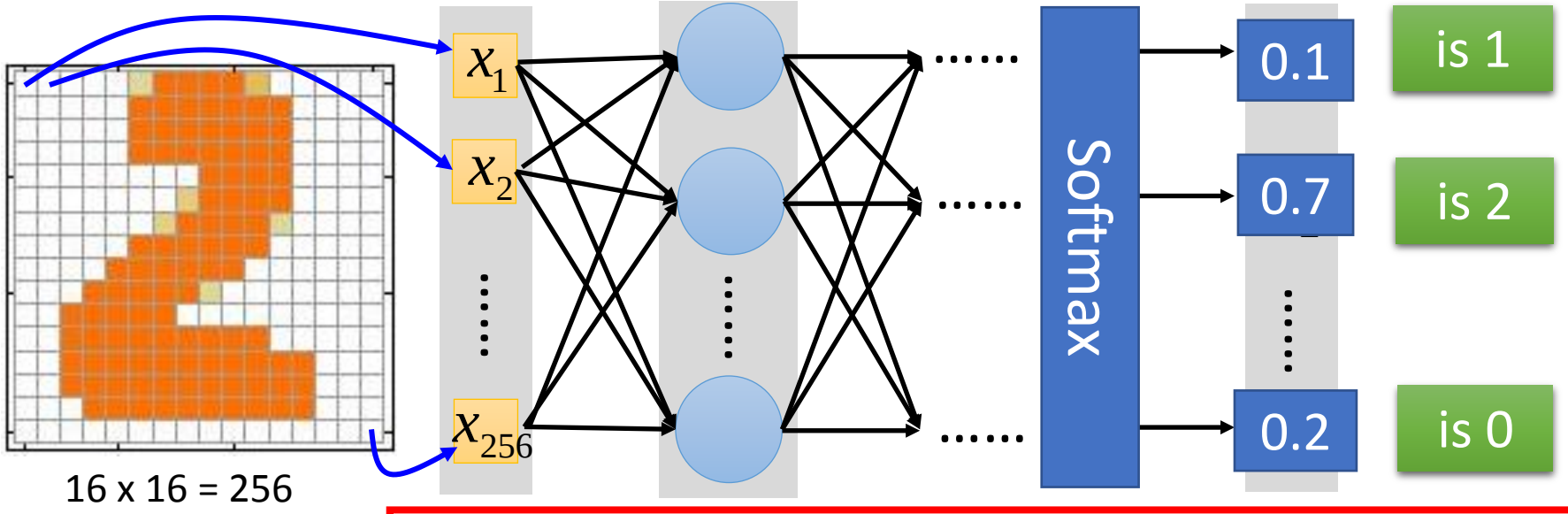
Probability:

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



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 ……

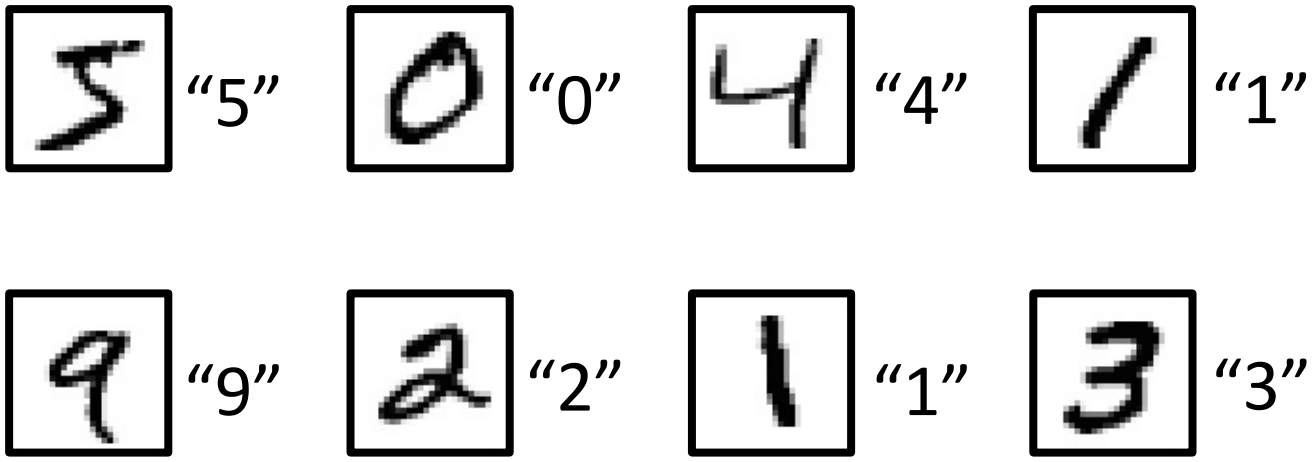
Input: m value

Input: y_2 has the maximum value

How to let the neural network achieve this

Training Data

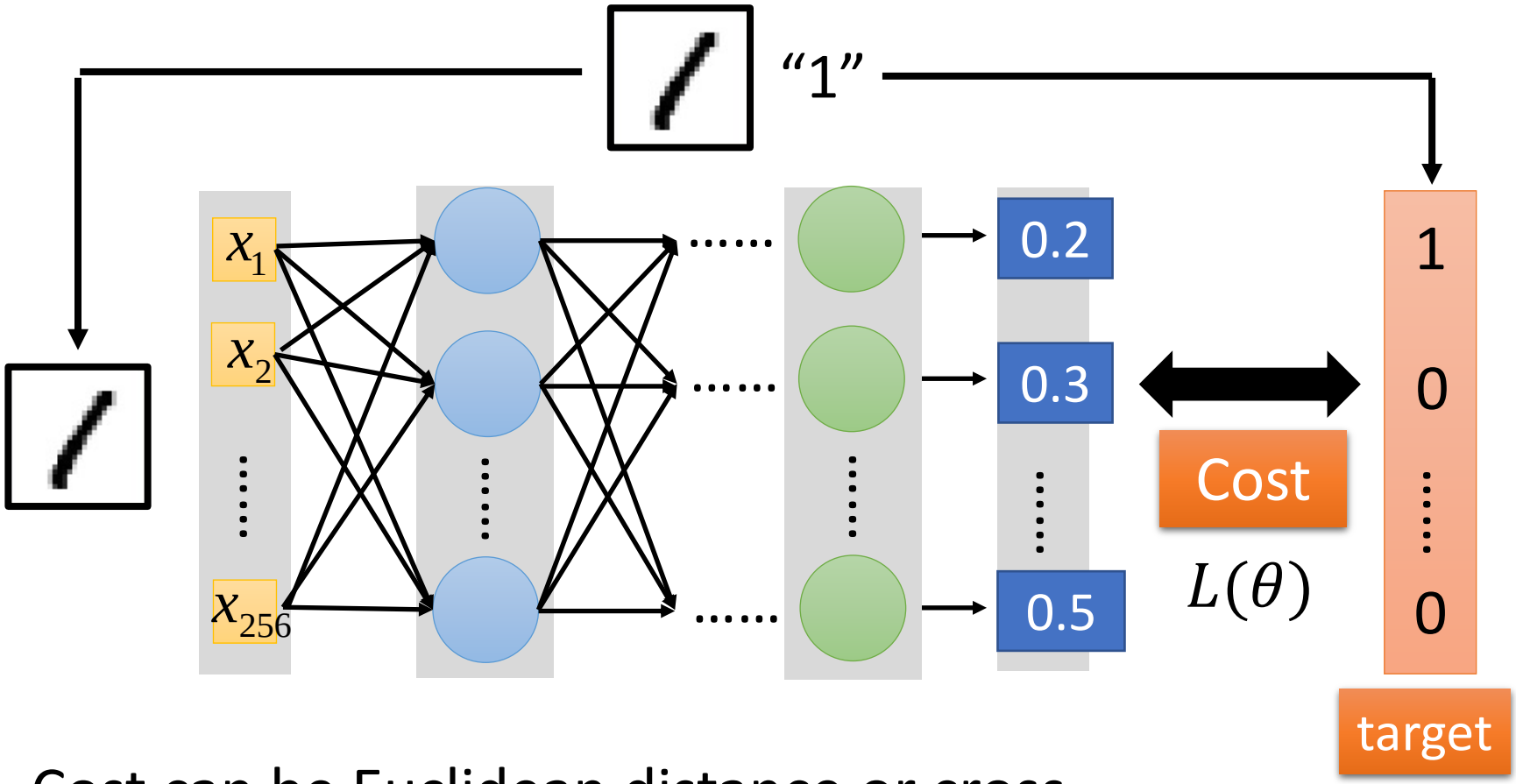
- Preparing training data: images and their labels



Using the training data to find the network parameters.

Cost

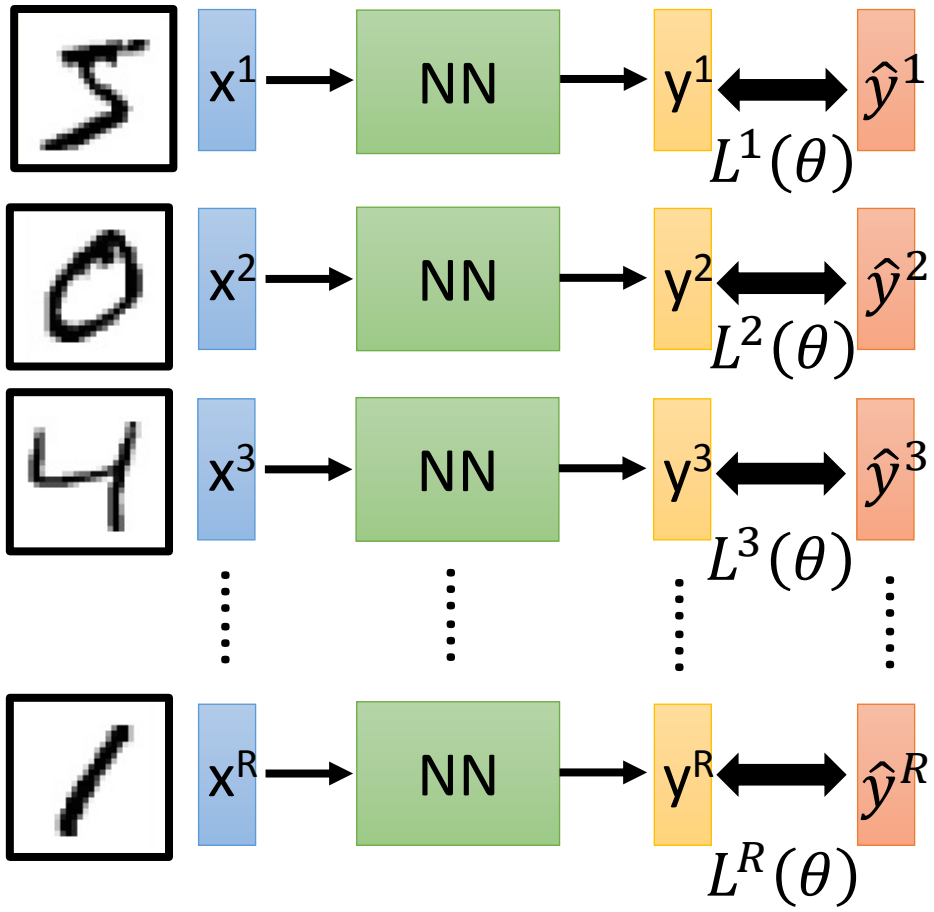
Given a set of network parameters θ, each example has a cost value.



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

Total Cost

For all training data …



Total Cost:

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

How bad the network parameters θ is on this task

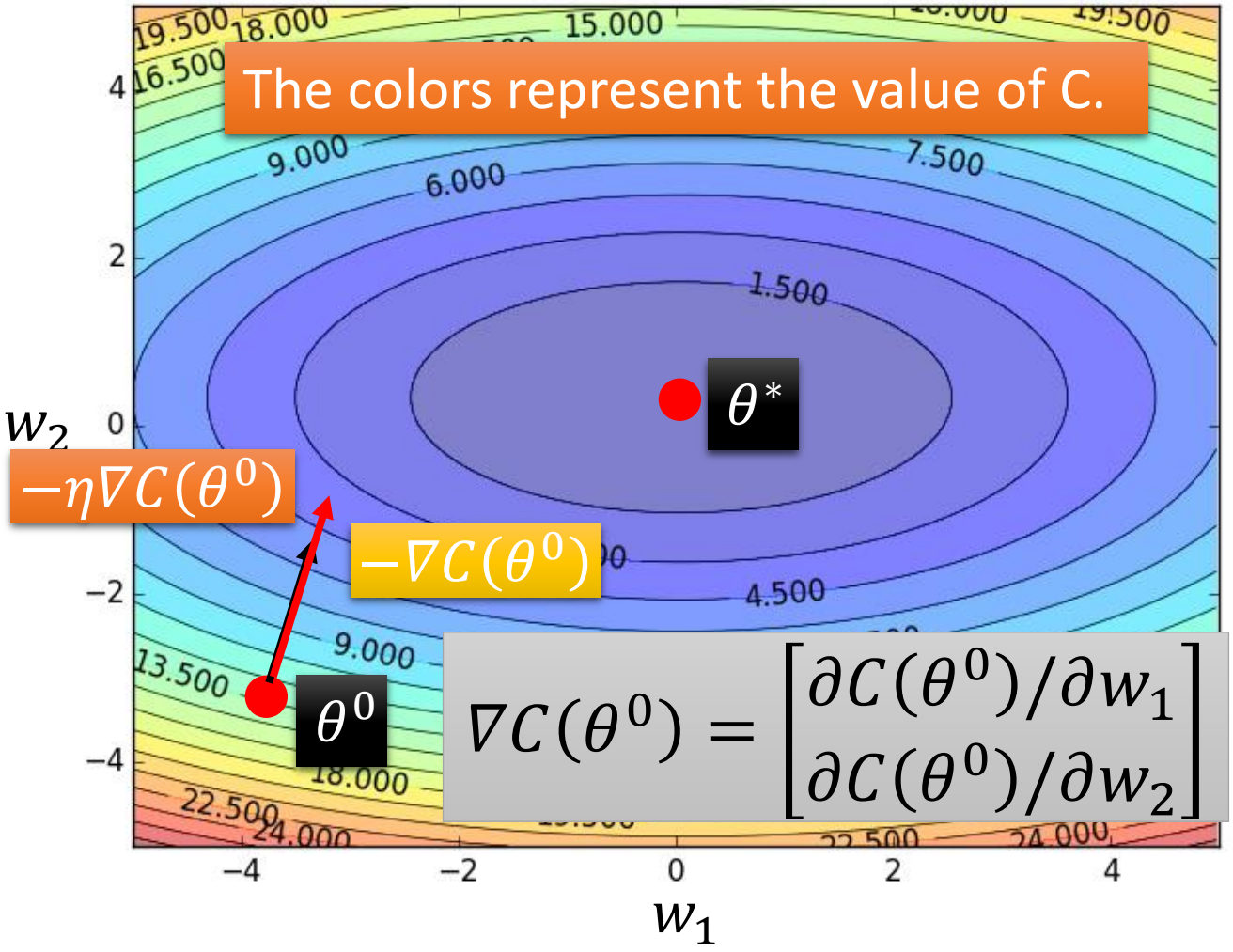
Find the network parameters θ^* that minimize this value

Gradient Descent

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

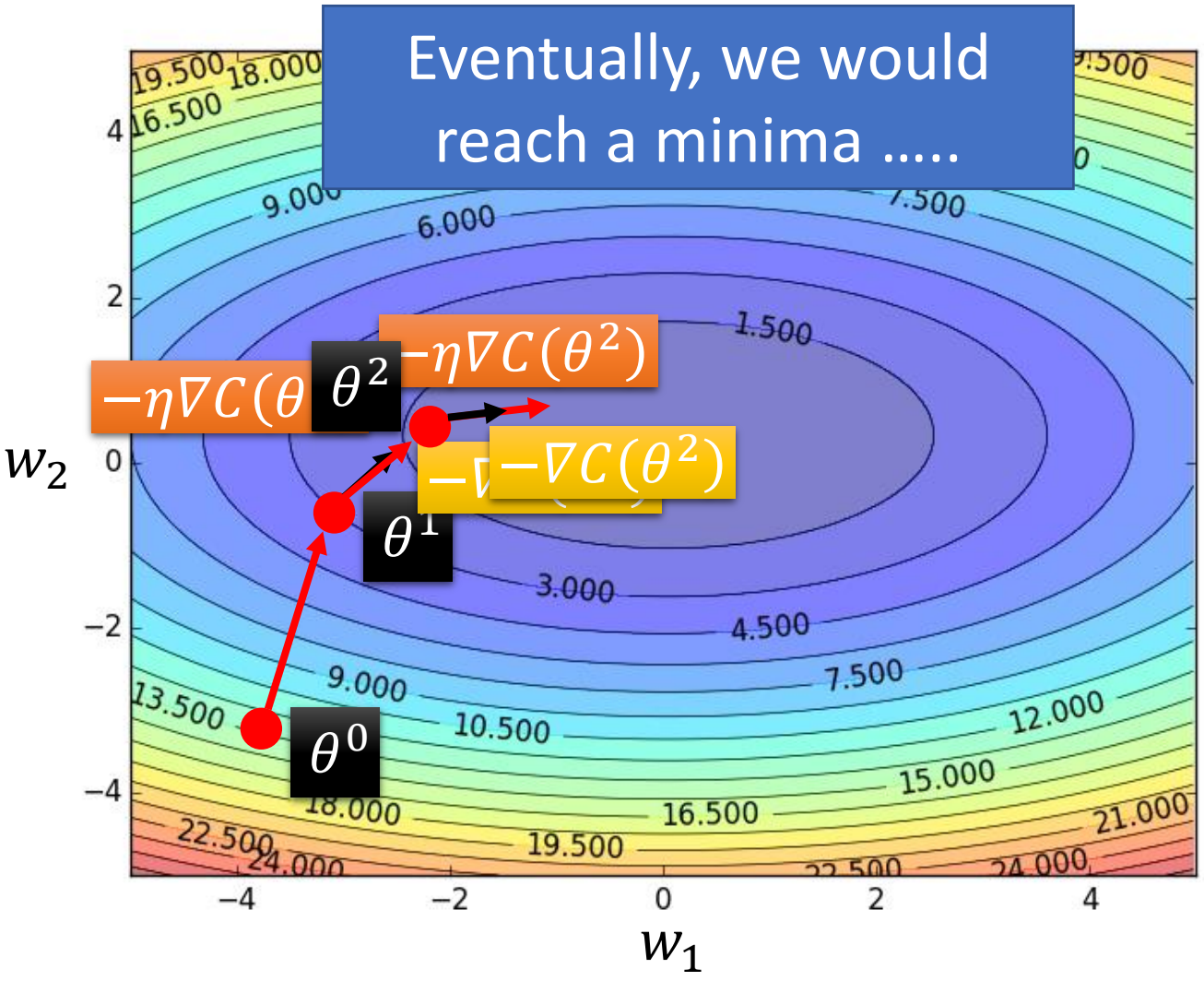
Compute the negative gradient at θ^0

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

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

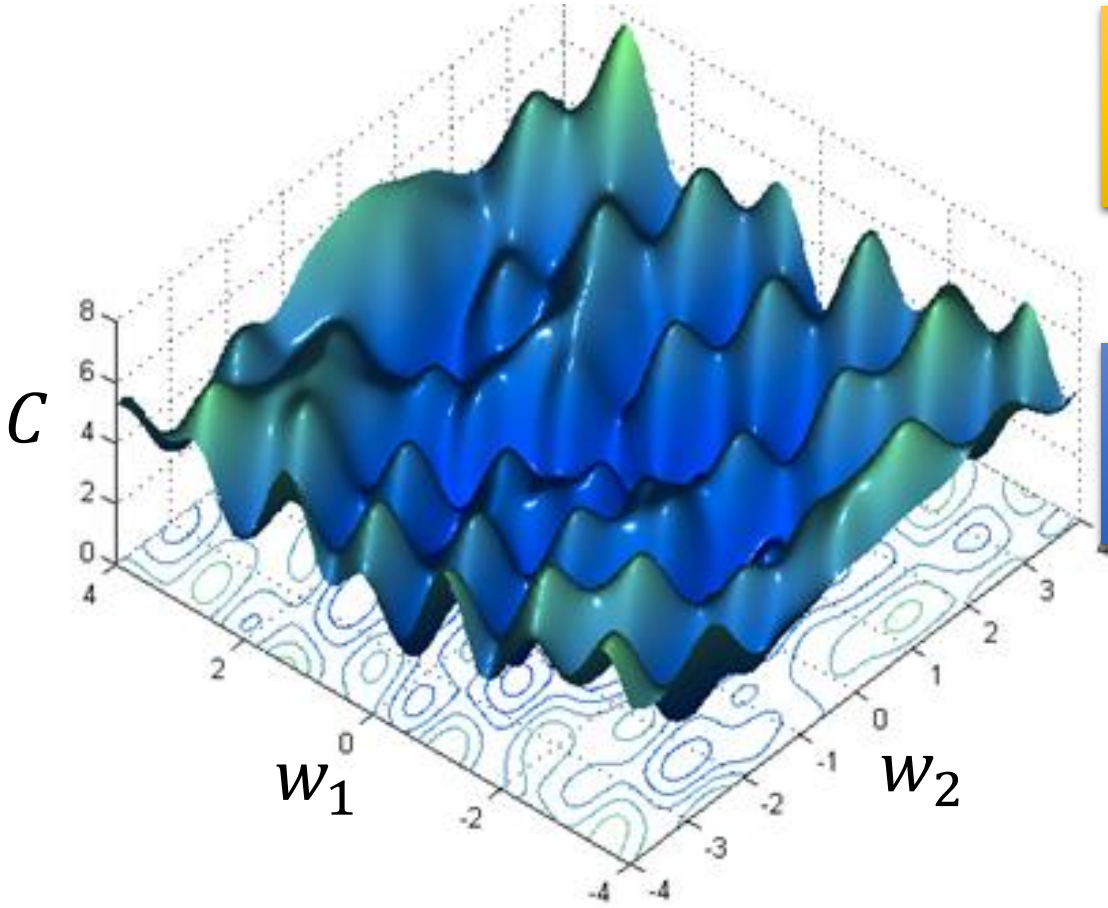
$\rightarrow -\nabla C(\theta^0)$

Times the learning rate η

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

Local Minima

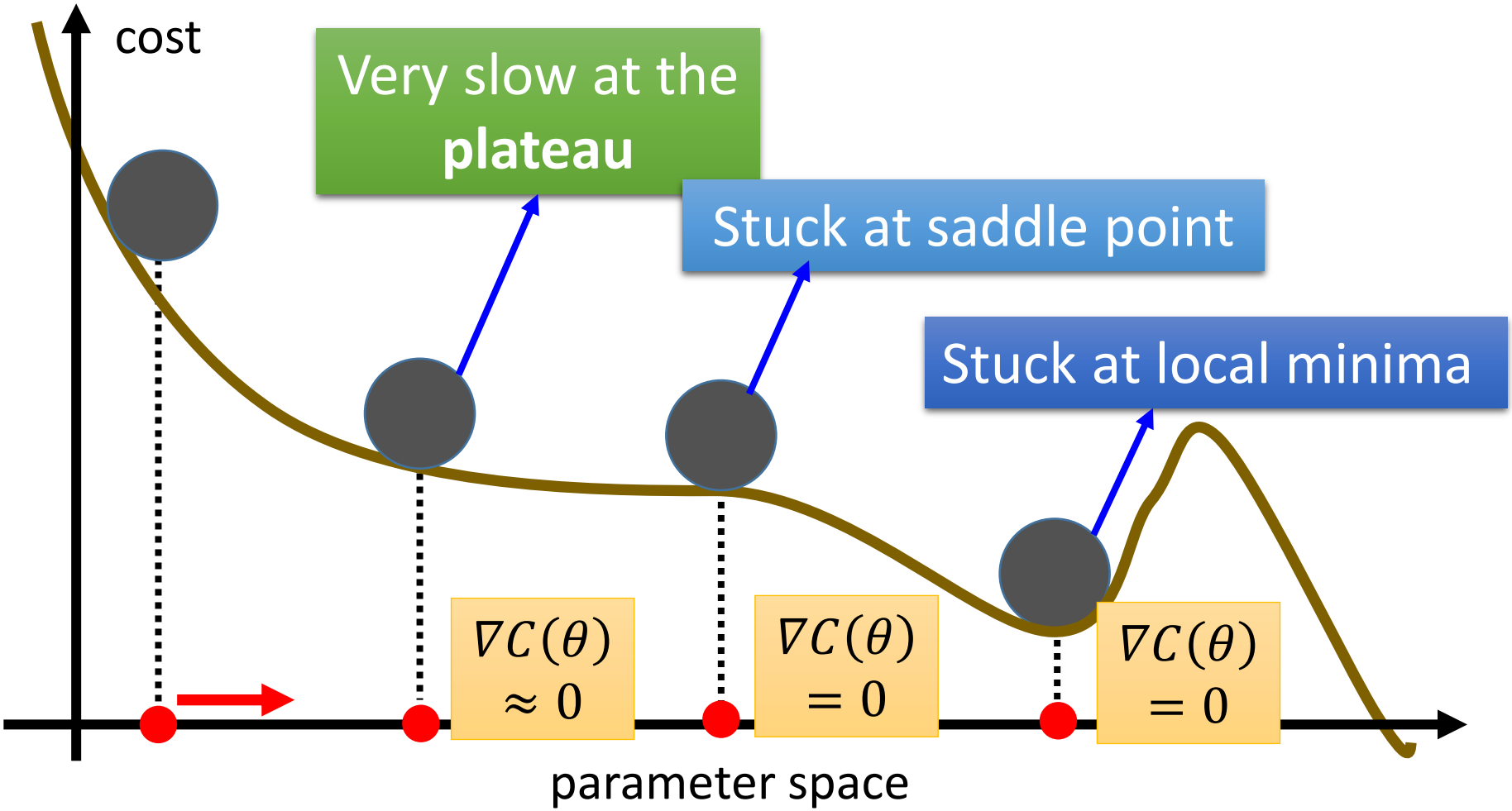
- Gradient descent never guarantee global minima



Different initial point θ^0

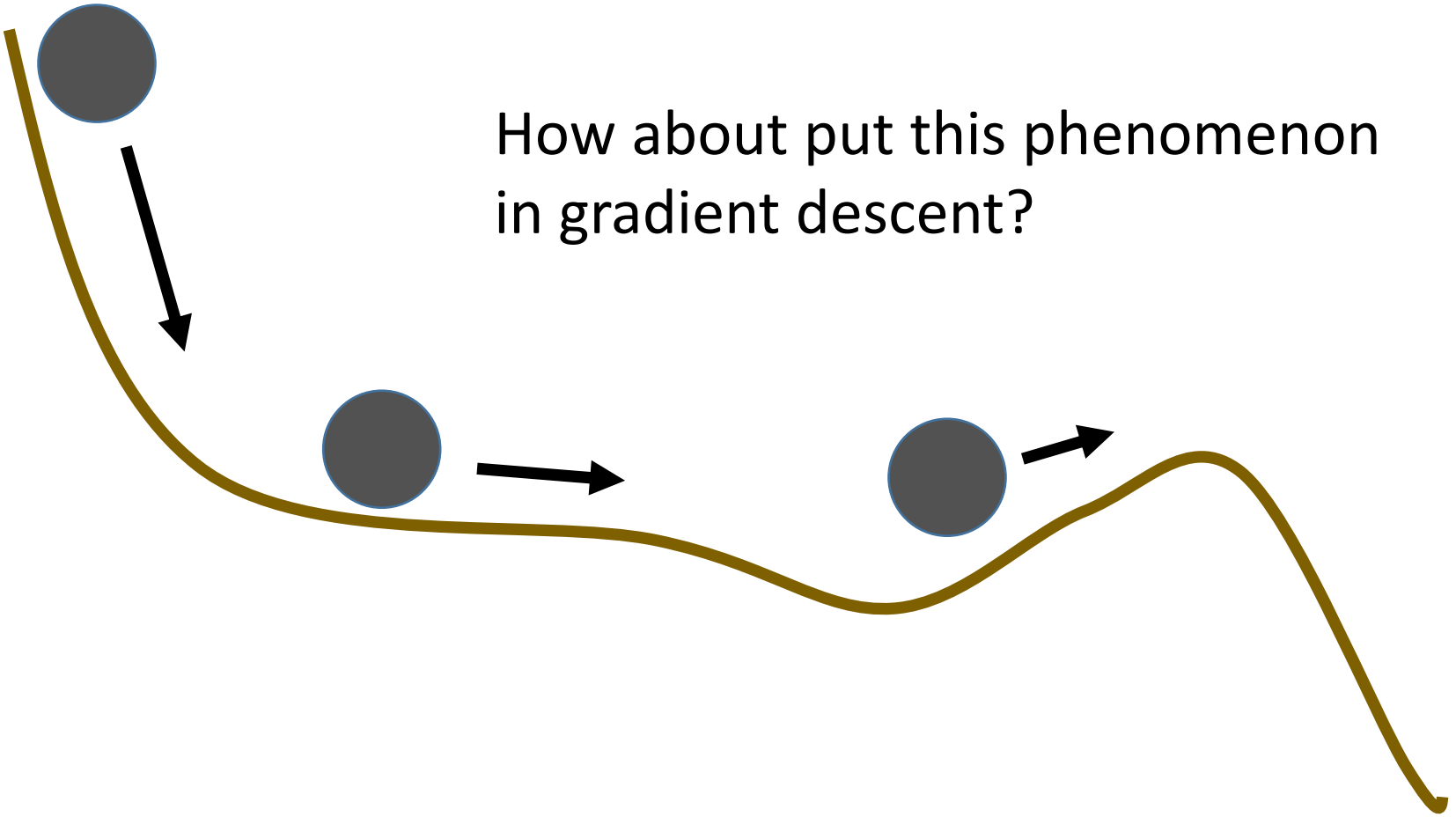
Reach different minima, so different results

Besides local minima ……



In physical world ……

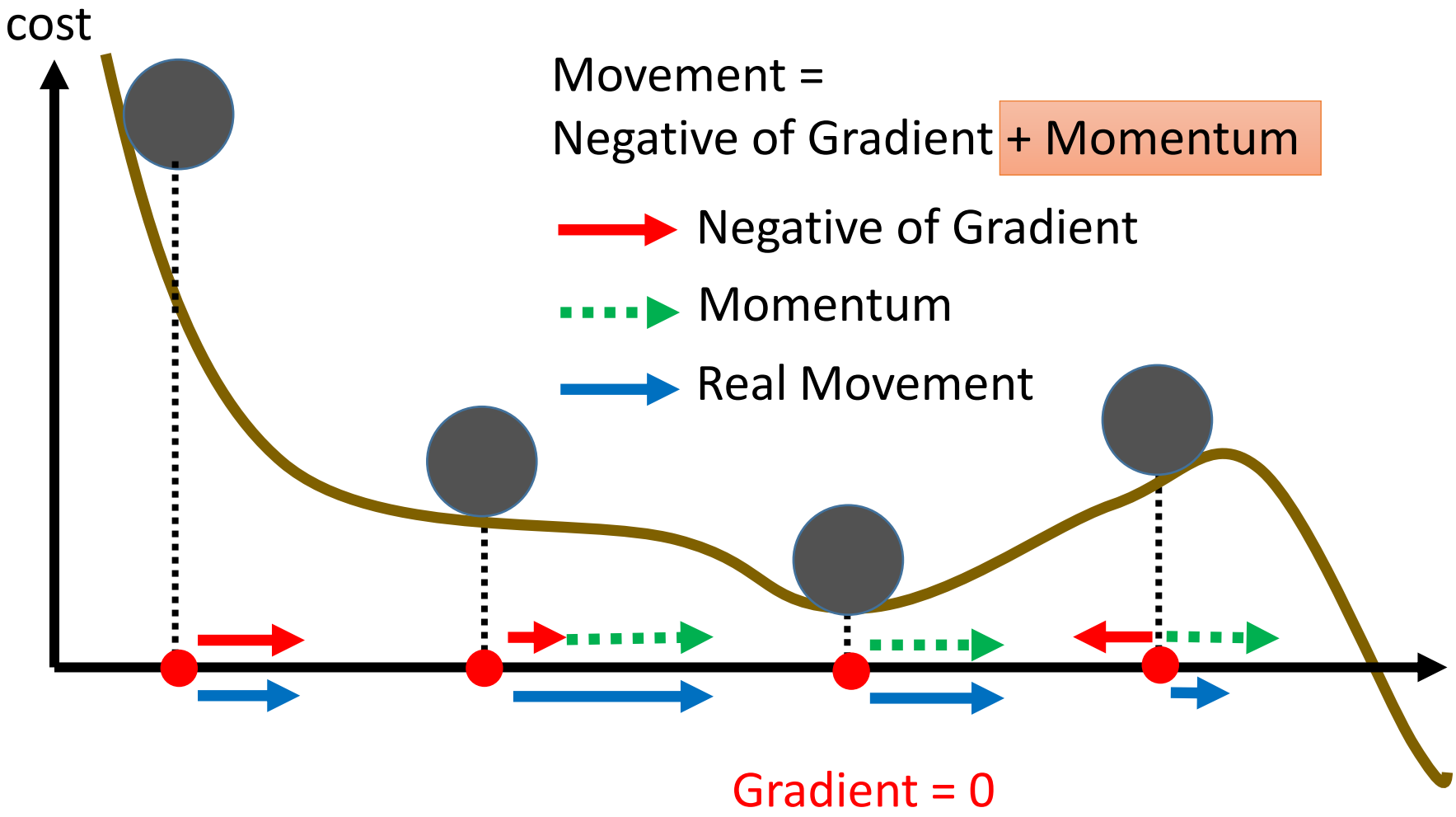
- Momentum



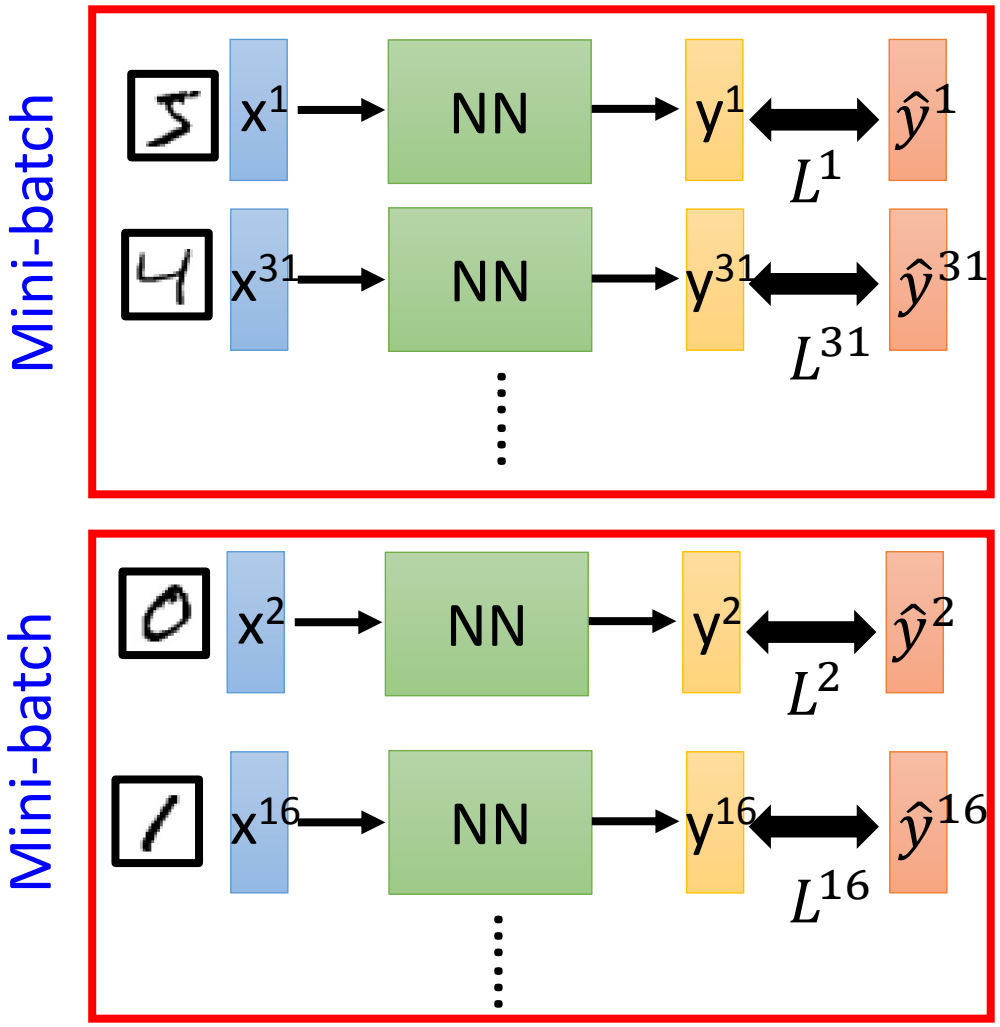
How about put this phenomenon in gradient descent?

Momentum

Still not guarantee reaching global minima, but give some hope ……



Mini-batch



- Randomly initialize θ^0
- Pick the 1st batch

$$C = L^1 + L^{31} + \cdots$$

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

- Pick the 2nd batch

$$C = L^2 + L^{16} + \cdots$$

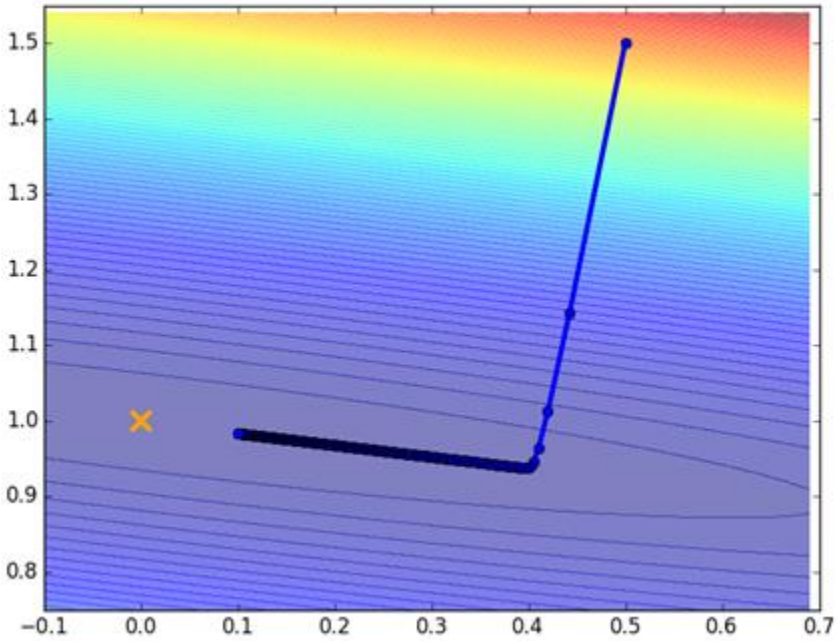
$$\theta^2 \leftarrow \theta^1 - \eta \nabla C(\theta^1)$$

$\vdots$

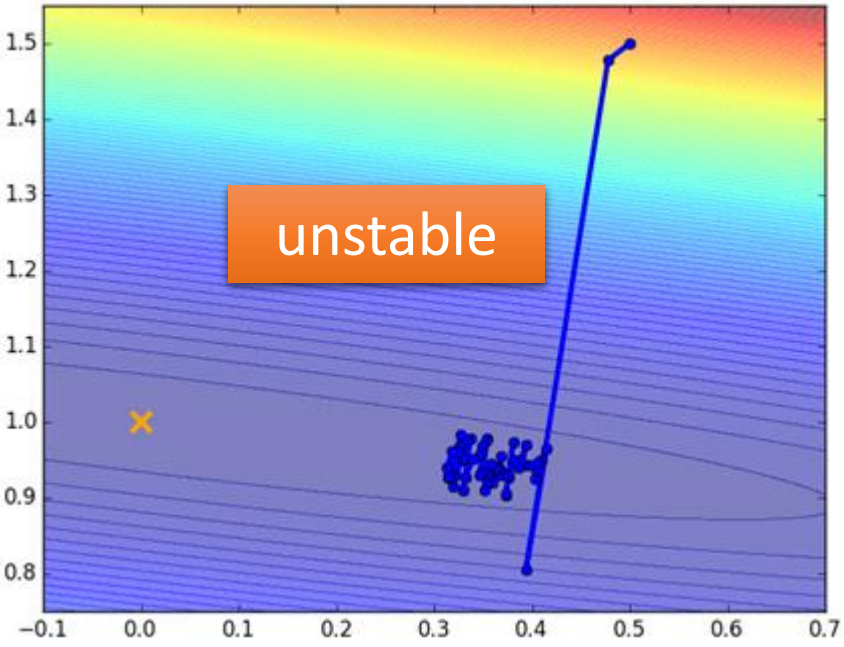
C is different each time when we update parameters!

Mini-batch

Original Gradient Descent



With Mini-batch

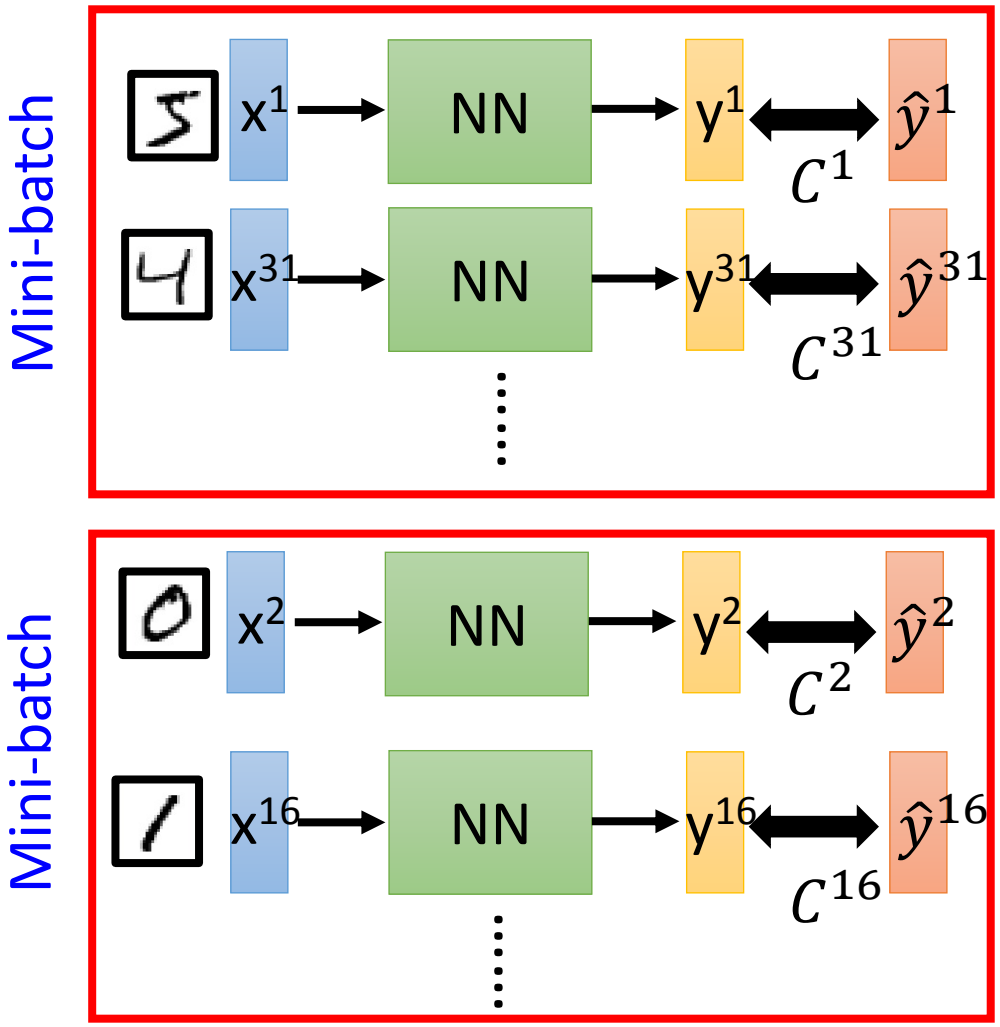


The colors represent the total C on all training data.

Mini-batch

Faster

Better!



- Randomly initialize θ^0
- Pick the 1st batch

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

  $\theta^1 \leftarrow \theta^0 - \eta \nabla C(\theta^0)$
- Pick the 2nd batch

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

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

  ⋮
- 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 (not today)
- Many toolkits can compute the gradients automatically

theano

TensorFlow

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.

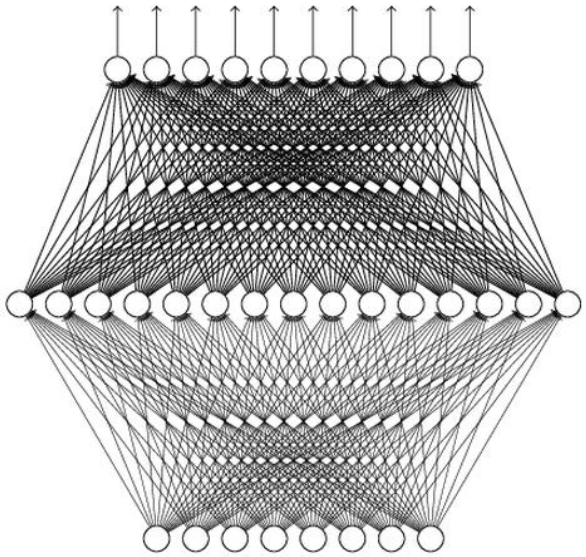
Universality Theorem

Any continuous function f

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

Can be realized by a network with one hidden layer

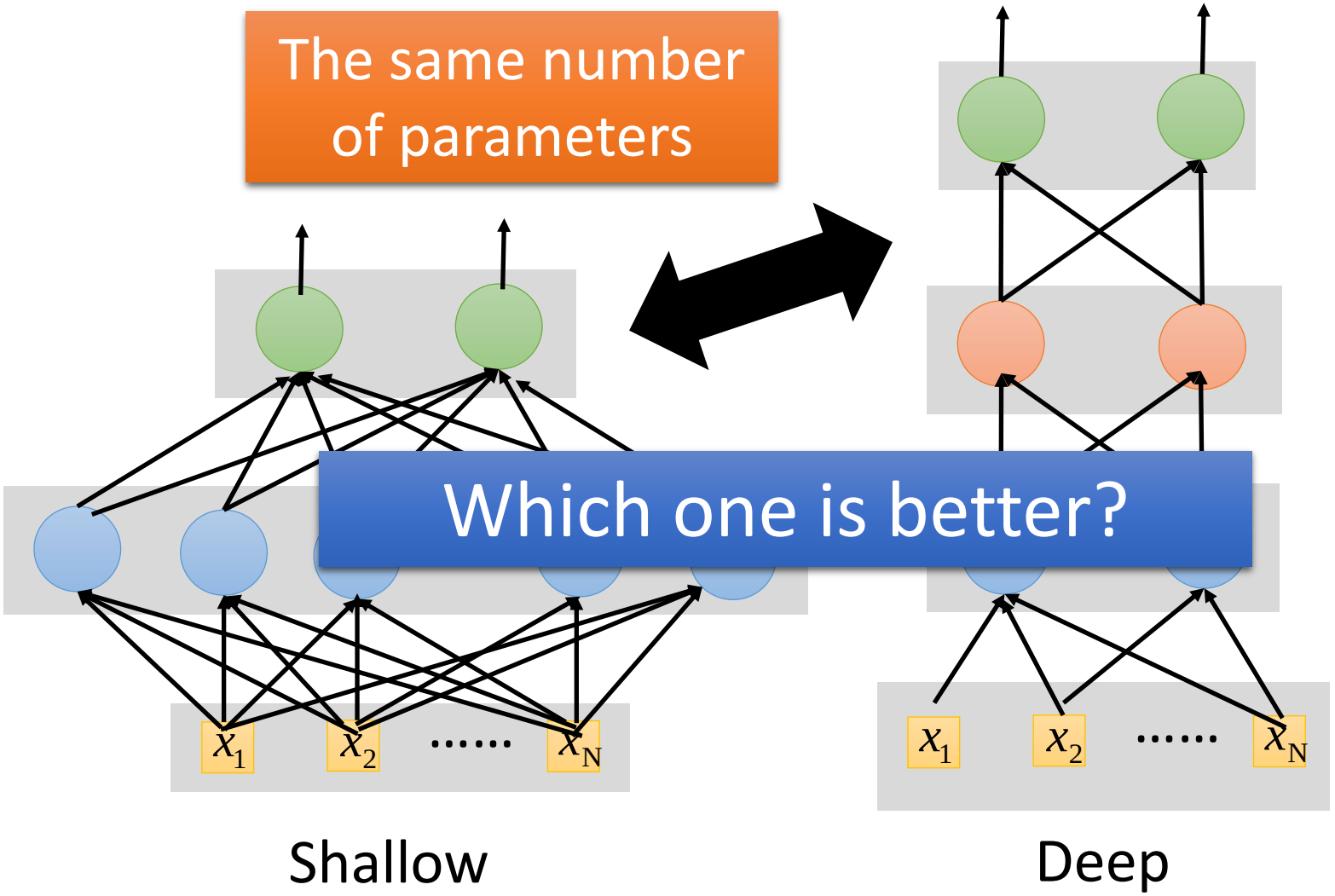
(given **enough** hidden neurons)



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

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

Fat + Short v.s. Thin + Tall



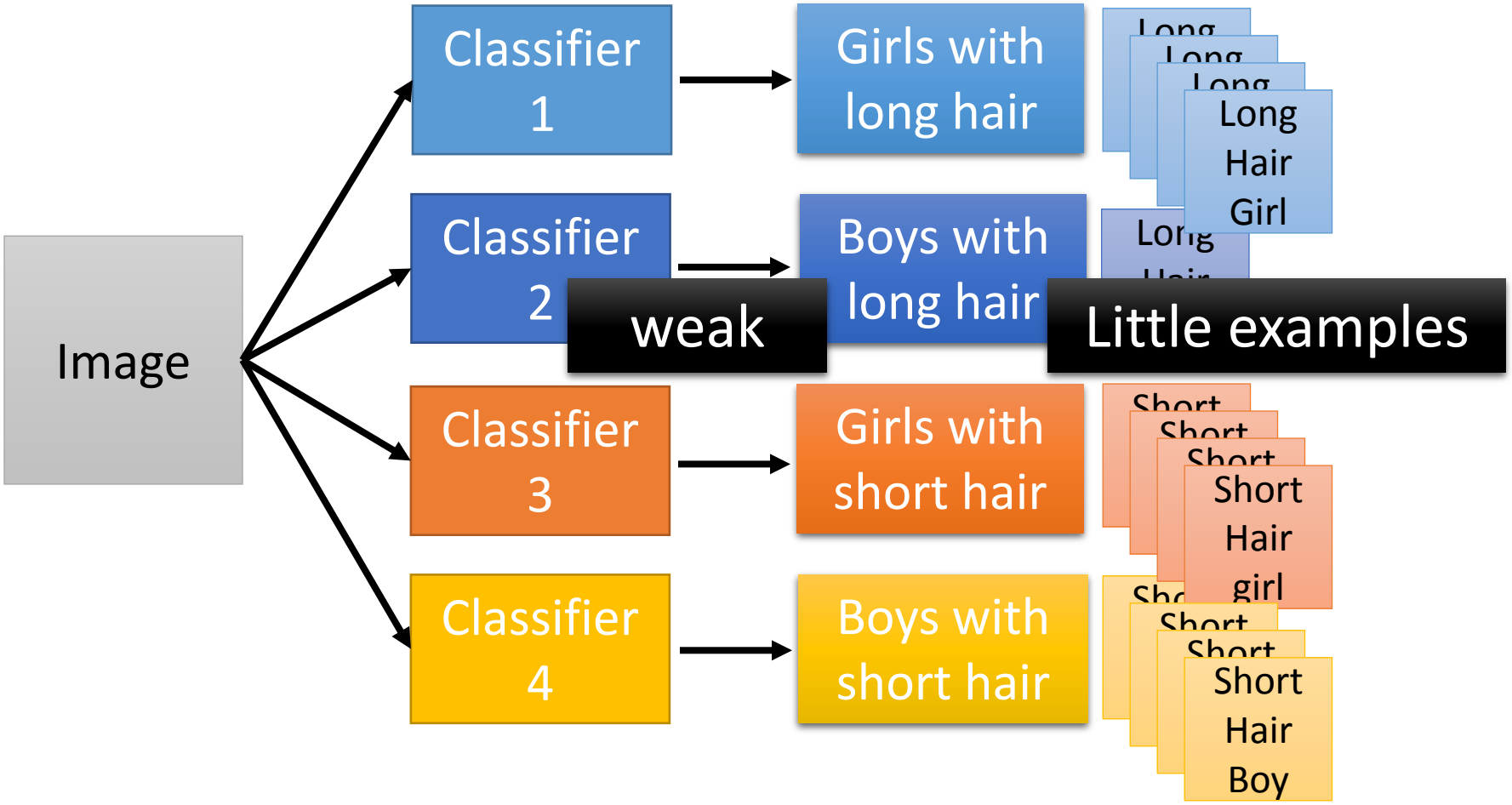
Fat + Short v.s. Thin + Tall

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?

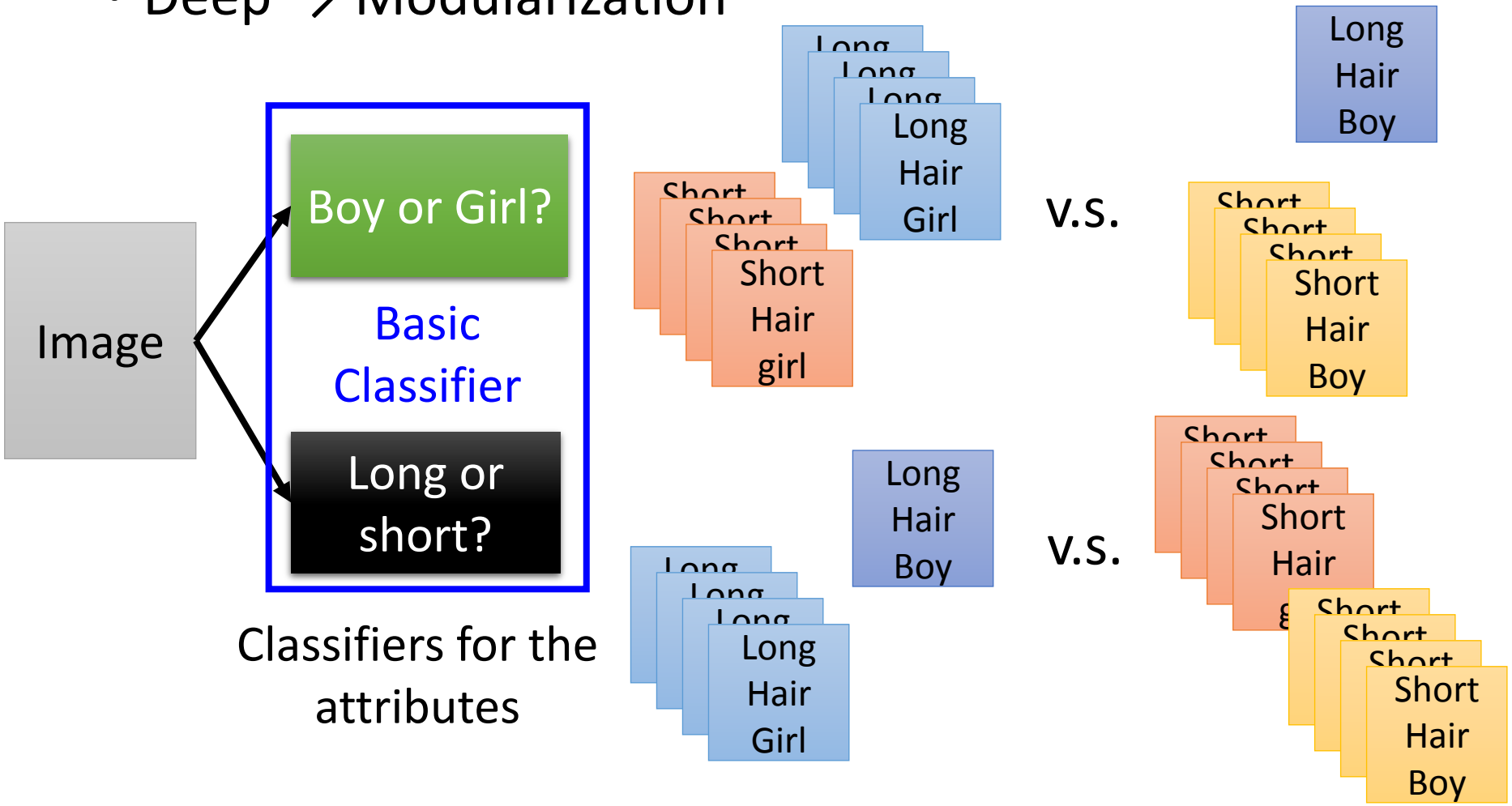
- Deep → Modularization



Why Deep?

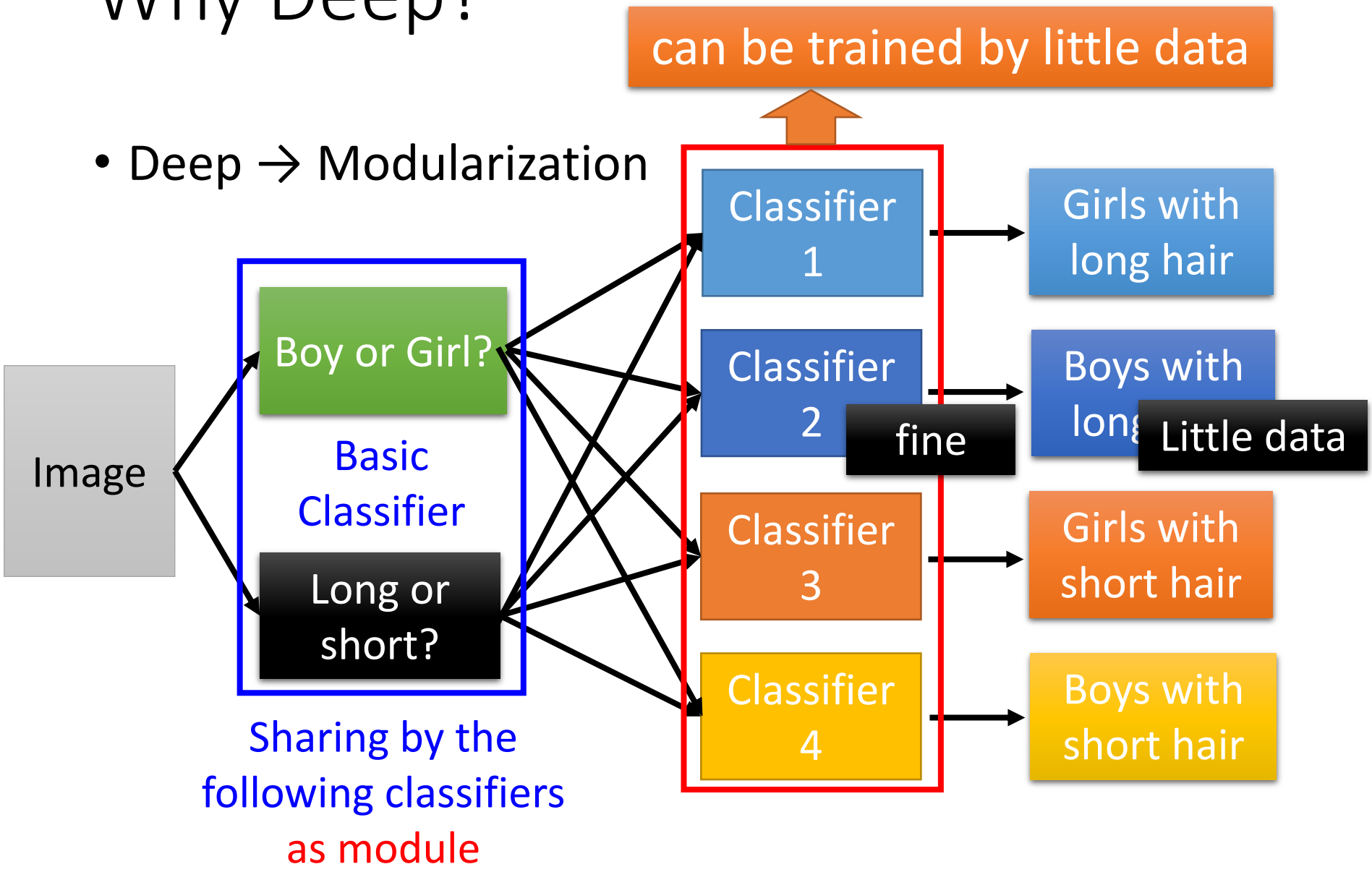
Each basic classifier can have sufficient training examples.

- Deep → Modularization



Why Deep?

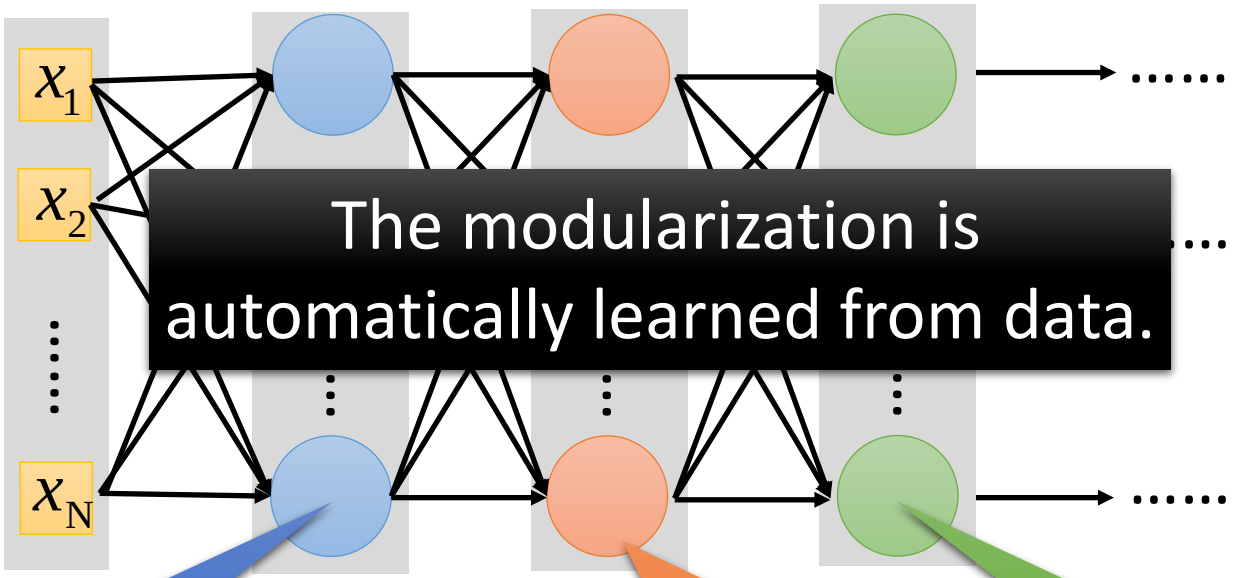
- Deep → Modularization



Why Deep?

Deep Learning also works on small data set like TIMIT.

- Deep → Modularization → Less training data?

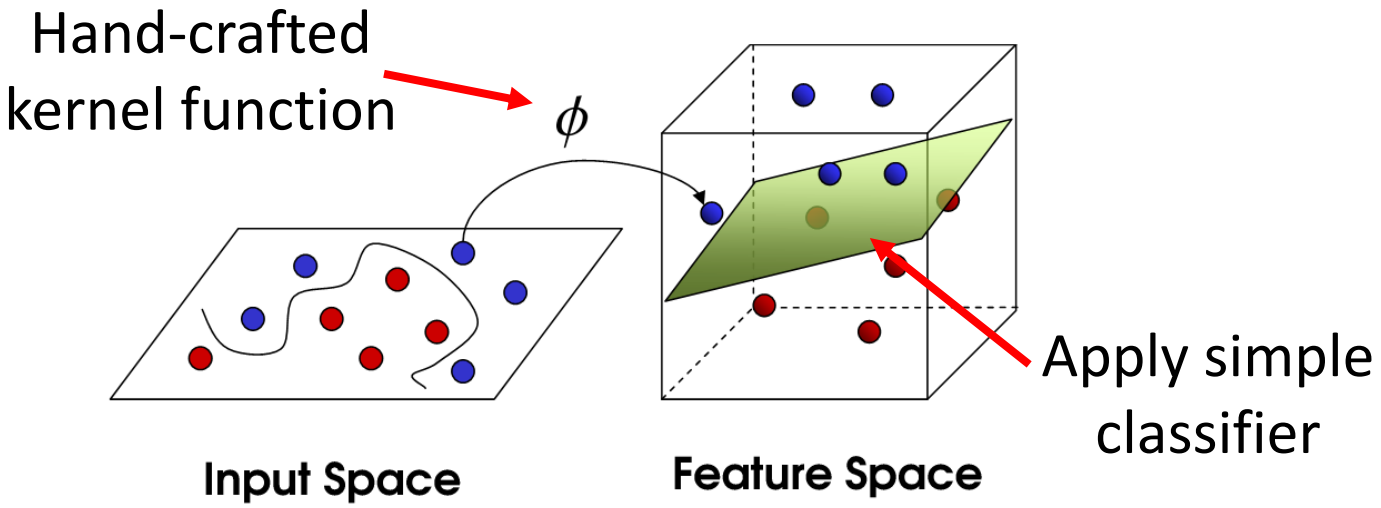


The most basic classifiers

Use 1st layer as module to build classifiers

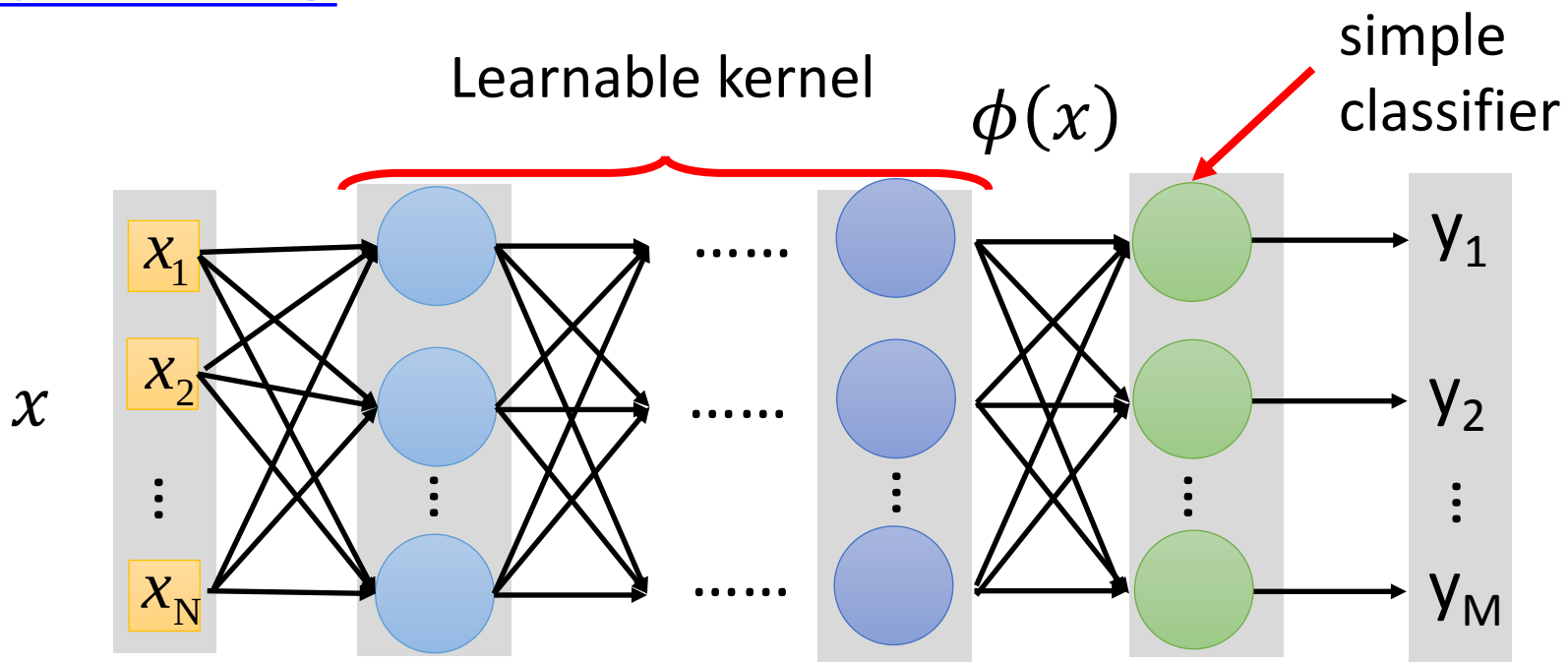
Use 2nd layer as module .....

SVM

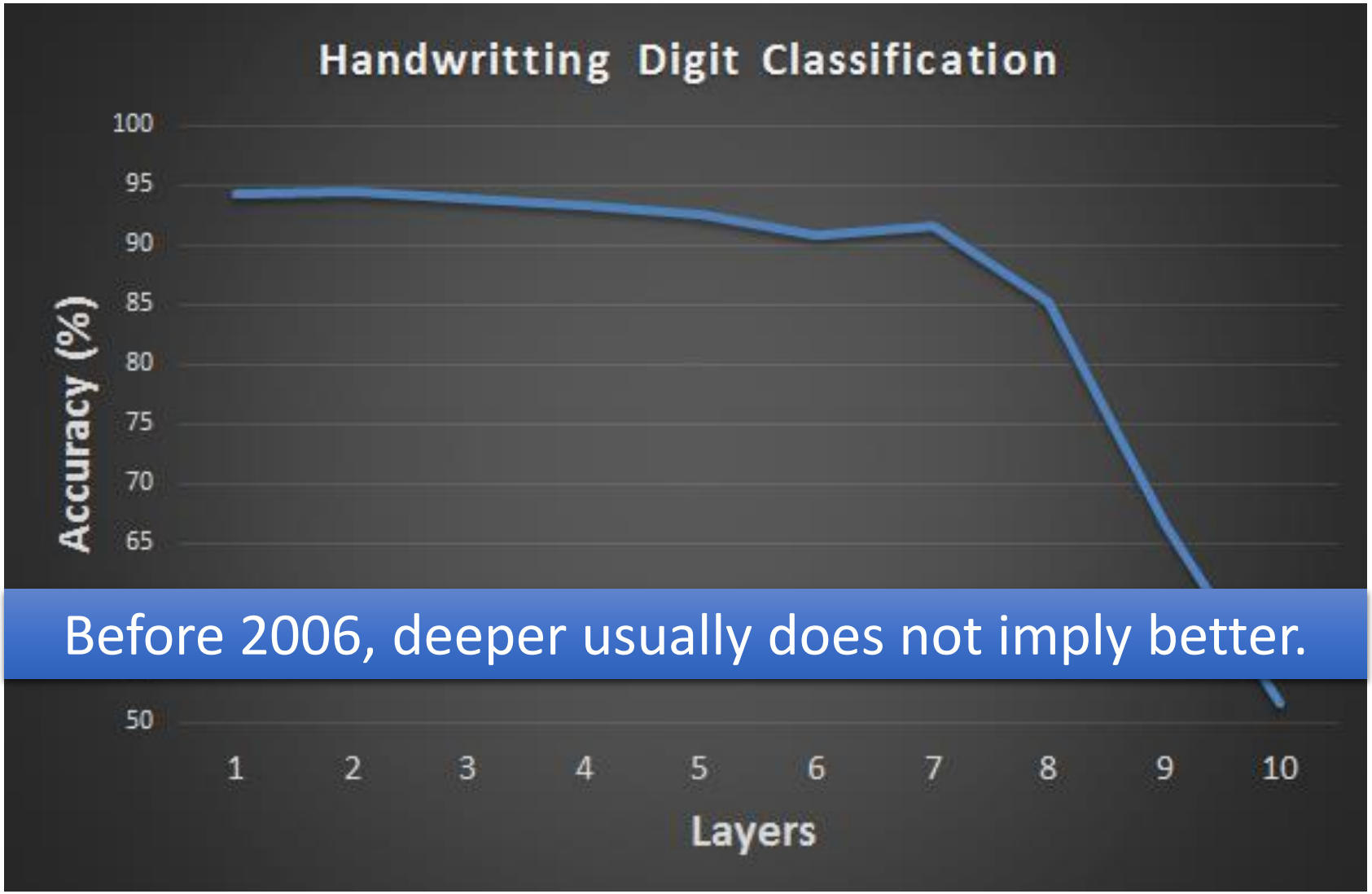


Source of image: http://www.gipsa-lab.grenoble-inp.fr/transfert/seminaire/455_Kadri2013Gipsa-lab.pdf

Deep Learning

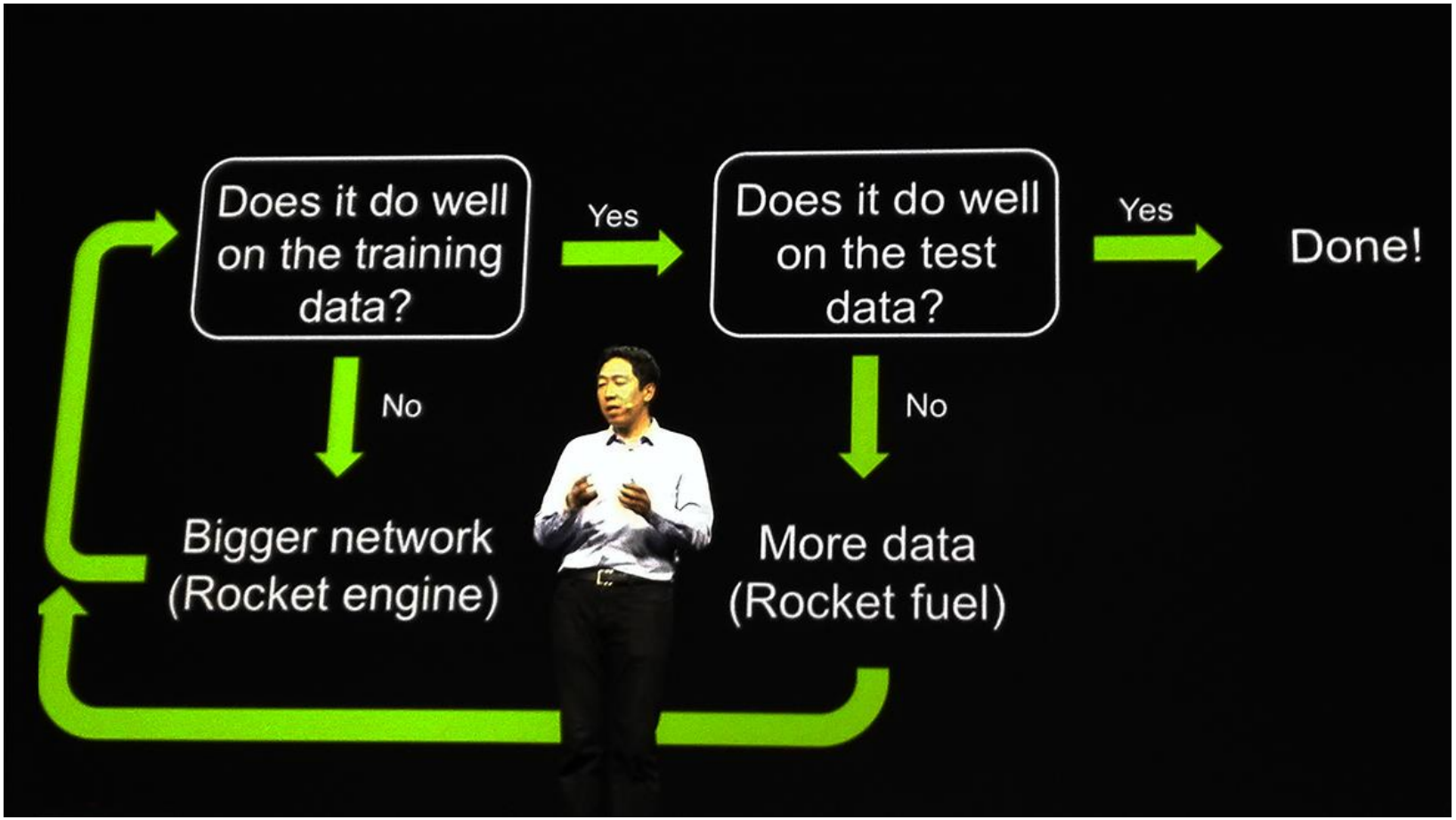


Hard to get the power of Deep …



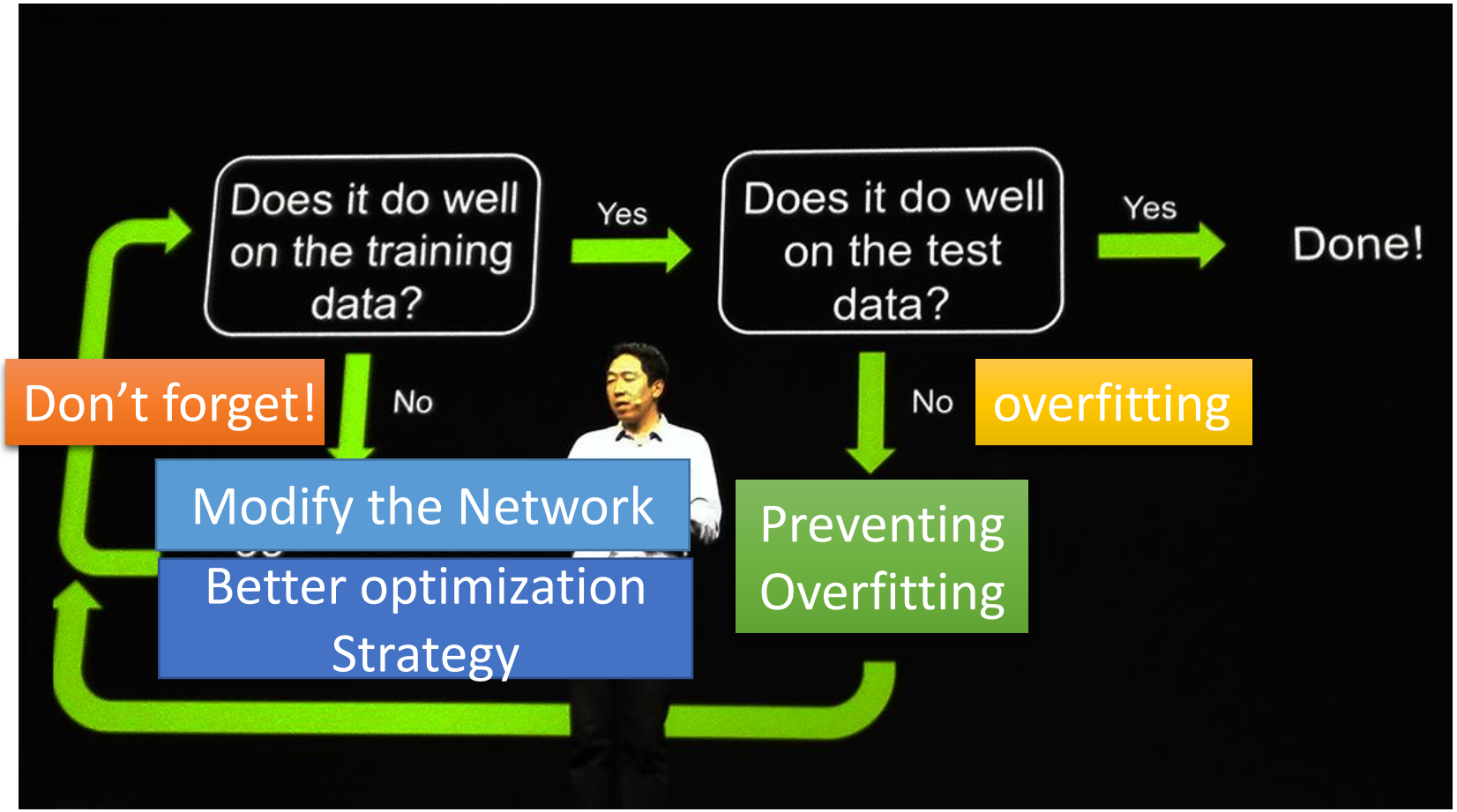
Part III:
Tips for Training DNN

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

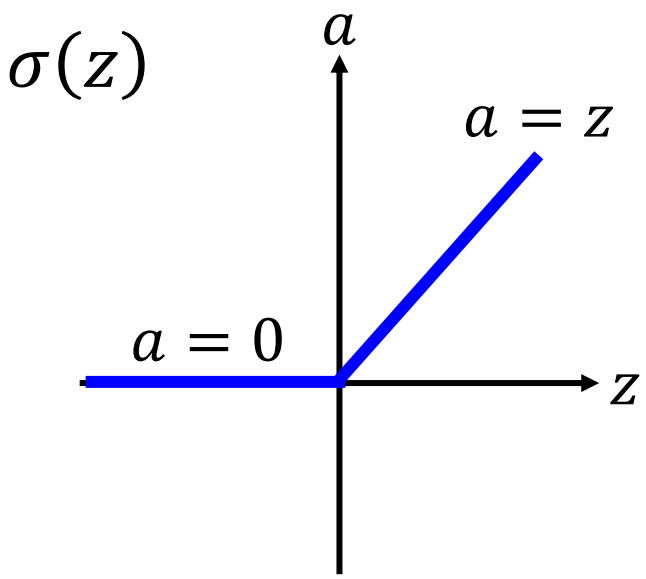
- Dropout

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

Part III:
Tips for Training DNN
New Activation Function

ReLU

- Rectified Linear Unit (ReLU)

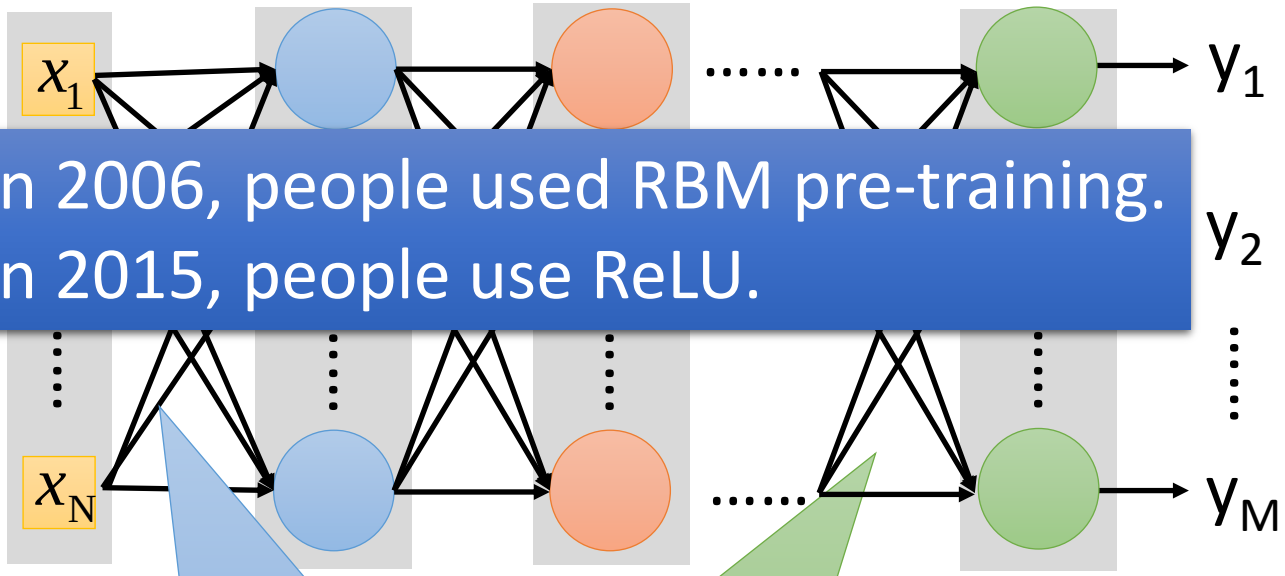


[Xavier Glorot, AISTATS'11]
[Andrew L. Maas, ICML'13]
[Kaiming He, arXiv'15]

Reason:

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

Vanishing Gradient Problem



In 2006, people used RBM pre-training.
In 2015, people use ReLU.

Smaller gradients

Learn very slow

Almost random

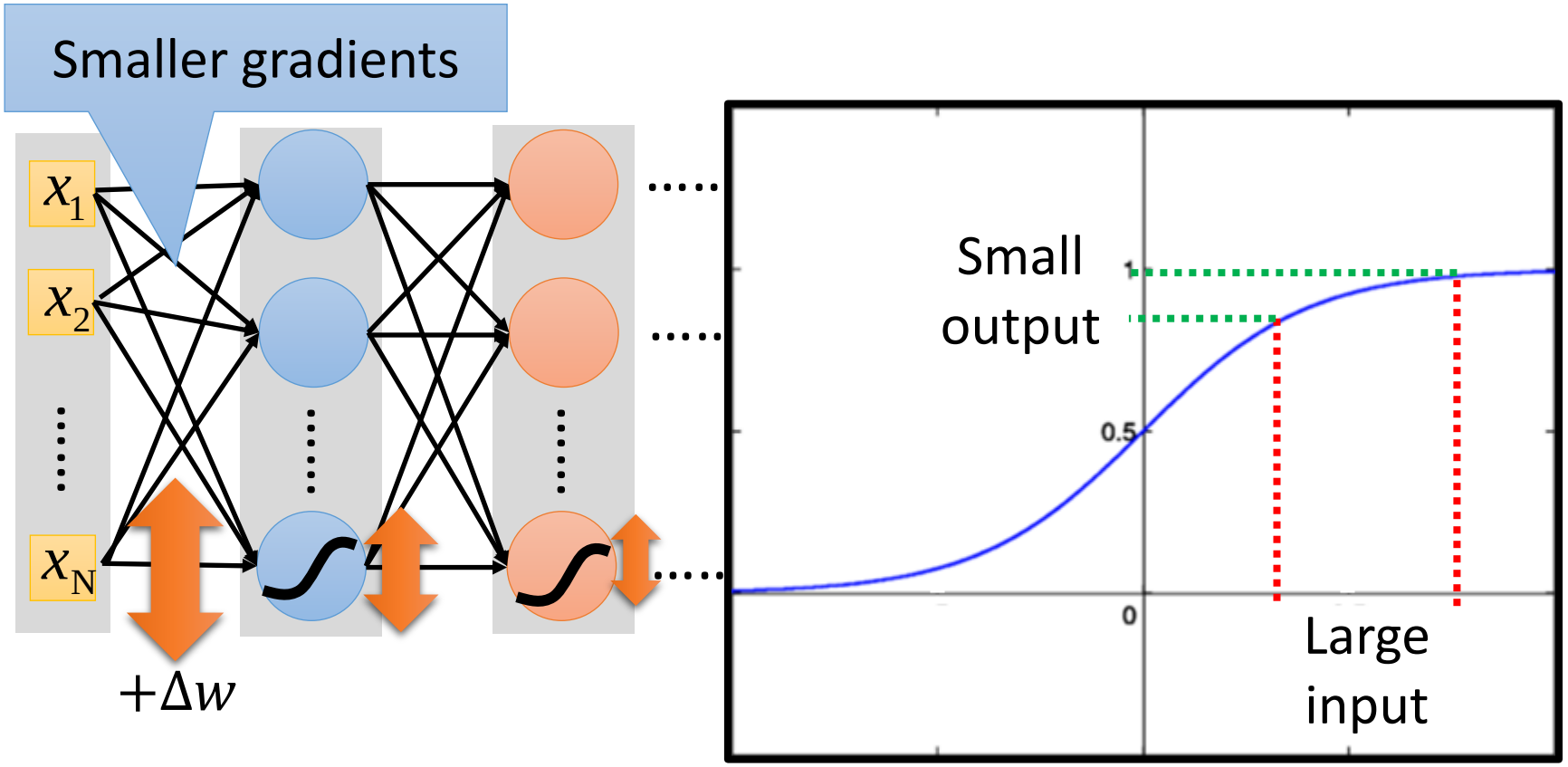
Larger gradients

Learn very fast

Already converge

based on random!?

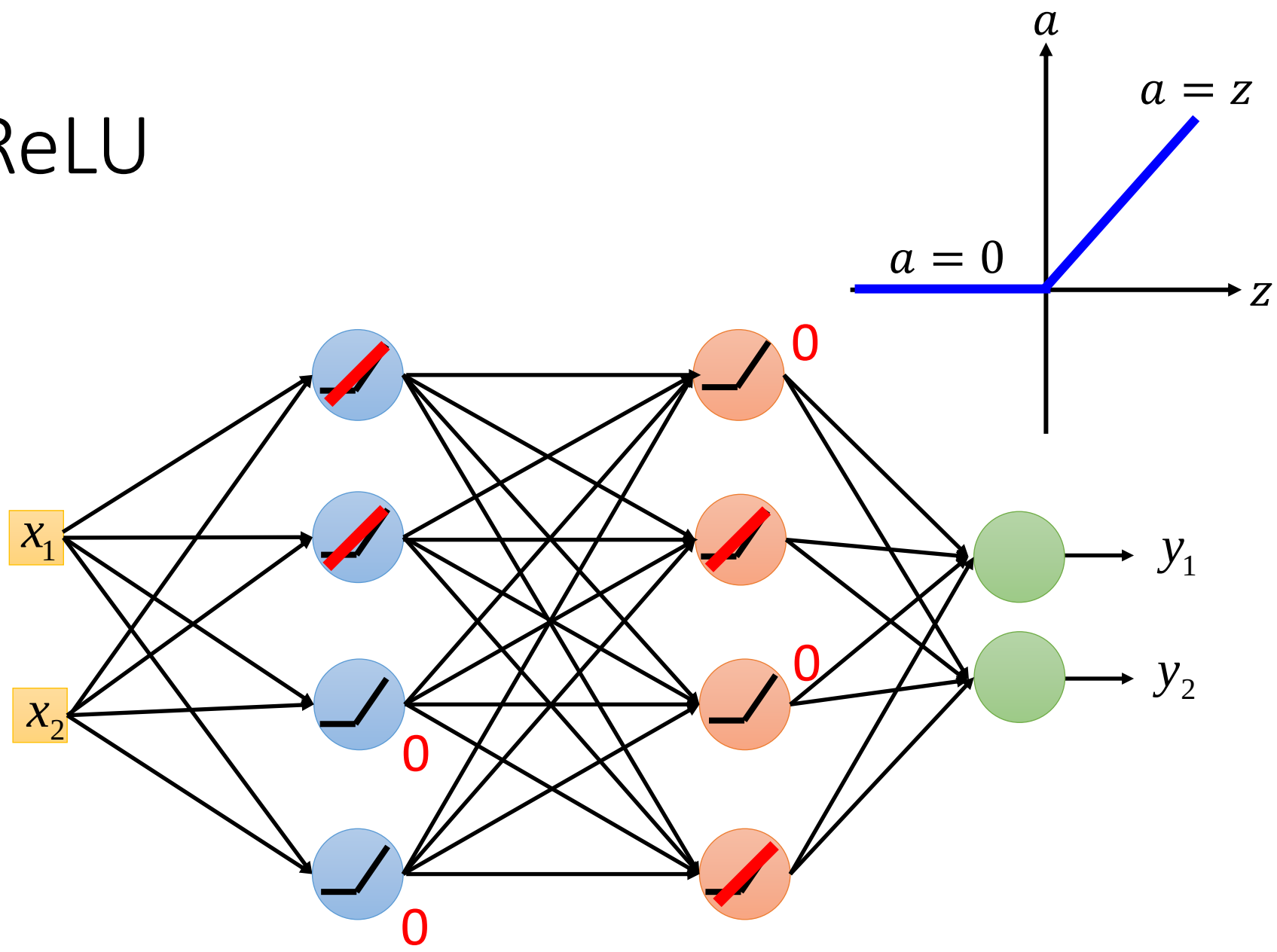
Vanishing Gradient Problem



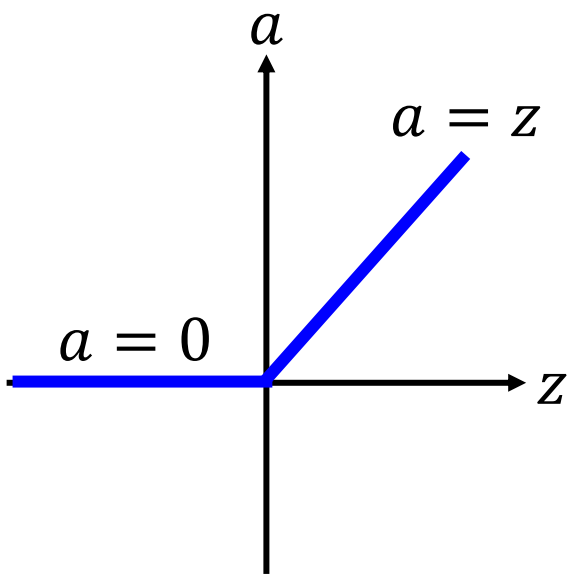
Intuitive way to compute the gradient …

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

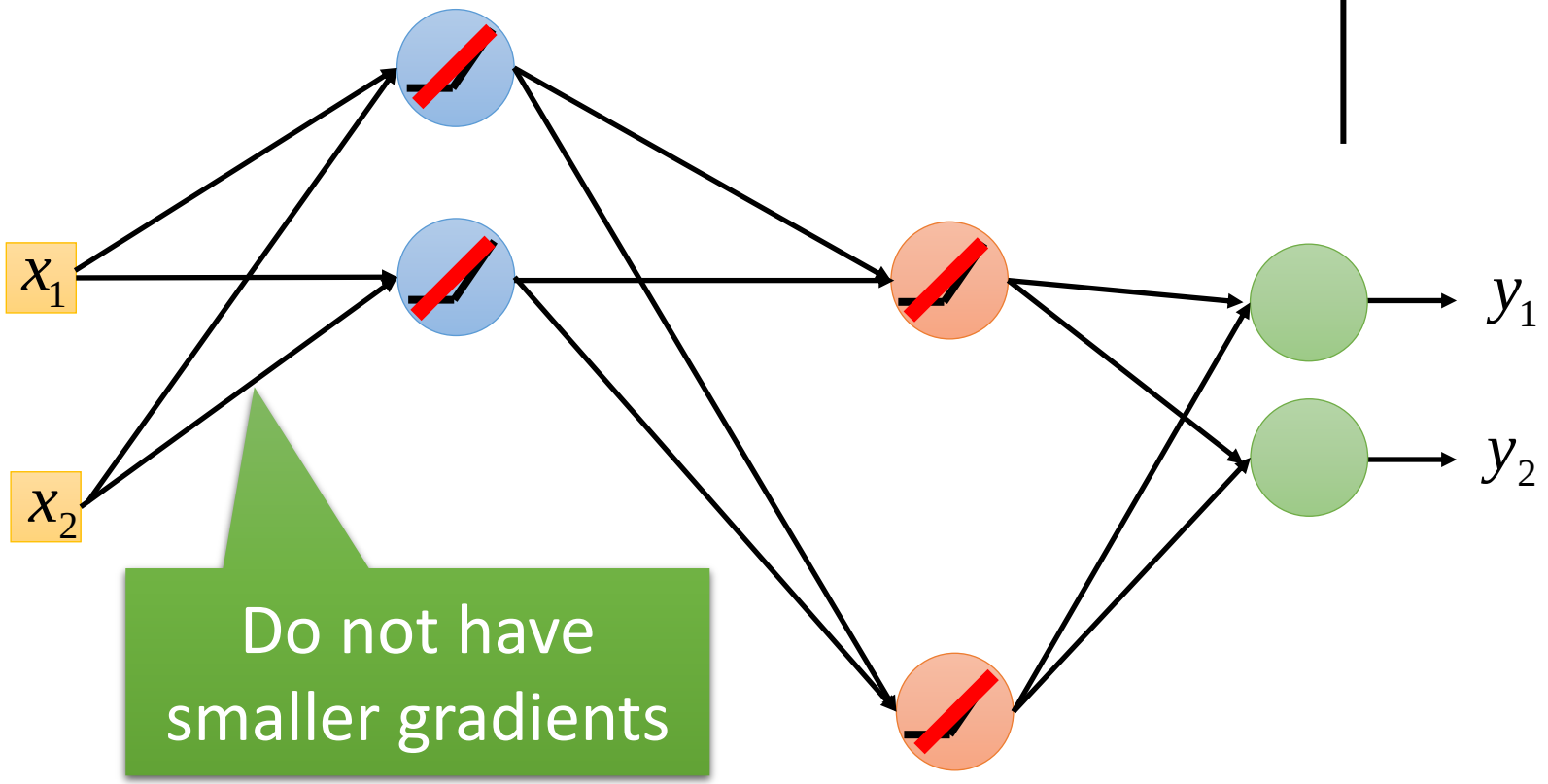
ReLU



ReLU



A Thinner linear network

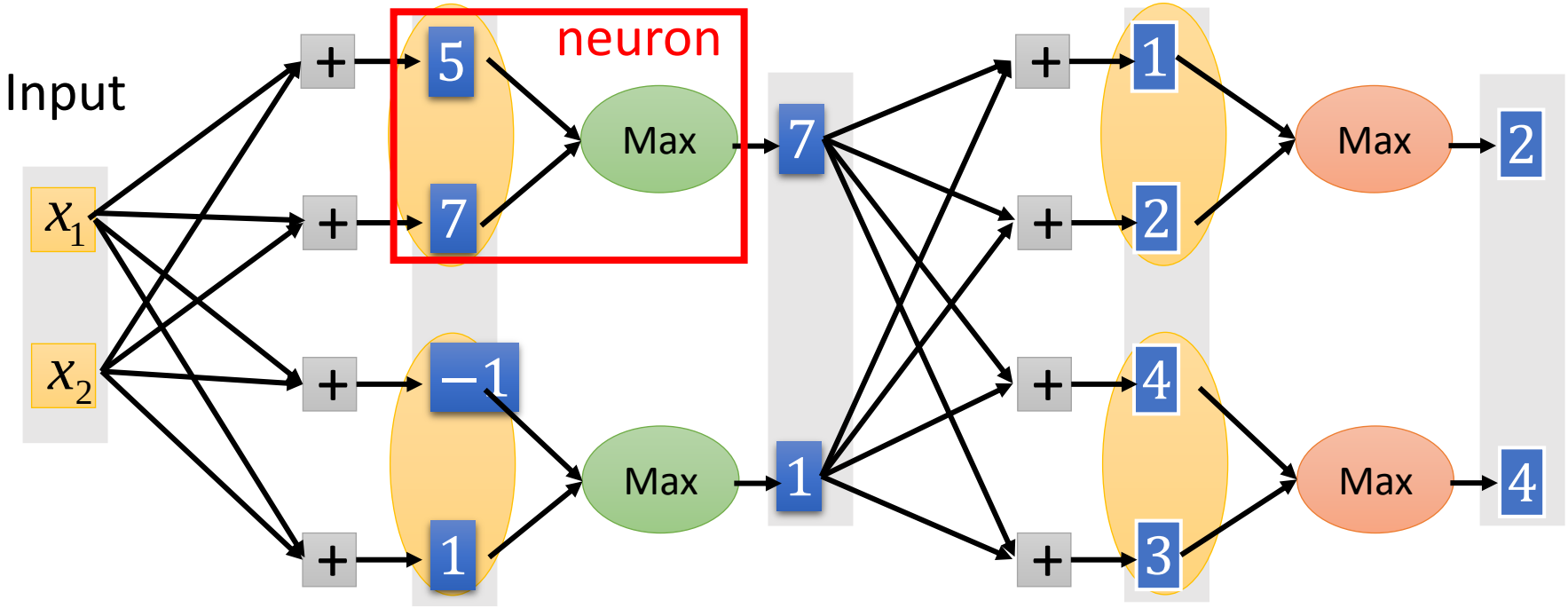


Do not have smaller gradients

Maxout

ReLU is a special cases of Maxout

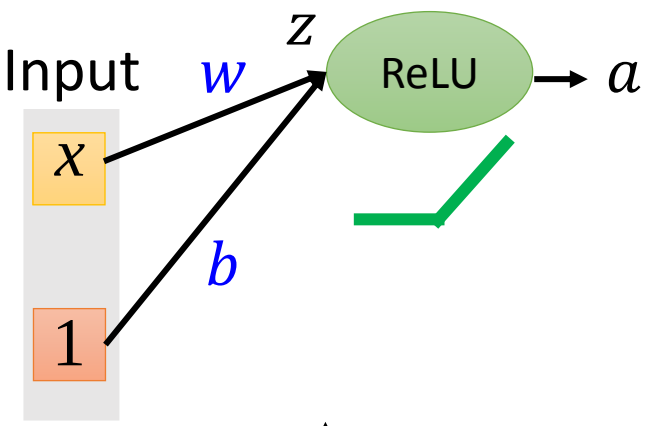
- Learnable activation function [Ian J. Goodfellow, ICML'13]

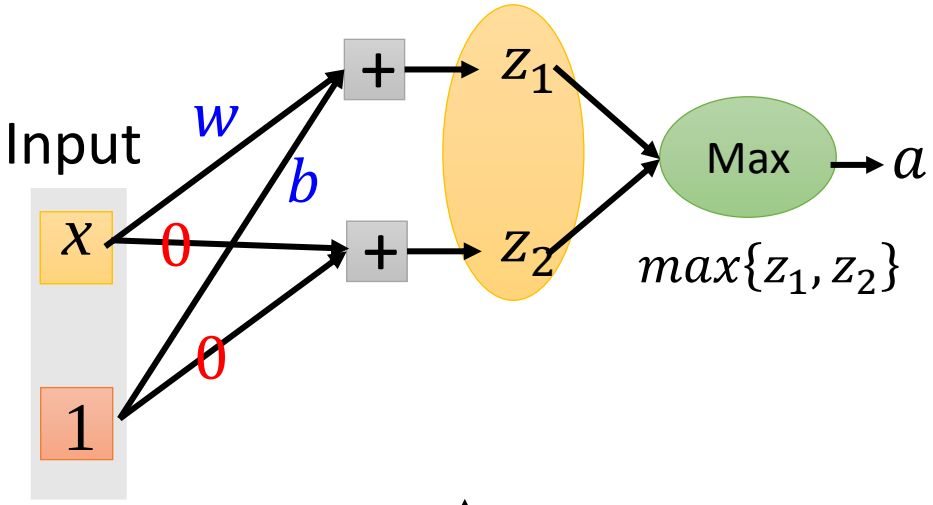


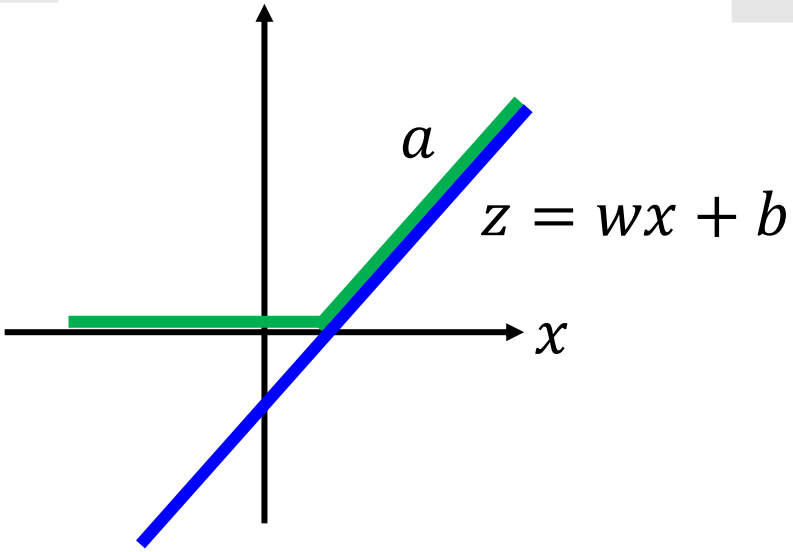
You can have more than 2 elements in a group.

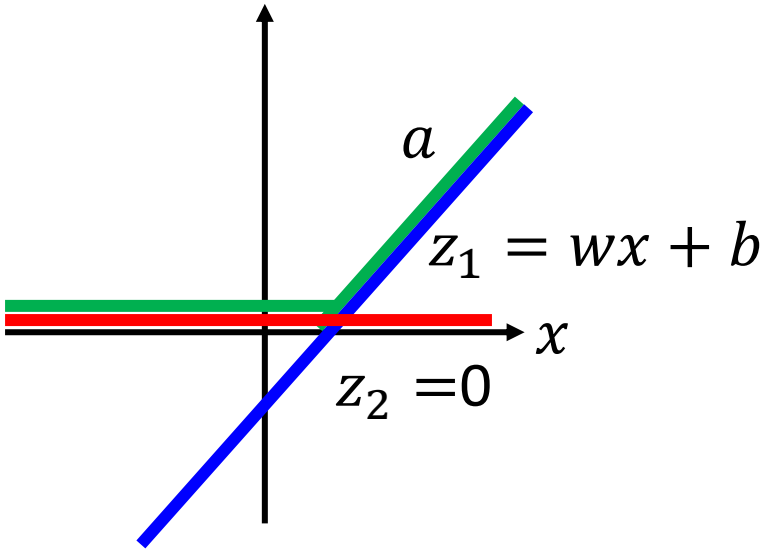
Maxout

ReLU is a special cases of Maxout



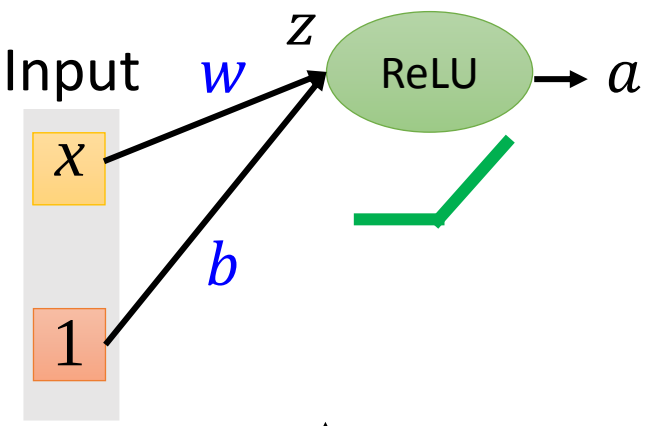


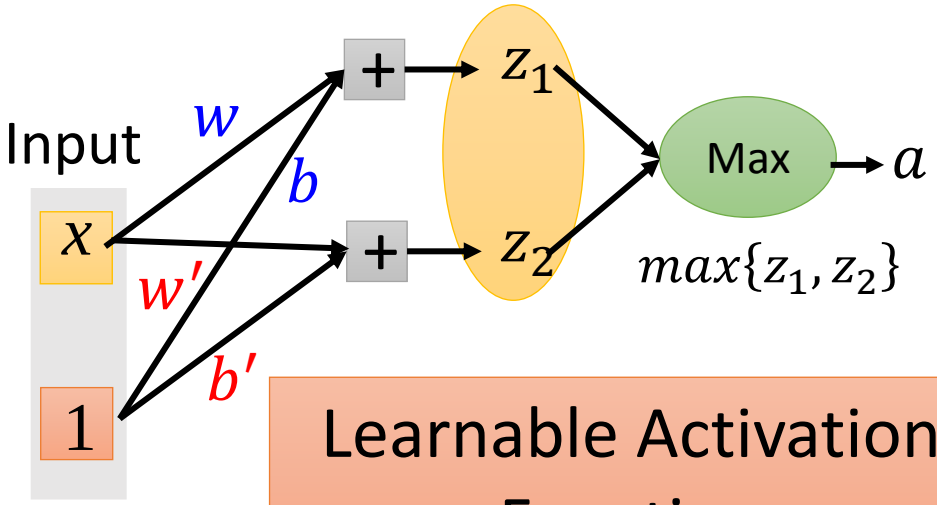




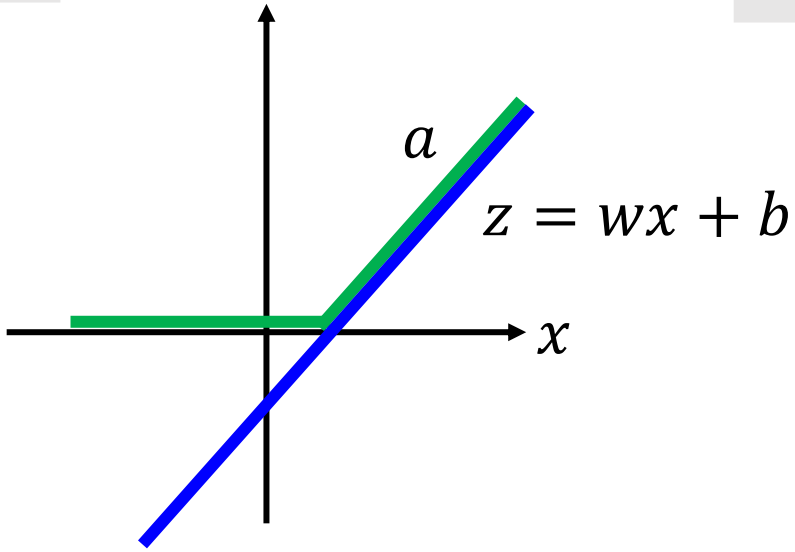
Maxout

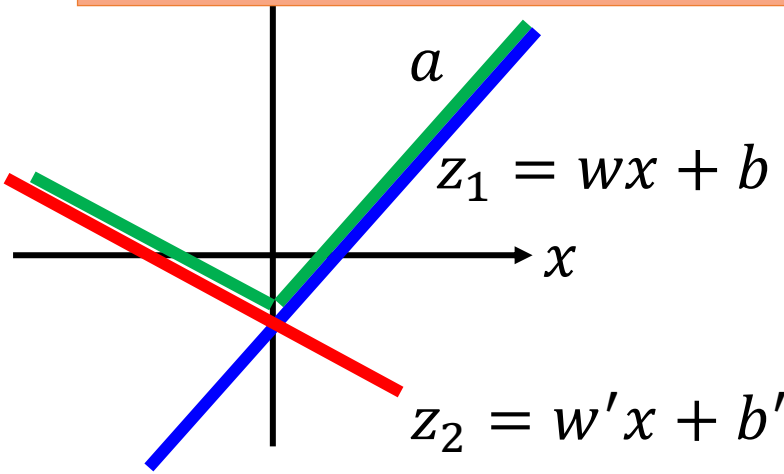
ReLU is a special cases of Maxout





Learnable Activation Function



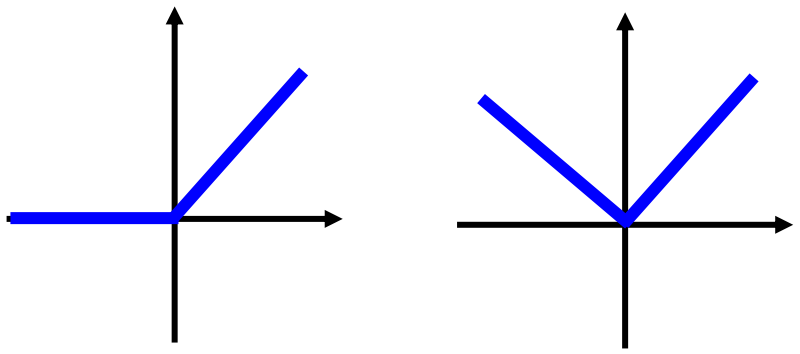


Maxout

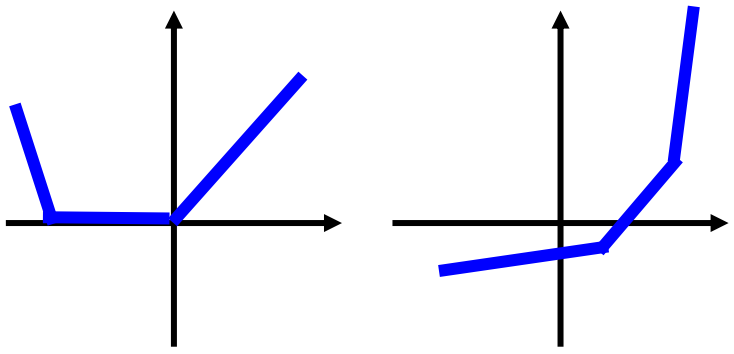
ReLU is a special cases of Maxout

- 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

2 elements in a group



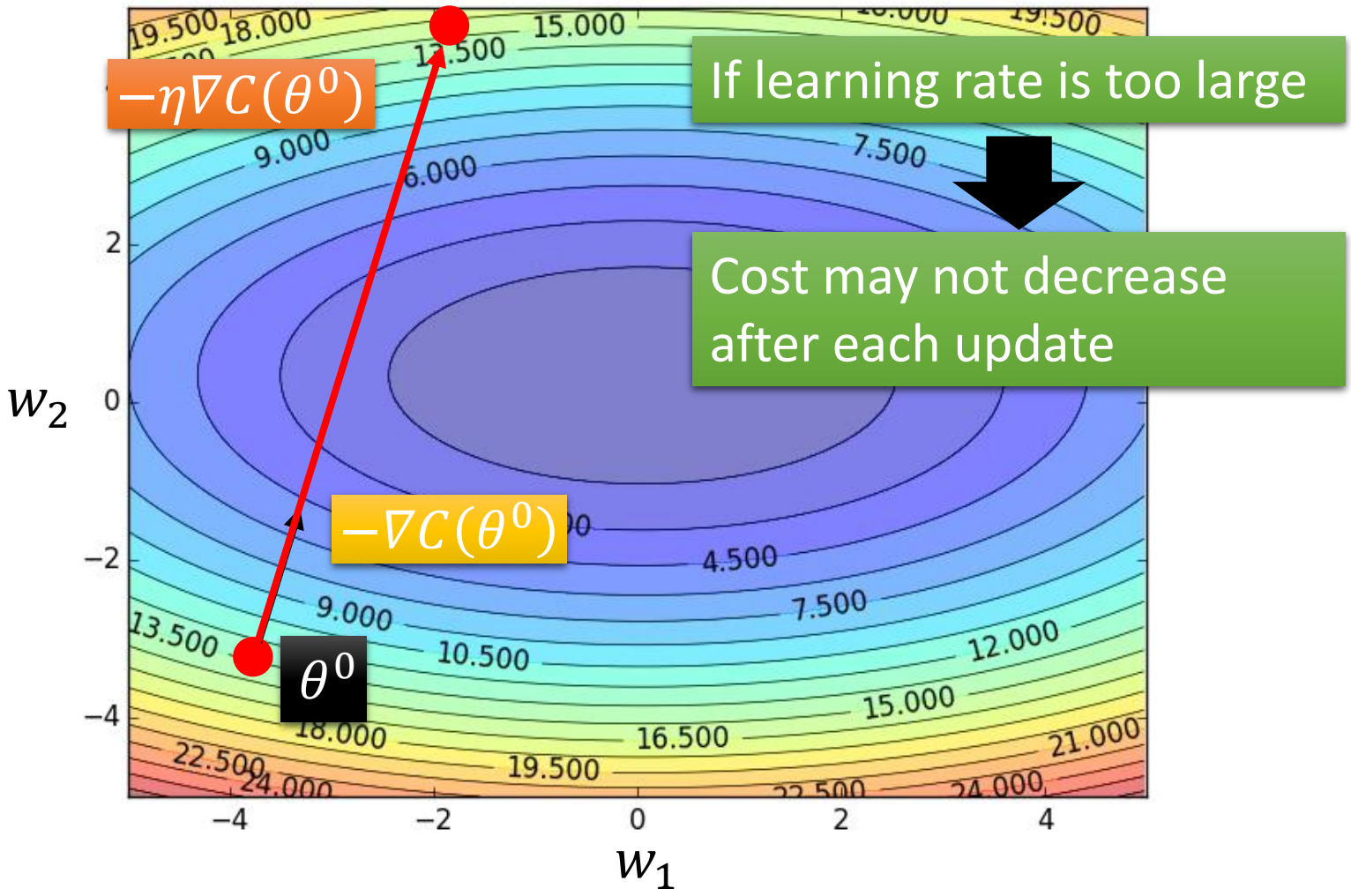
3 elements in a group



Part III:
Tips for Training DNN
Adaptive Learning Rate

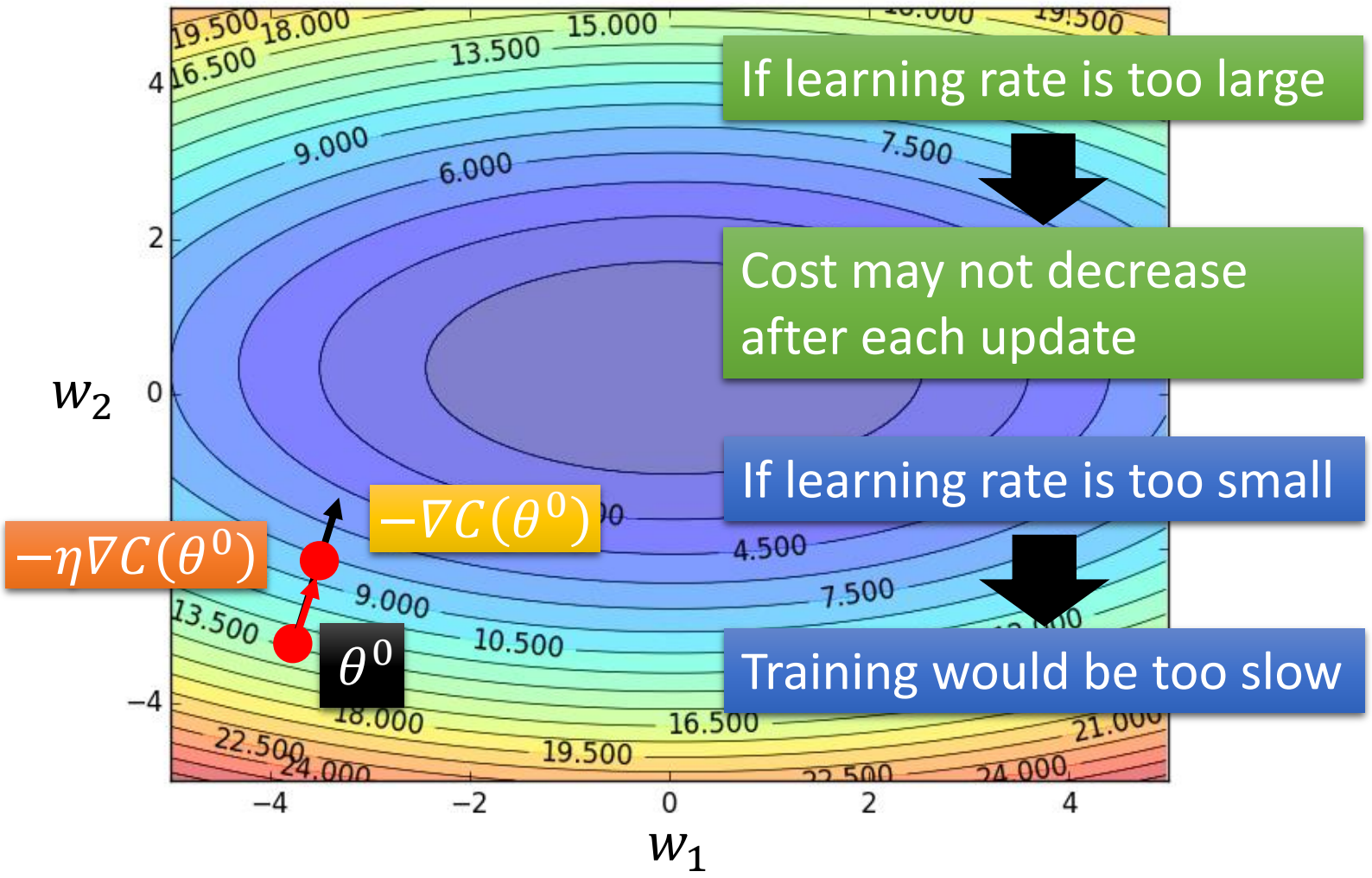
Learning Rate

Set the learning rate η carefully



Learning Rate

Can we give different parameters different learning rates?



Adagrad

Original Gradient Descent

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

Each parameter w are considered separately

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

Parameter dependent learning rate

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

constant

Summation of the square of the previous derivatives

Adagrad

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

w_1	g^0
	0.1

w_2	g^0
	20.0

Learning rate:

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

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

Learning rate:

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

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

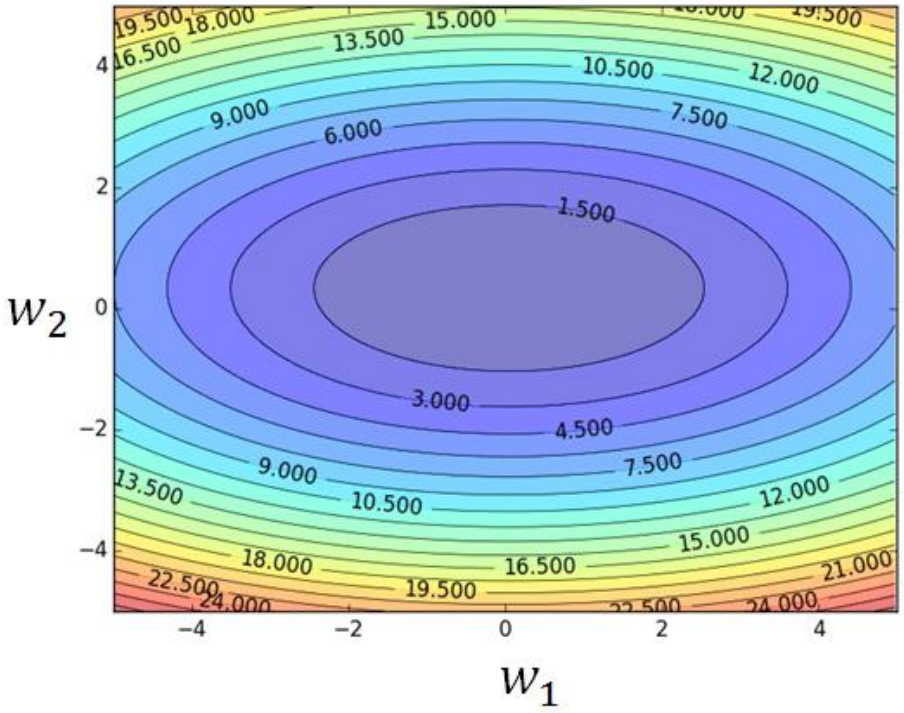
Observation:

1. Learning rate is smaller and smaller for all parameters
2. Smaller derivatives, larger learning rate, and vice versa

Why?

Larger derivatives

Smaller Learning Rate



Smaller Derivatives

Larger Learning Rate

2. Smaller derivatives, larger learning 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

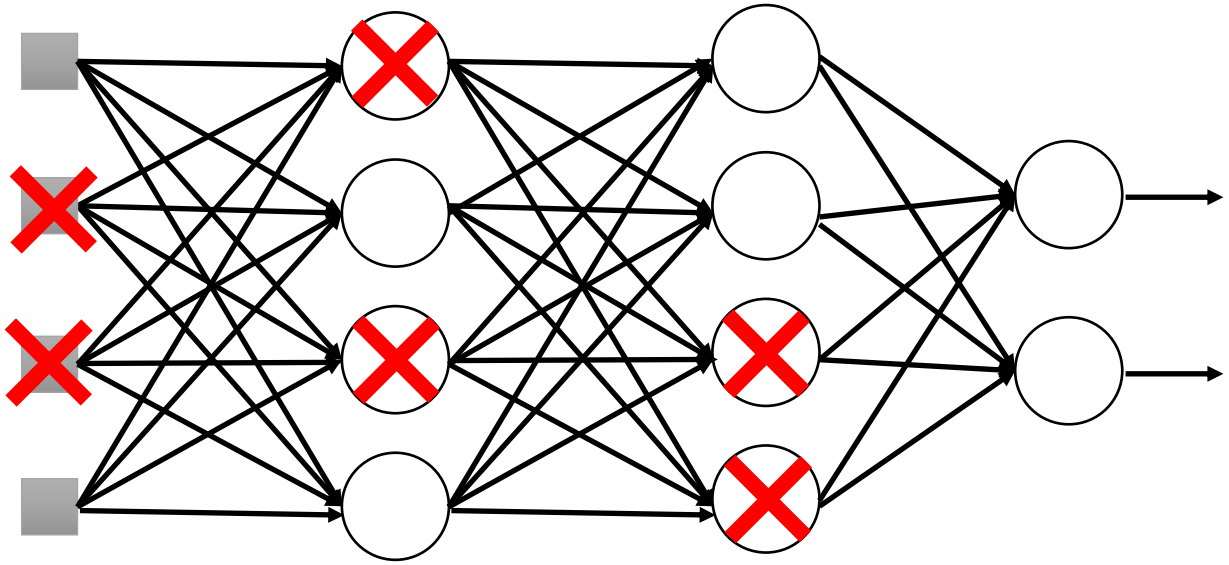
Dropout

Dropout

Pick a mini-batch

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

Training:



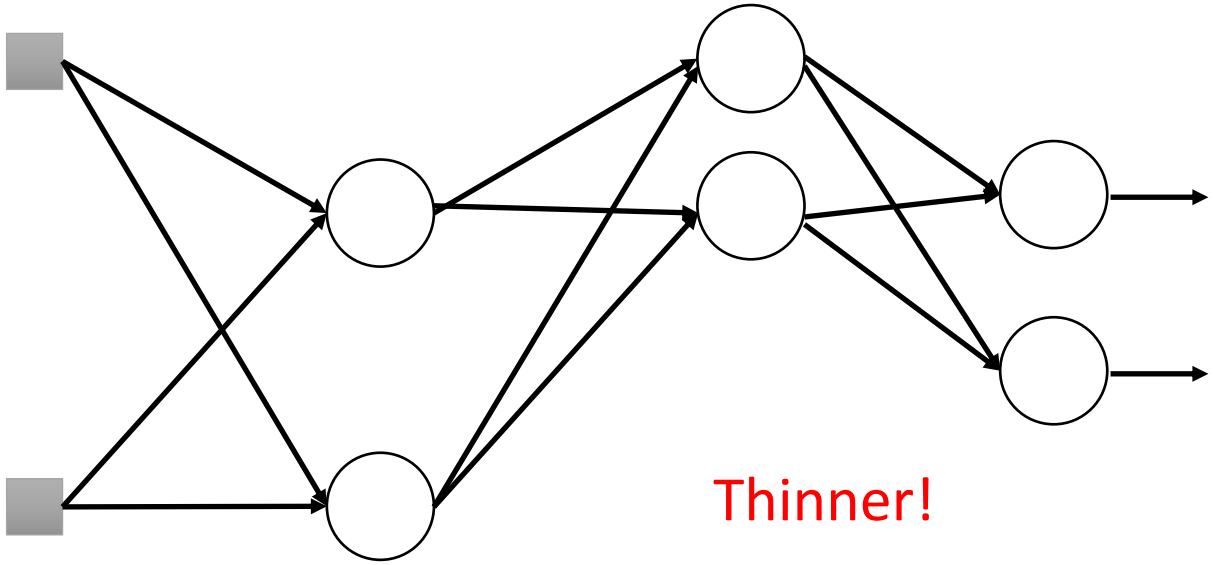
- **Each time before computing the gradients**
  - Each neuron has p% to dropout

Dropout

Pick a mini-batch

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

Training:



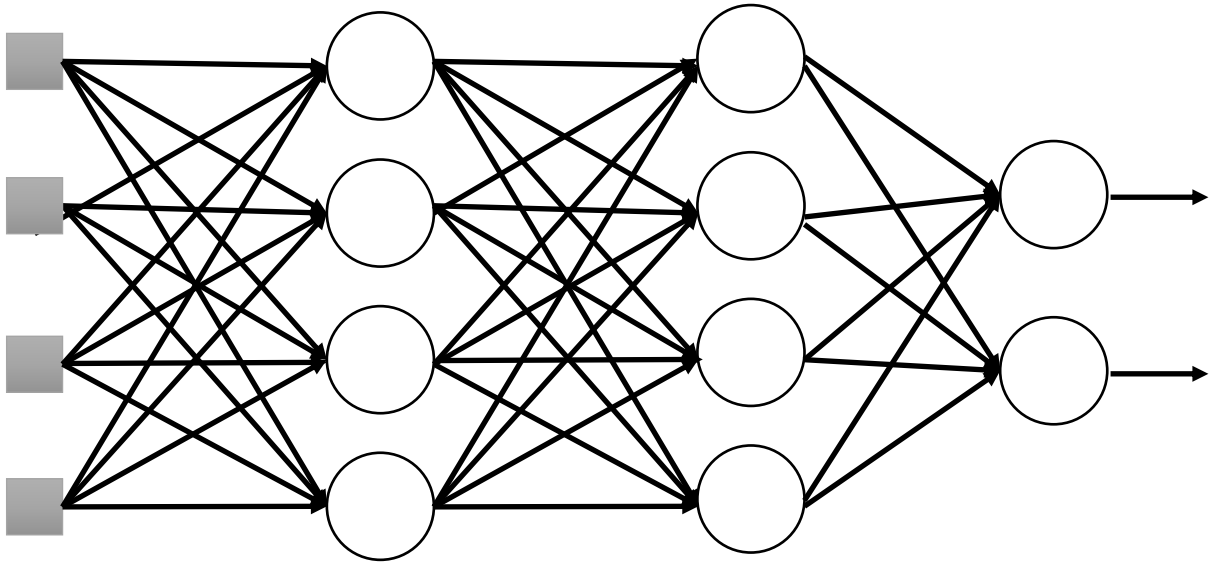
- **Each time before computing the gradients**
  - Each neuron has p% to dropout

    The structure of the network is changed.
  - Using the new network for training

For each mini-batch, we resample the dropout neurons

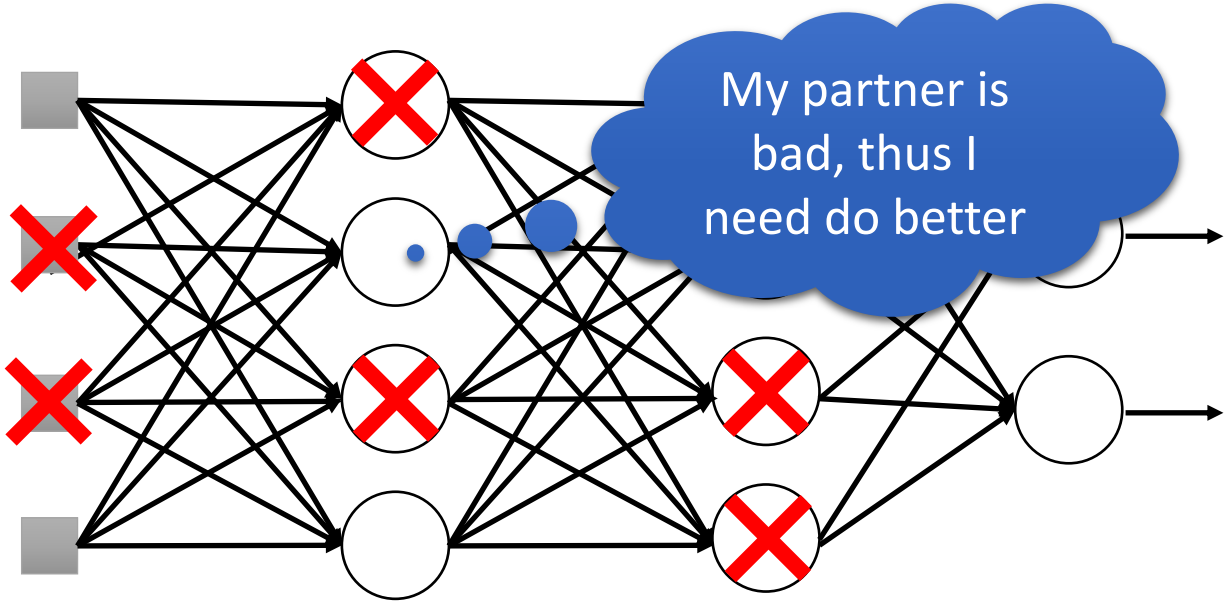
Dropout

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



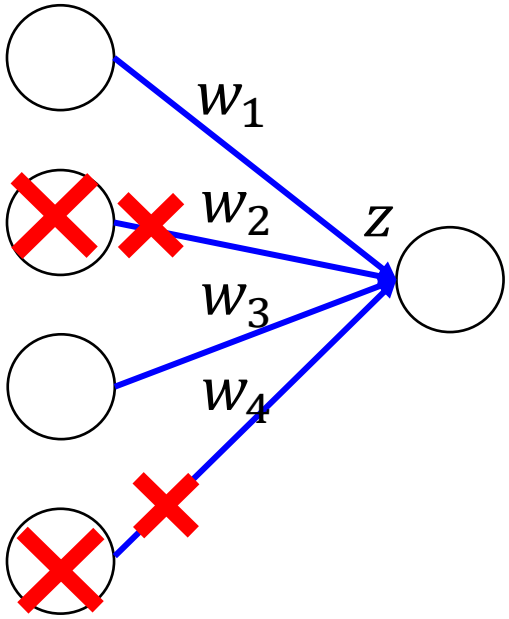
- When teams up, if everyone expect the partner will do the work, nothing will be done finally.
- However, if you know your partner will dropout, you will do better.
- When testing, no one dropout actually, so obtaining good results eventually.

Dropout - Intuitive Reason

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

Training of Dropout

Assume dropout rate is 50%

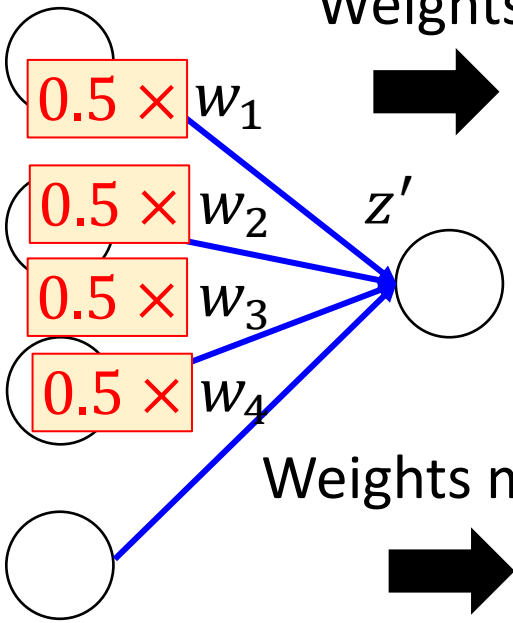


Testing of Dropout

No dropout

Weights from training

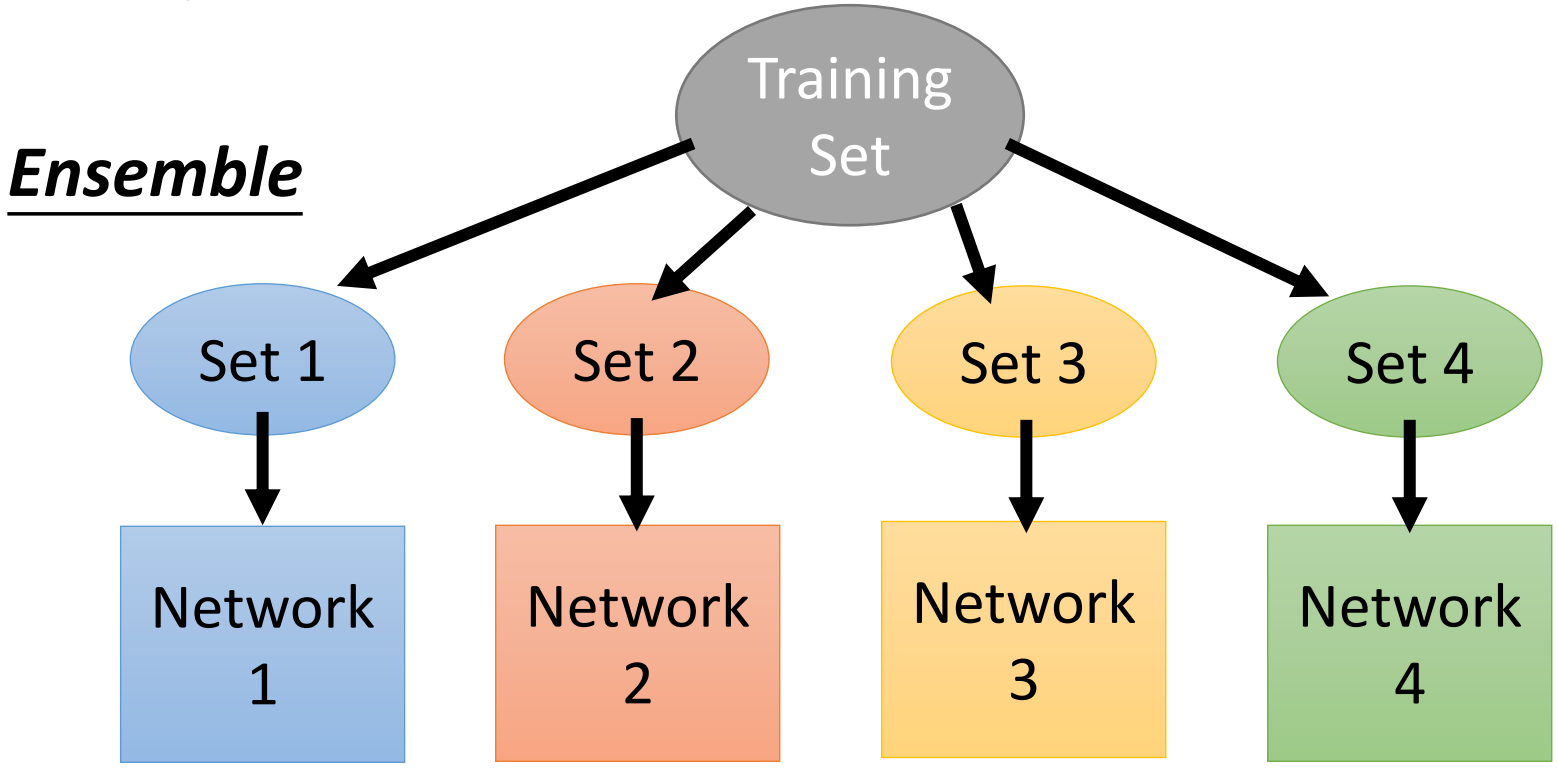
$z' \approx 2z$



Weights multiply (1-p)%

$z' \approx z$

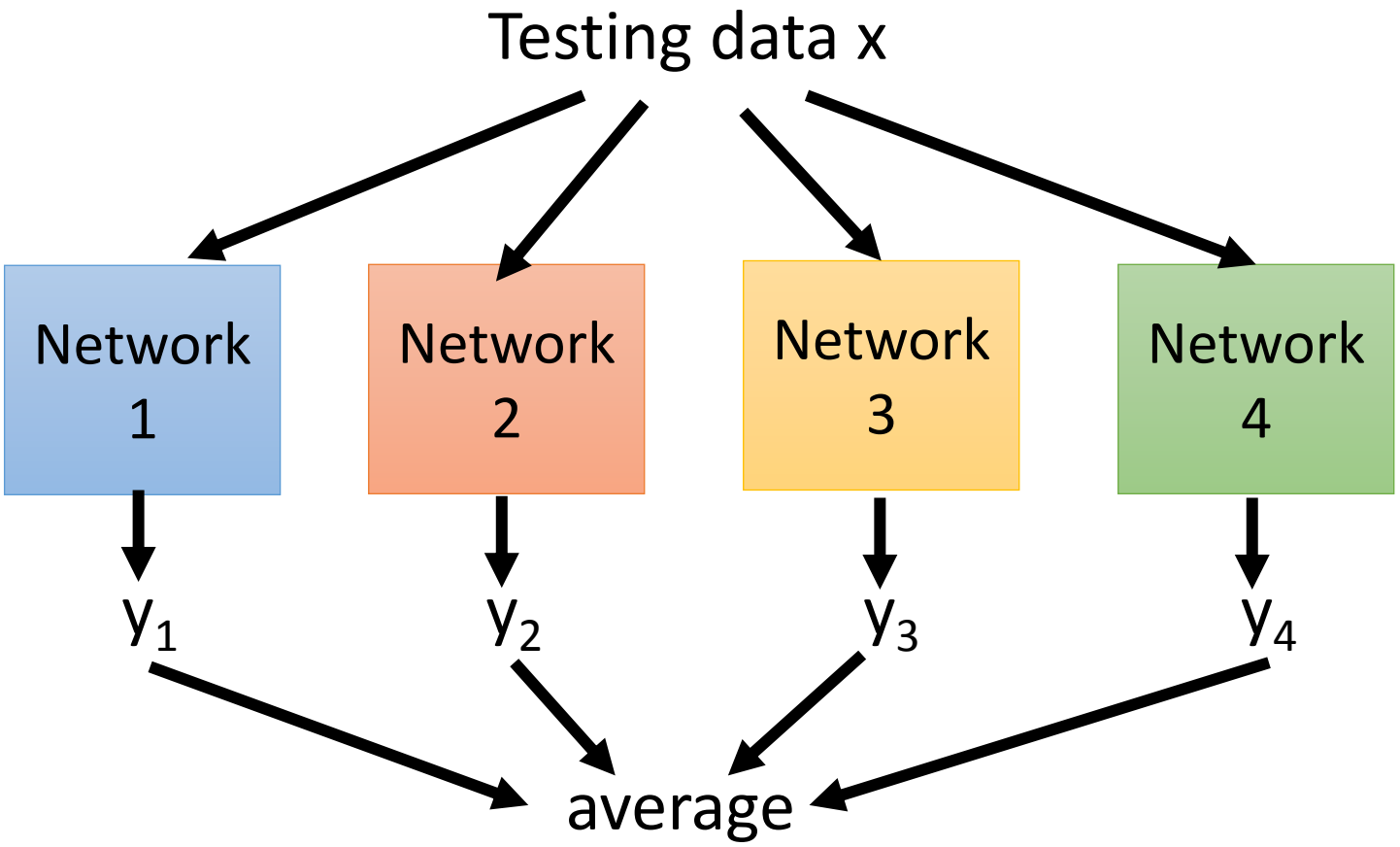
Dropout is a kind of ensemble.



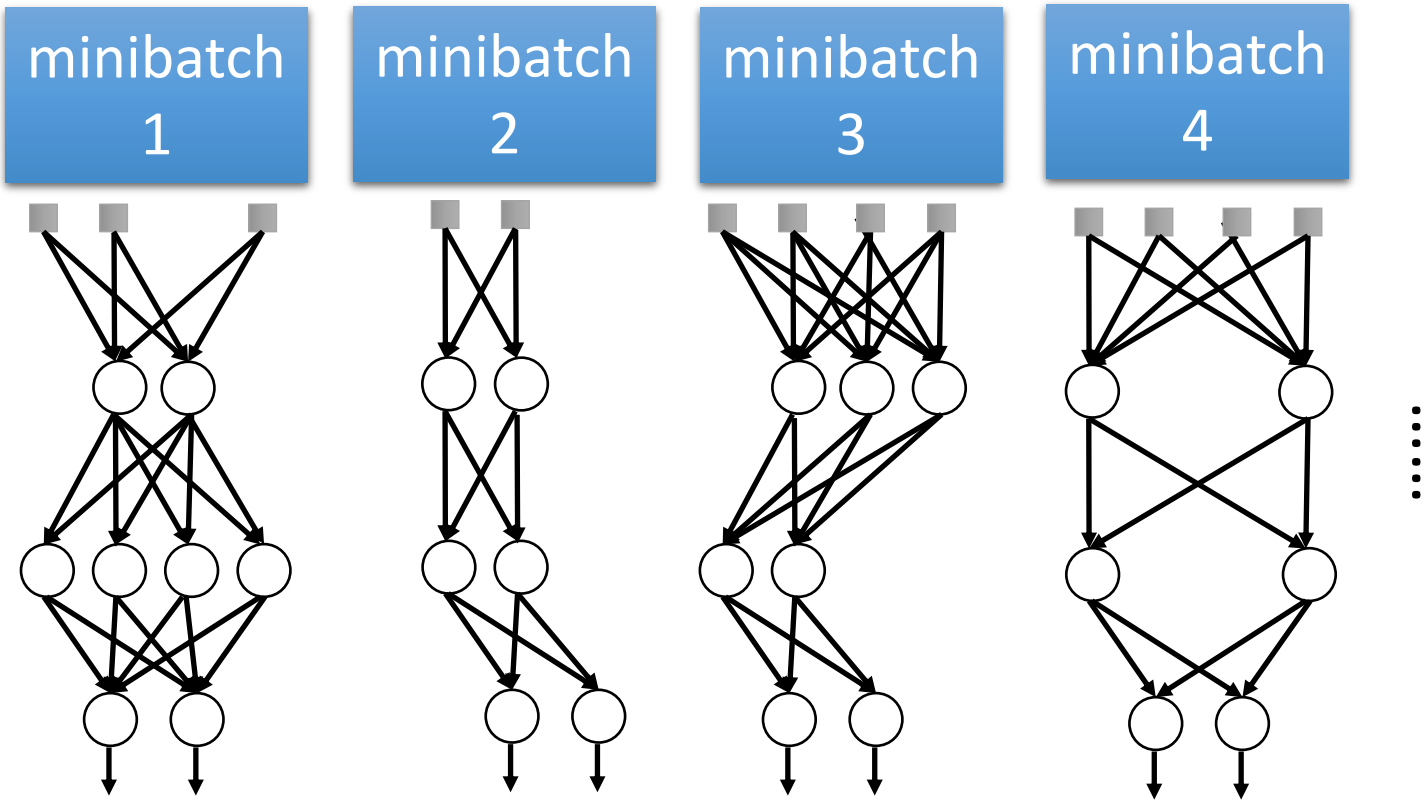
Train a bunch of networks with different structures

Dropout is a kind of ensemble.

Ensemble



Dropout is a kind of ensemble.



Training of Dropout

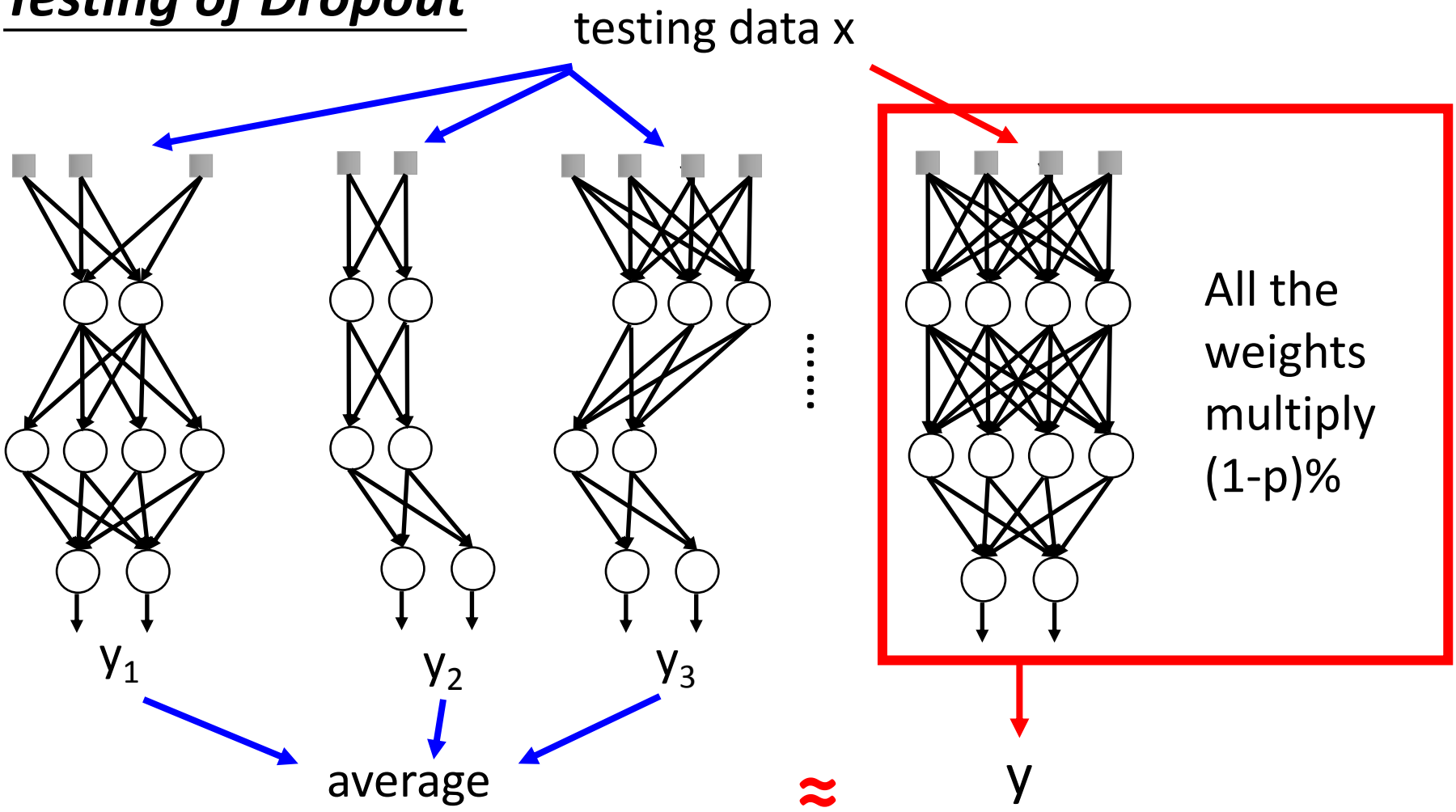
M neurons

2^M possible networks

- Using one mini-batch to train one network
- Some parameters in the network are shared

Dropout is a kind of ensemble.

Testing of Dropout



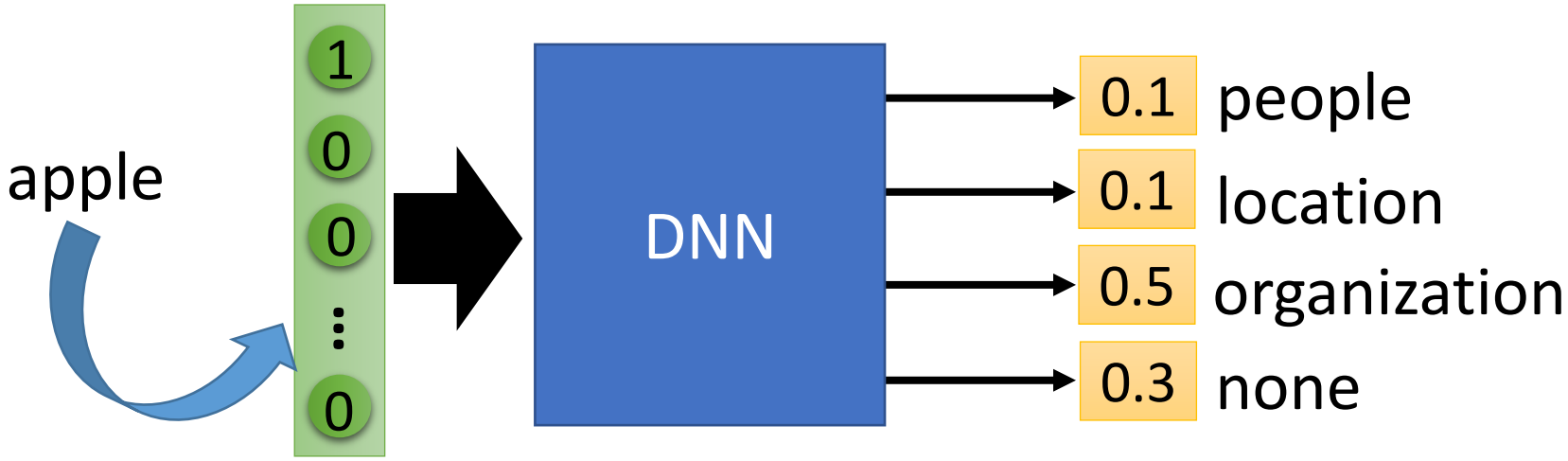
More about dropout

- More reference for dropout [Nitish Srivastava, JMLR'14] [Pierre Baldi, NIPS'13][Geoffrey E. Hinton, arXiv'12]
- Dropout works better with Maxout [Ian J. Goodfellow, ICML'13]
- Dropconnect [Li Wan, *ICML'13*]
  - Dropout delete neurons
  - Dropconnect deletes the connection between neurons
- Annealed dropout [S.J. Rennie, SLT'14]
  - Dropout rate decreases by epochs
- Standout [J. Ba, NISP'13]
  - Each neural has different dropout rate

Part IV:
Neural Network with Memory

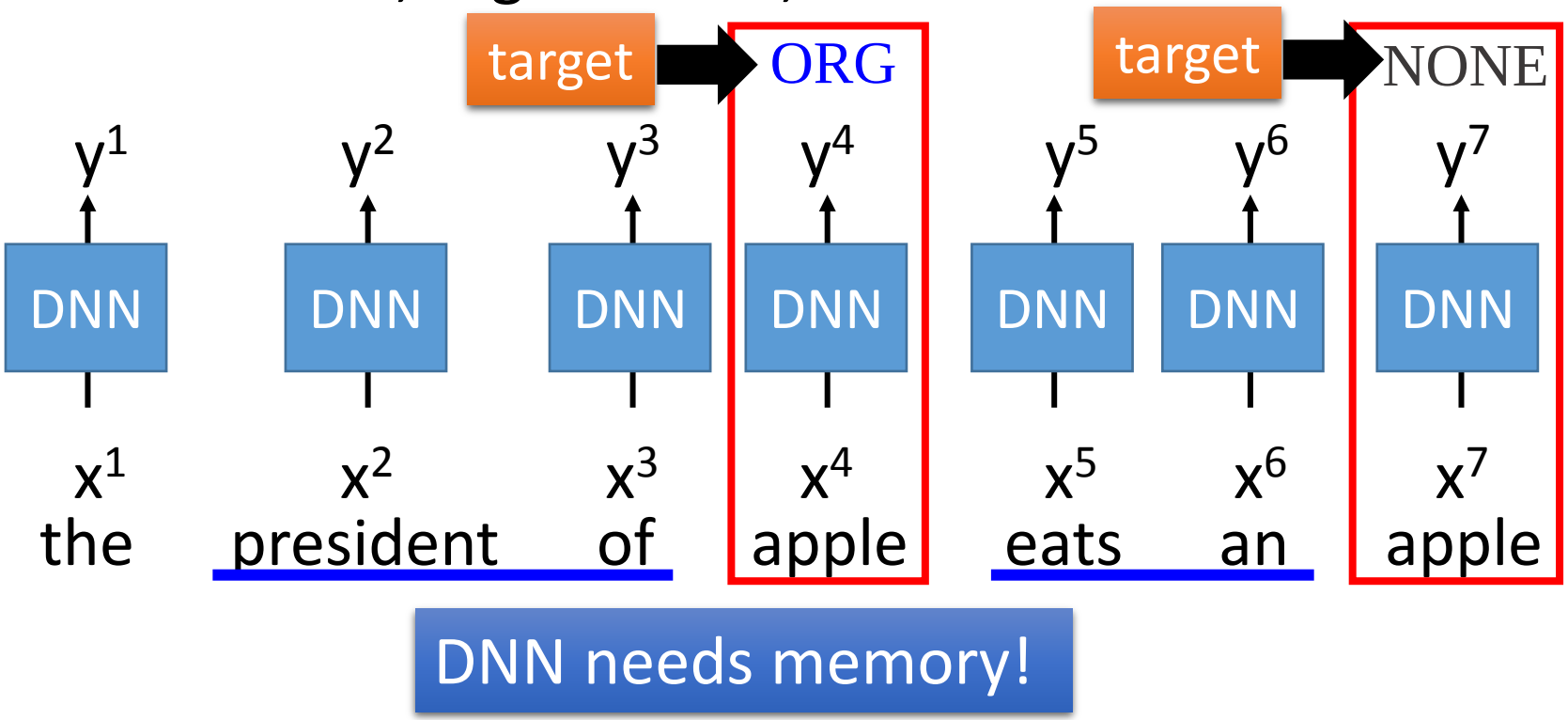
Neural Network needs Memory

- Name Entity Recognition
  - Detecting named entities like name of people, locations, organization, etc. in a sentence.



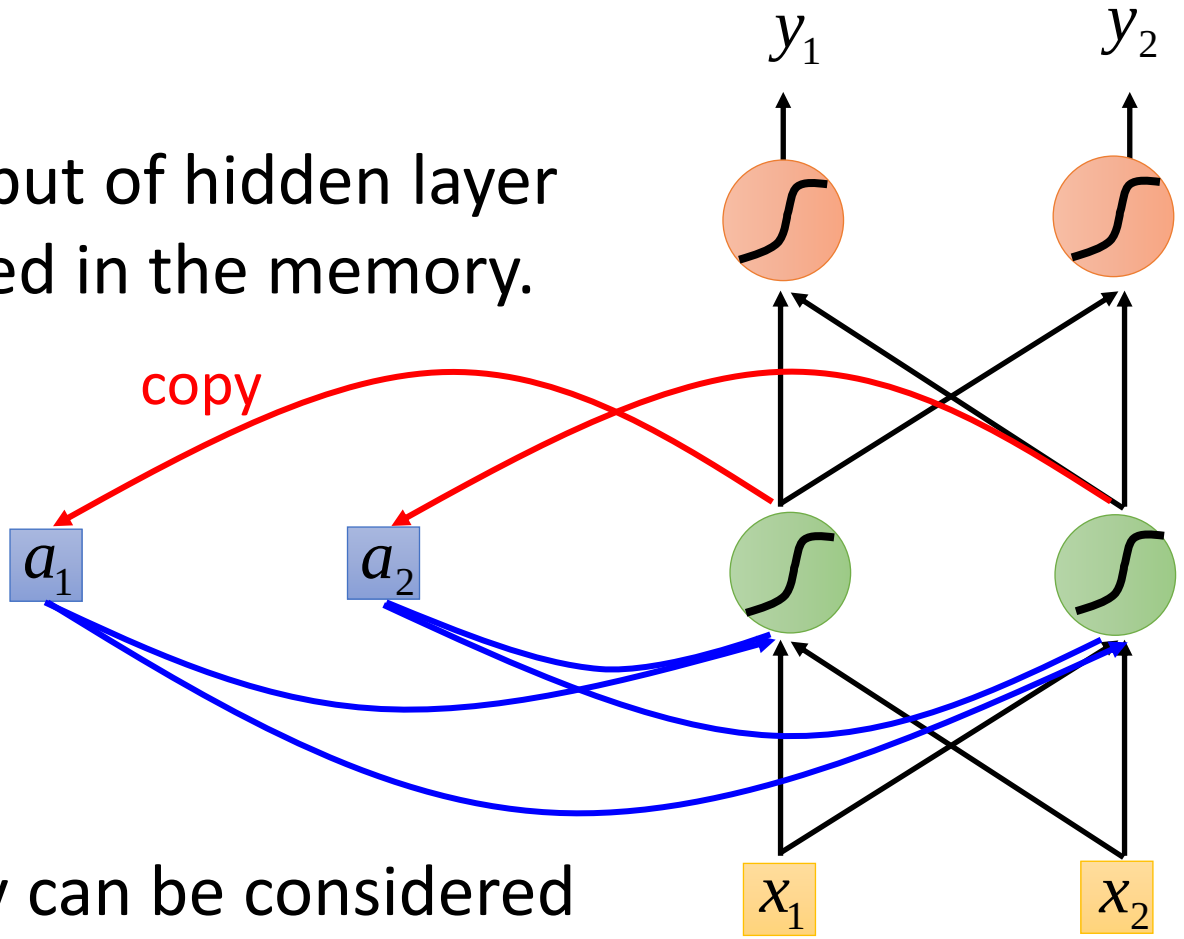
Neural Network needs Memory

- Name Entity Recognition
  - Detecting named entities like name of people, locations, organization, etc. in a sentence.



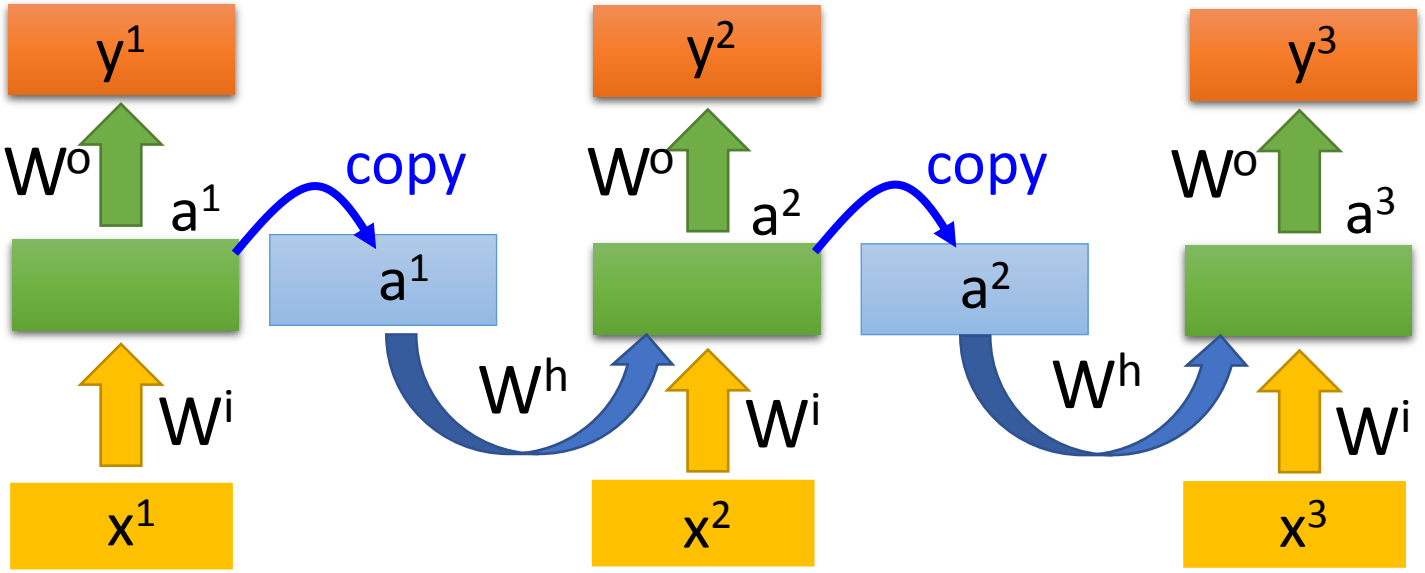
Recurrent Neural Network (RNN)

The output of hidden layer are stored in the memory.



Memory can be considered as another input.

RNN

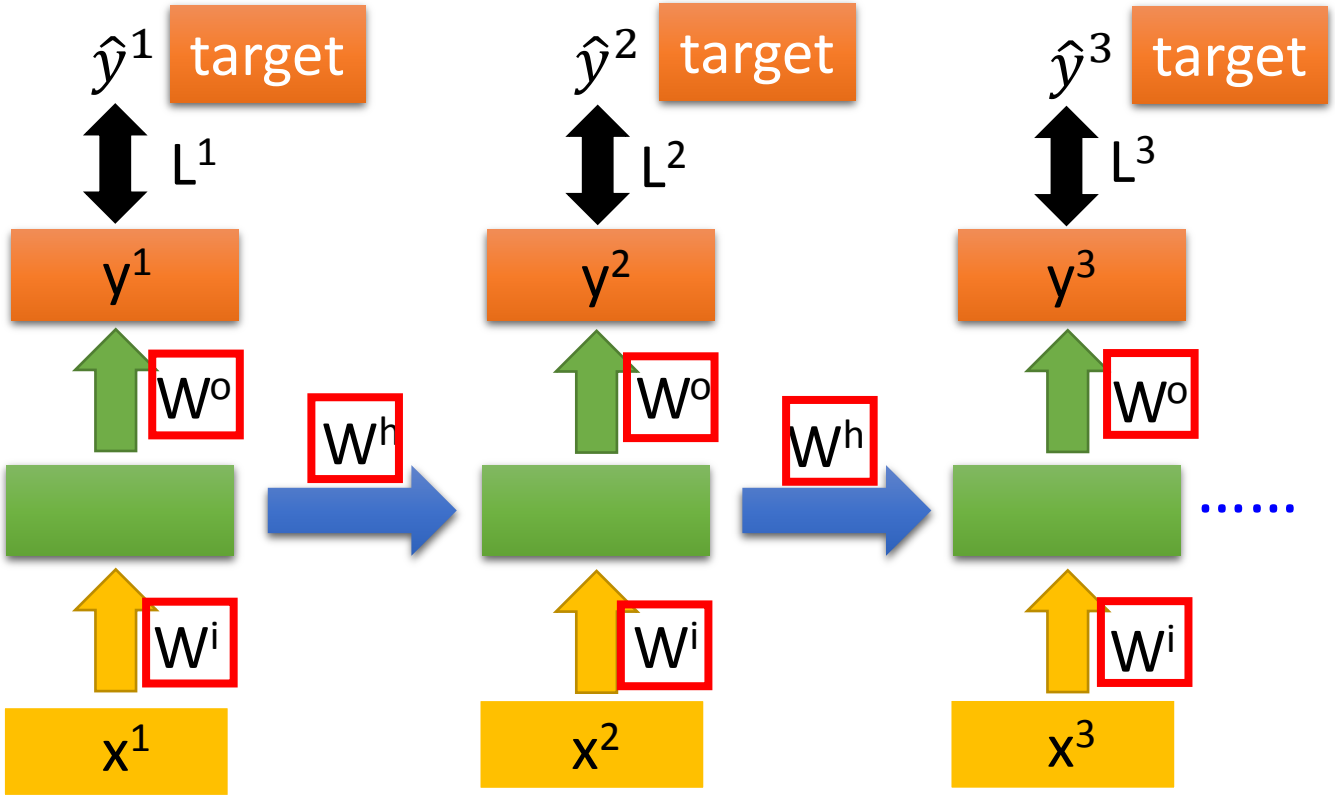


The same network is used again and again.

Output y^i depends on x^1, x^2, …… x^i

RNN

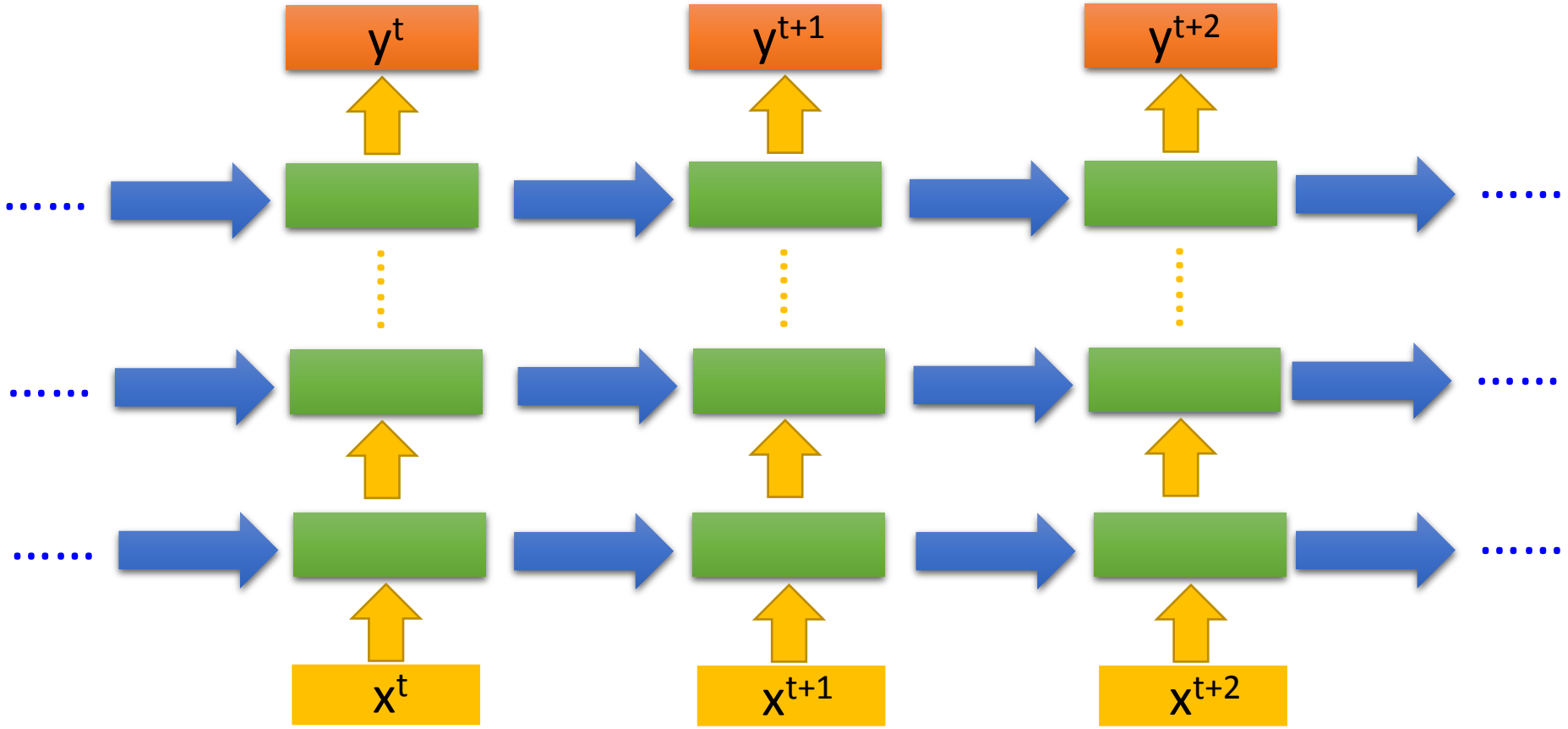
How to train?



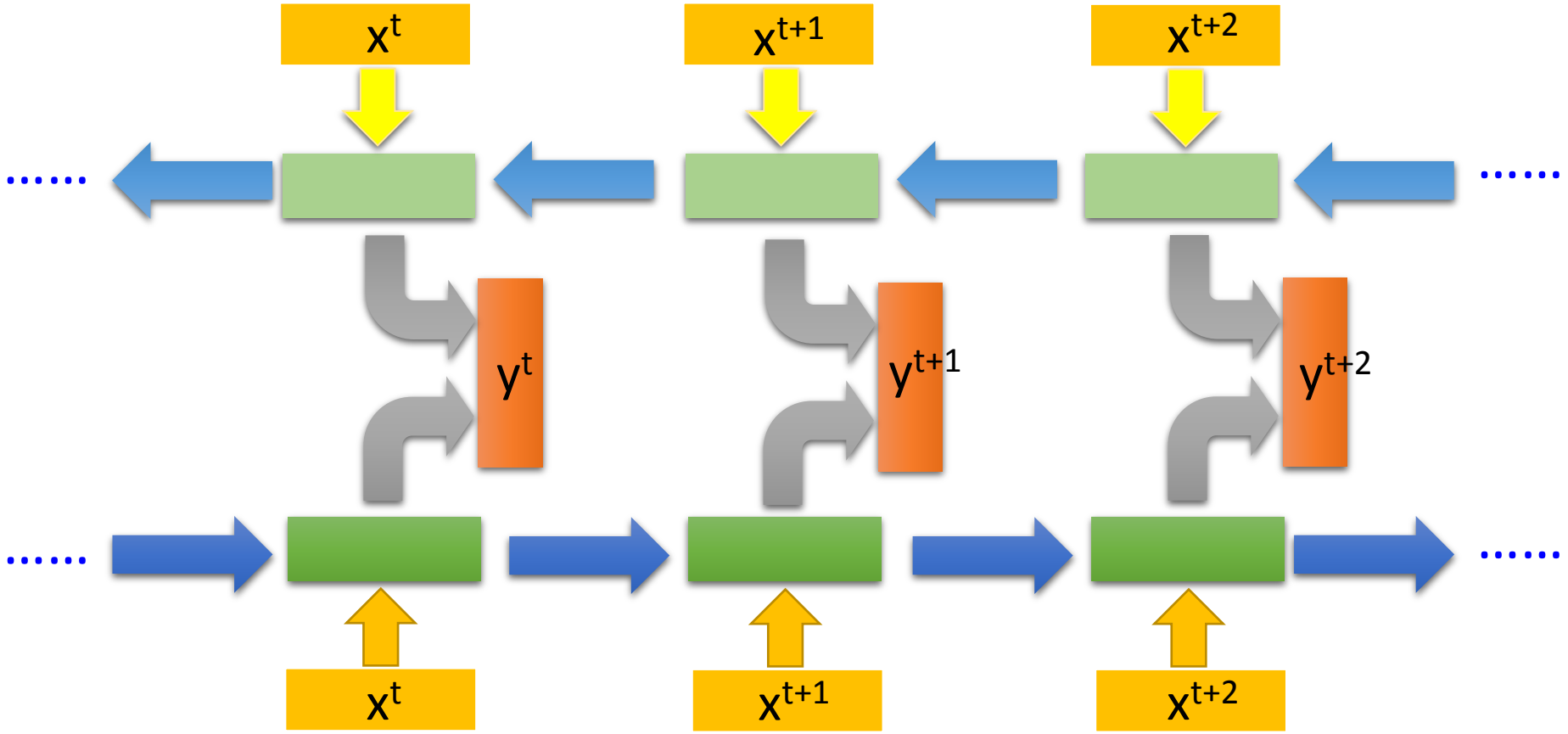
Find the network parameters to minimize the total cost:

Backpropagation through time (BPTT)

Of course it can be deep …

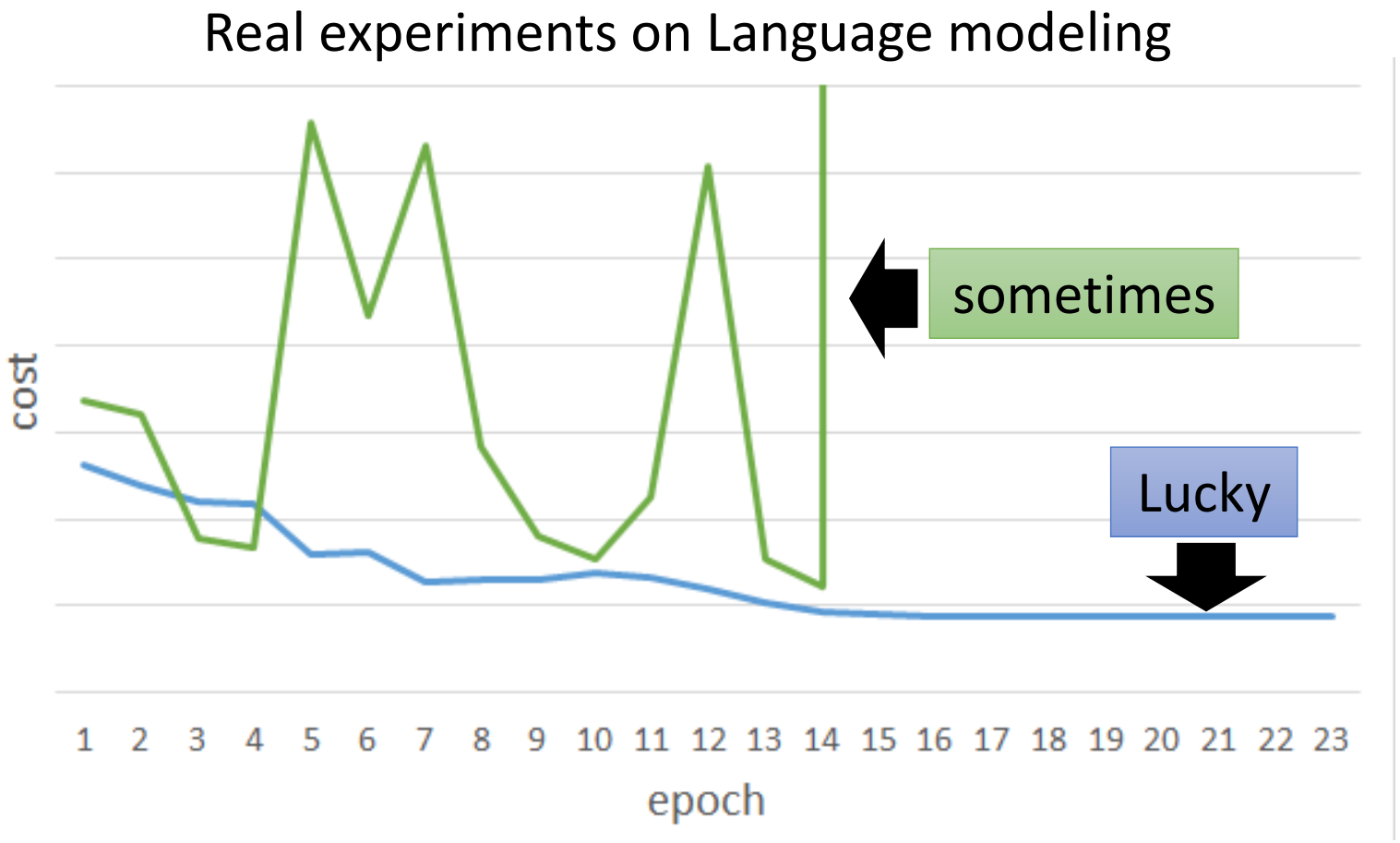


Bidirectional RNN

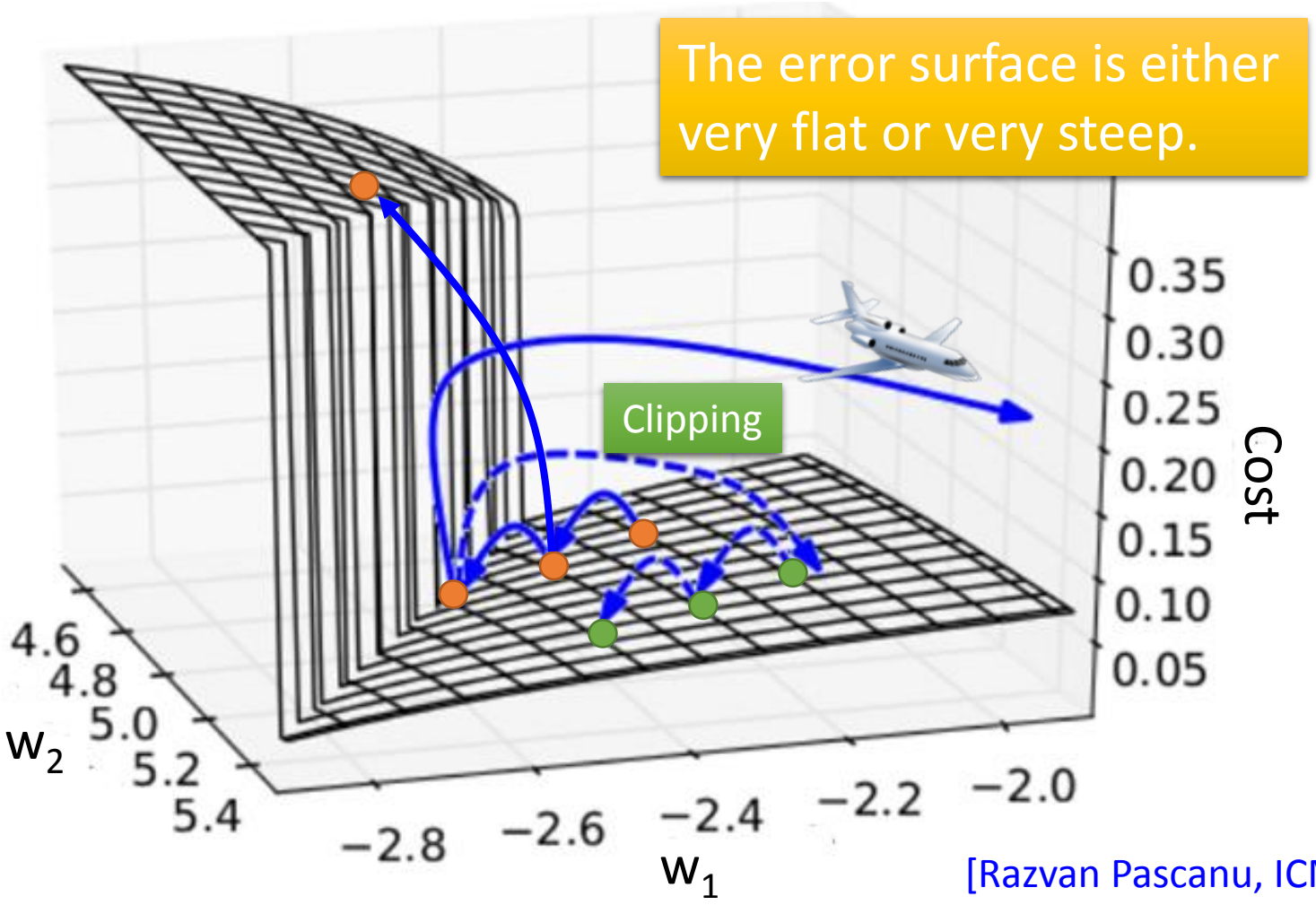


Unfortunately ……

- RNN-based network is not always easy to learn



The error surface is rough.

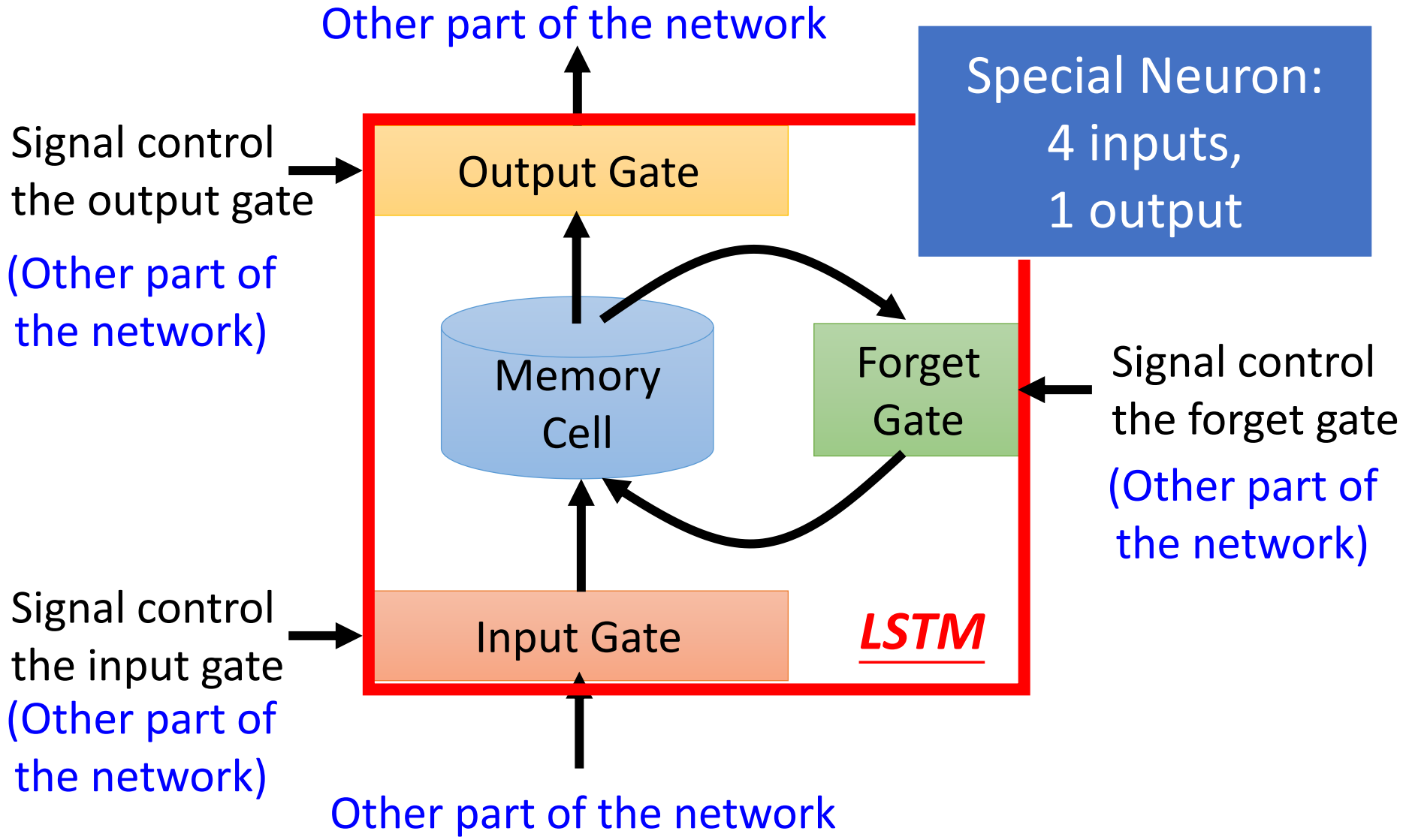


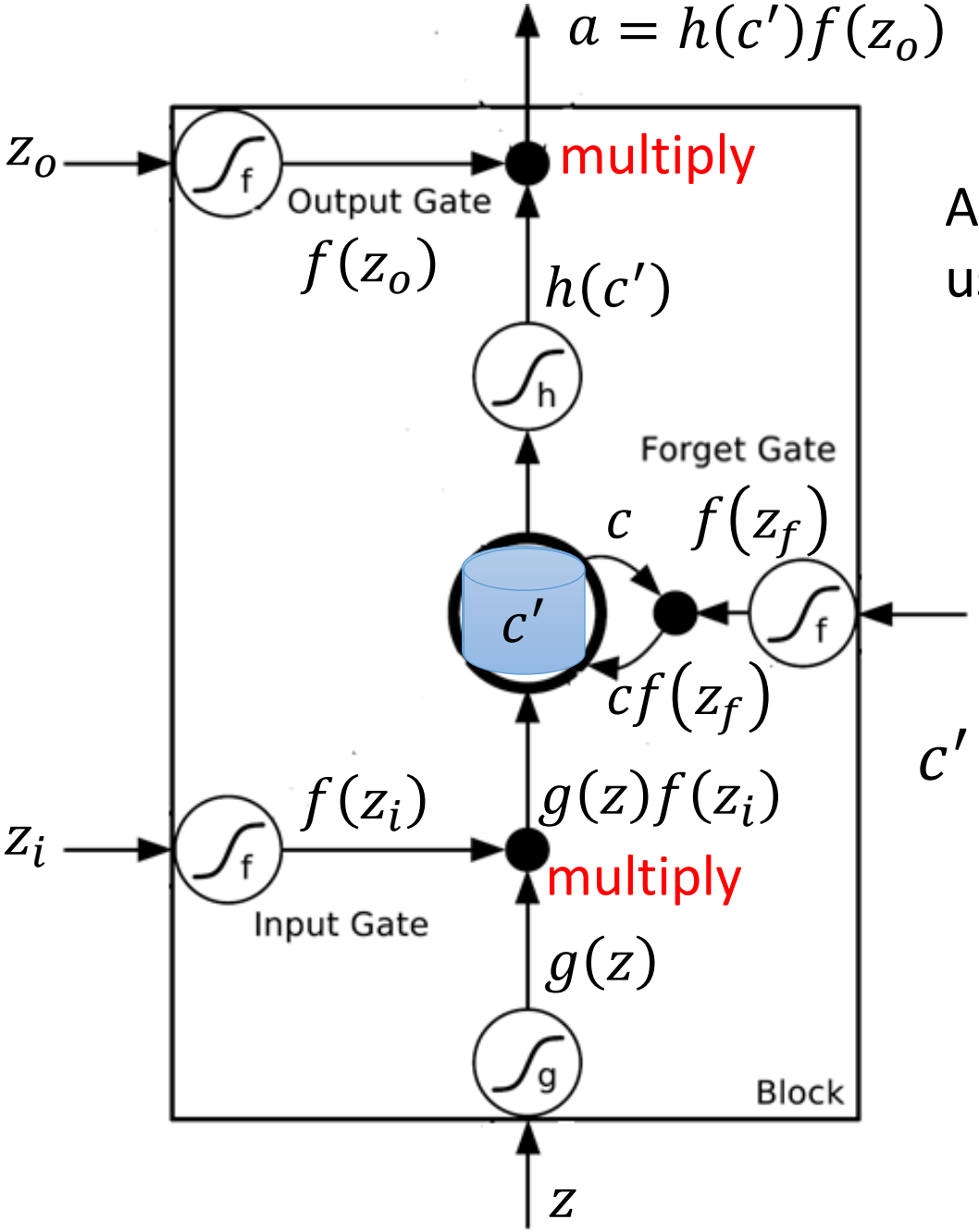
[Razvan Pascanu, ICML'13]

Helpful Techniques

- Nesterov's Accelerated Gradient (NAG):
  - Advance momentum method
- RMS Prop
  - Advanced approach to give each parameter different learning rates
  - Considering the change of Second derivatives
- Long Short-term Memory (LSTM)
  - Can deal with gradient vanishing (not gradient explode)

Long Short-term Memory (LSTM)





Activation function f is usually a sigmoid function

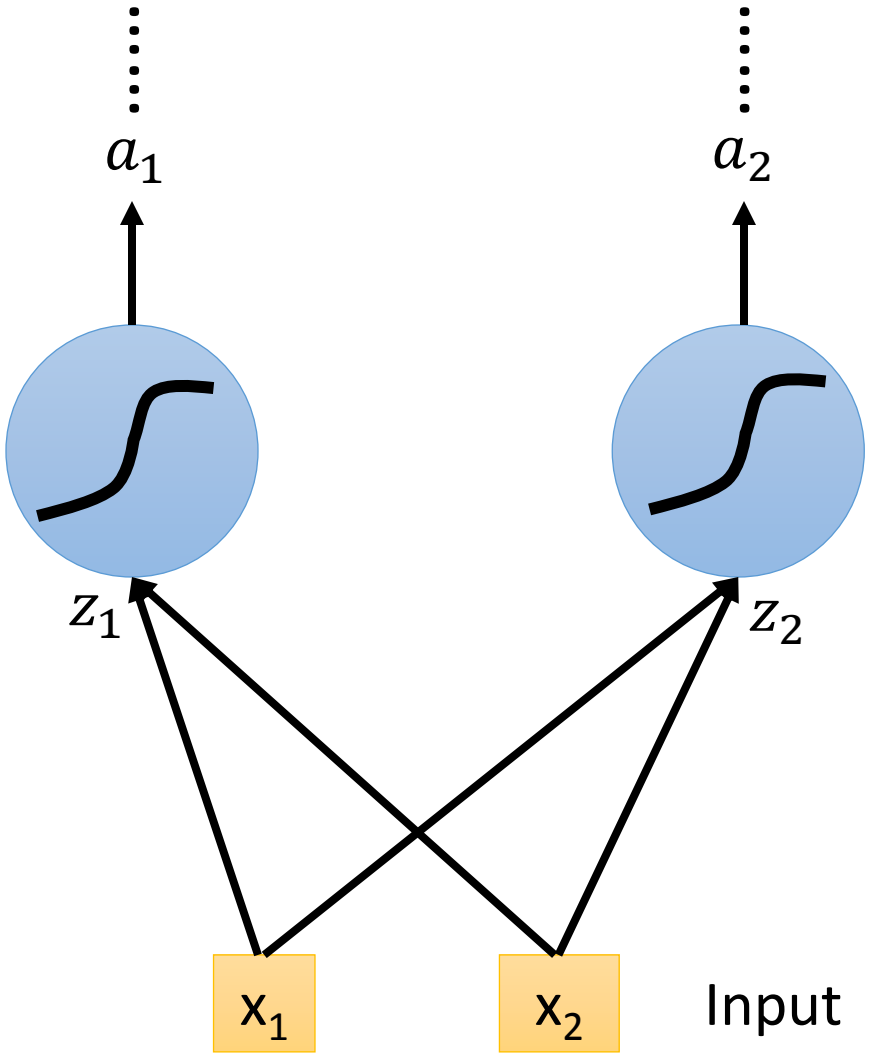
Between 0 and 1

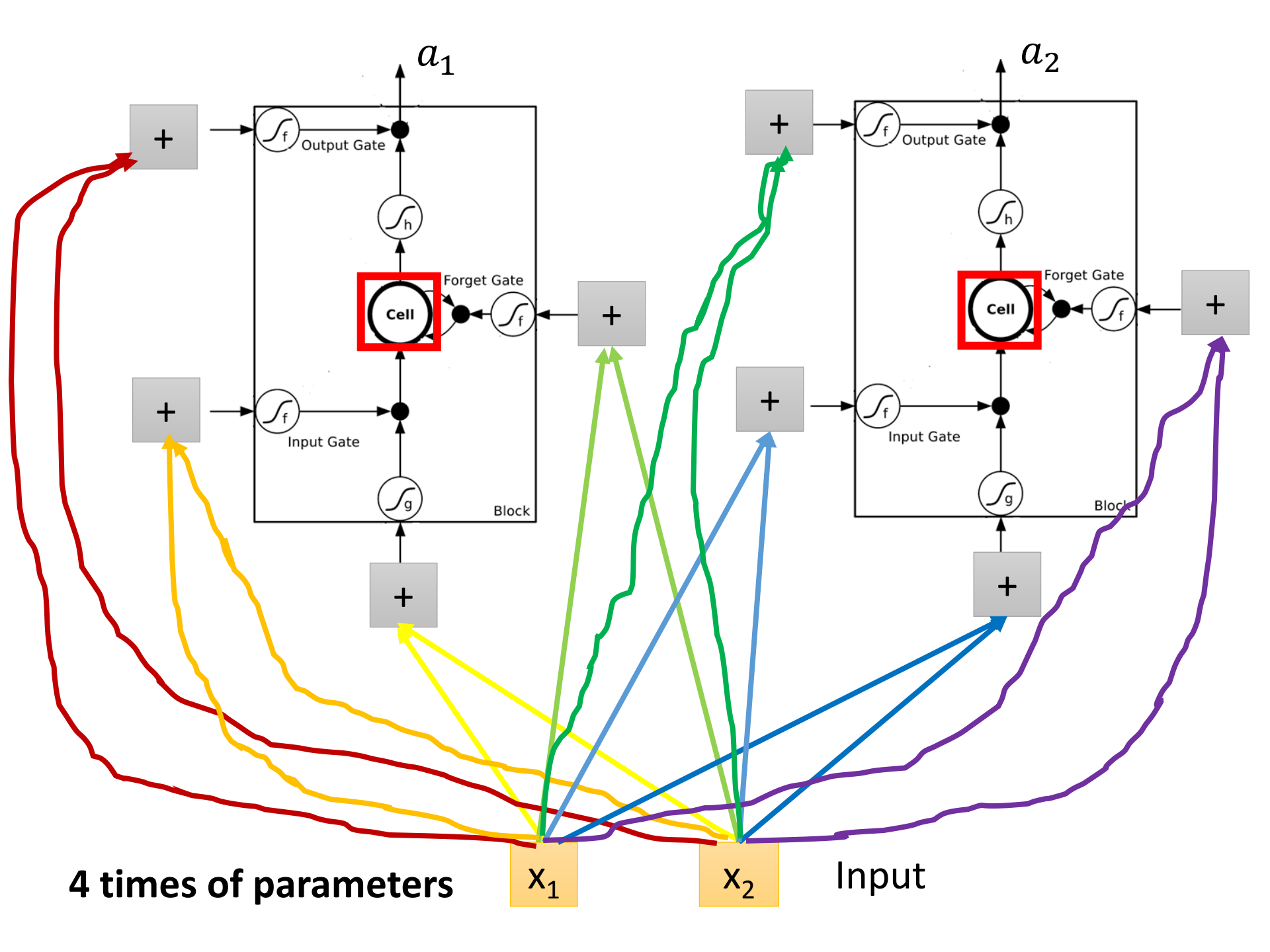
Mimic open and close gate

$$c' = g(z)f(z_i) + cf(z_f)$$

Original Network:

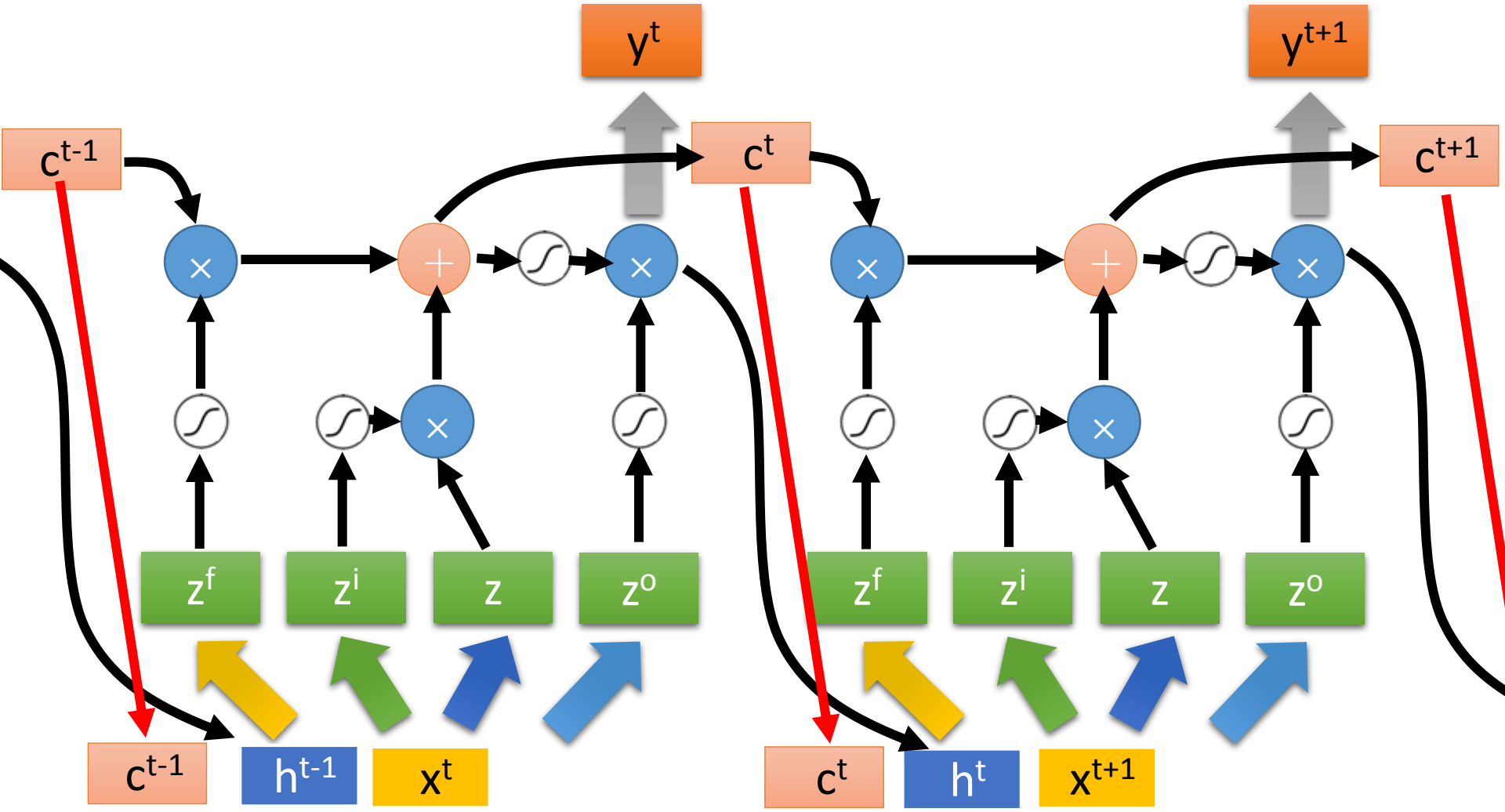
- Simply replace the neurons with LSTM




a_1
a_2
Output Gate
Forget Gate
Input Gate
Cell
Block
x_1
x_2
Input
4 times of parameters

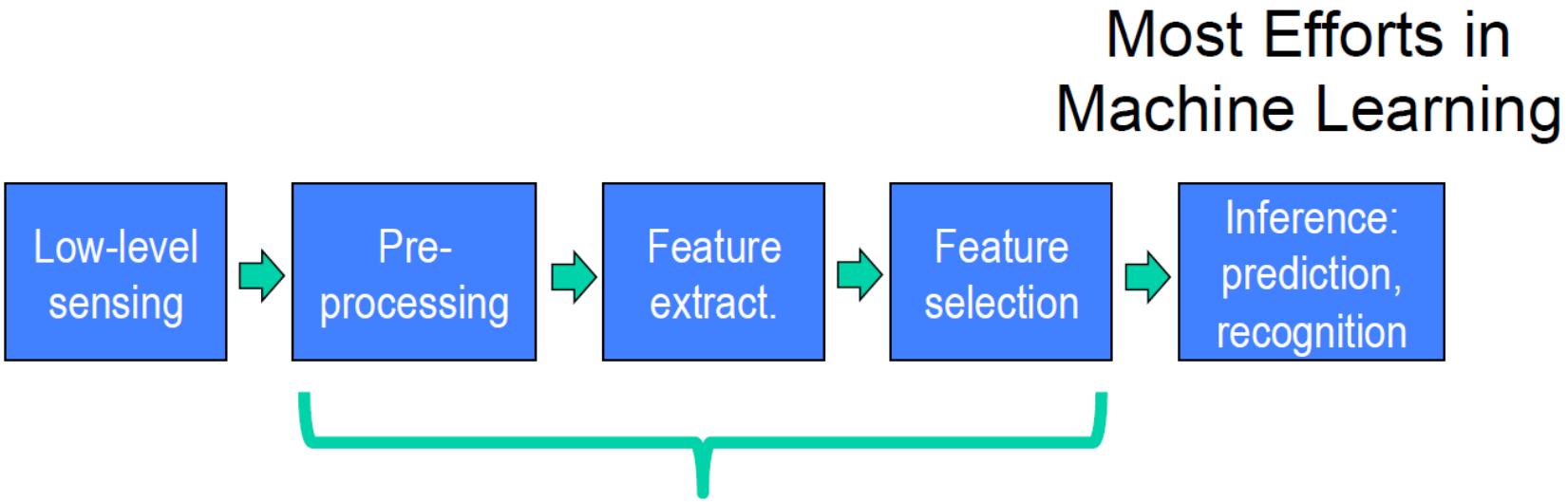
LSTM

Extension: “peephole”



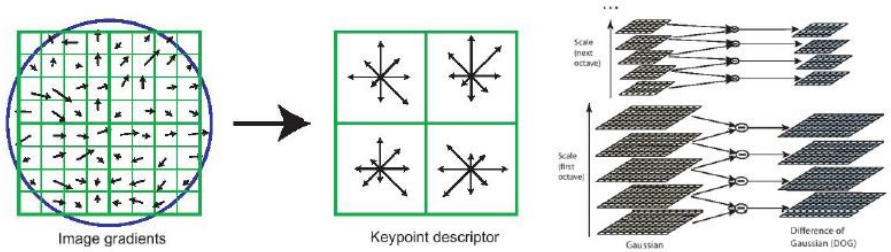
Part V:
Convolution Neural Network

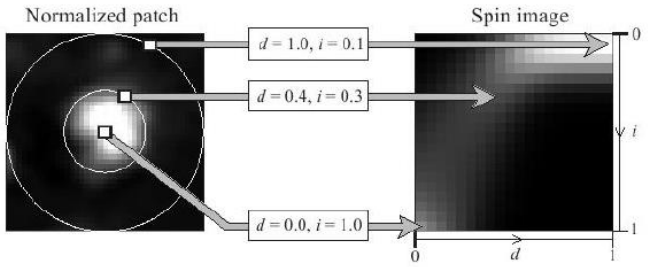
The pipeline of machine visual perception

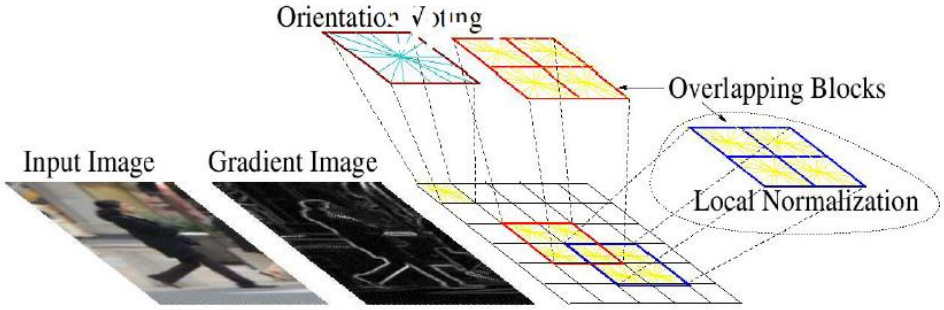


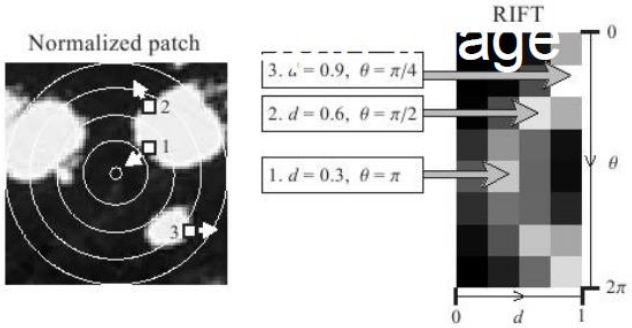
- Most critical for accuracy
- Account for most of the computation for testing
- Most time-consuming in development cycle
- Often hand-craft in practice

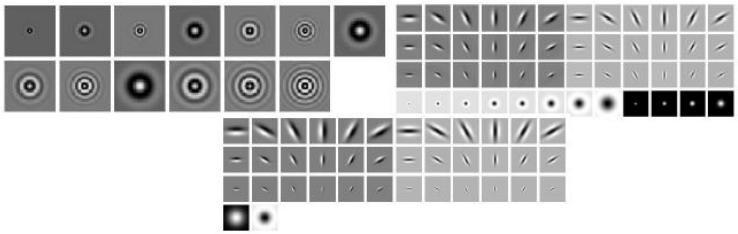
Computer vision features

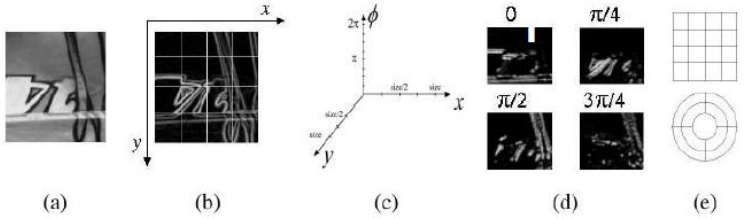




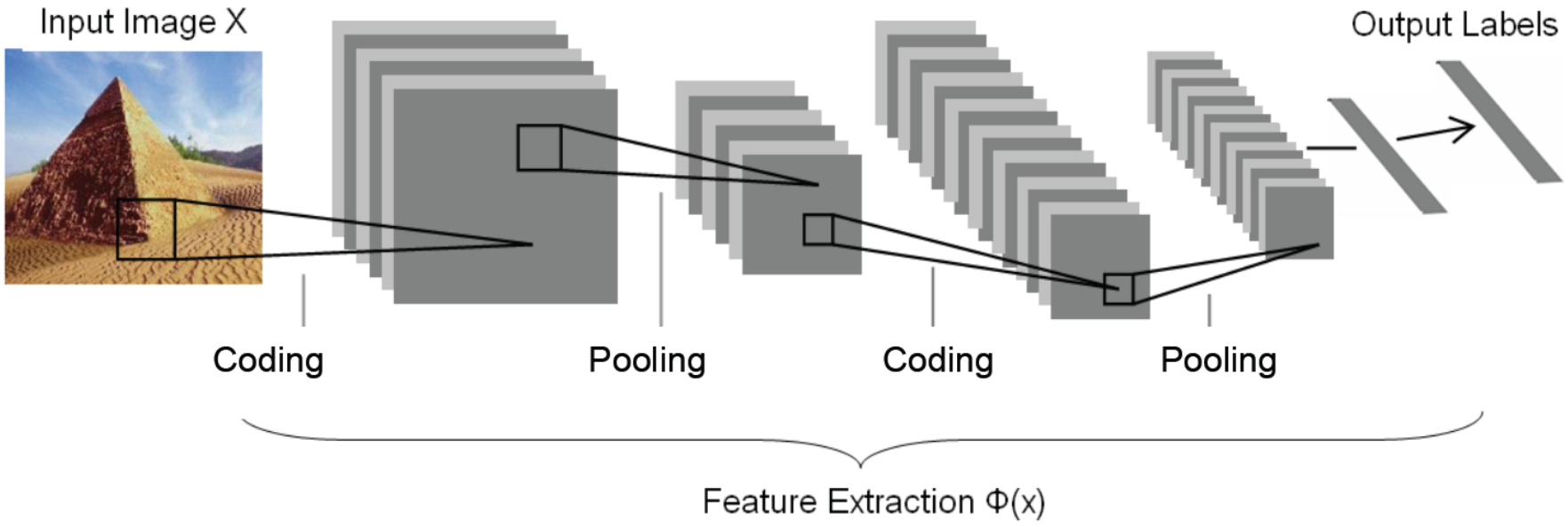






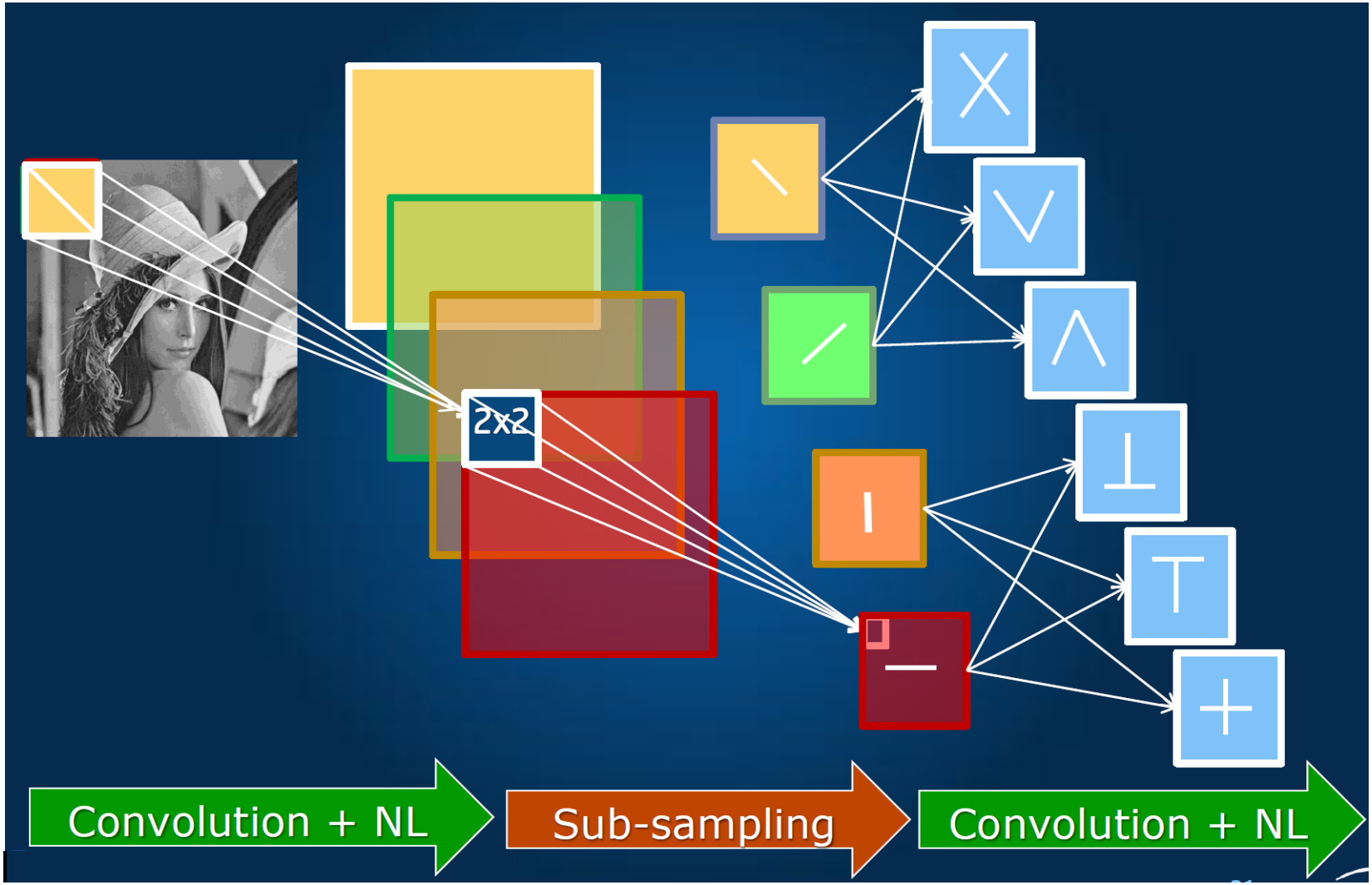


Convolution Neural Networks



Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. Neural Computation, 1989.

Intuitive idea



The Architecture (Geoff Hinton)

- Max-pooling layers follow first, second, and fifth convolutional layers
- The number of neurons in each layer is given by 253440, 186624, 64896, 64896, 43264, 4096, 4096, 1000