6

Neural Network Representation

6.1 Model Representation of MLP

6.2 Activation Function

6.3 Forward Propagation

6.4 Cost Function

6.5 Back Propagation

6

Neural Network Representation

6.6 Initialization

6.7 Dimension of Weight Matrices

6.8 Approximate Numerical Gradients

6.1 Model Representation of Multi-Layer Perceptions (MLP)

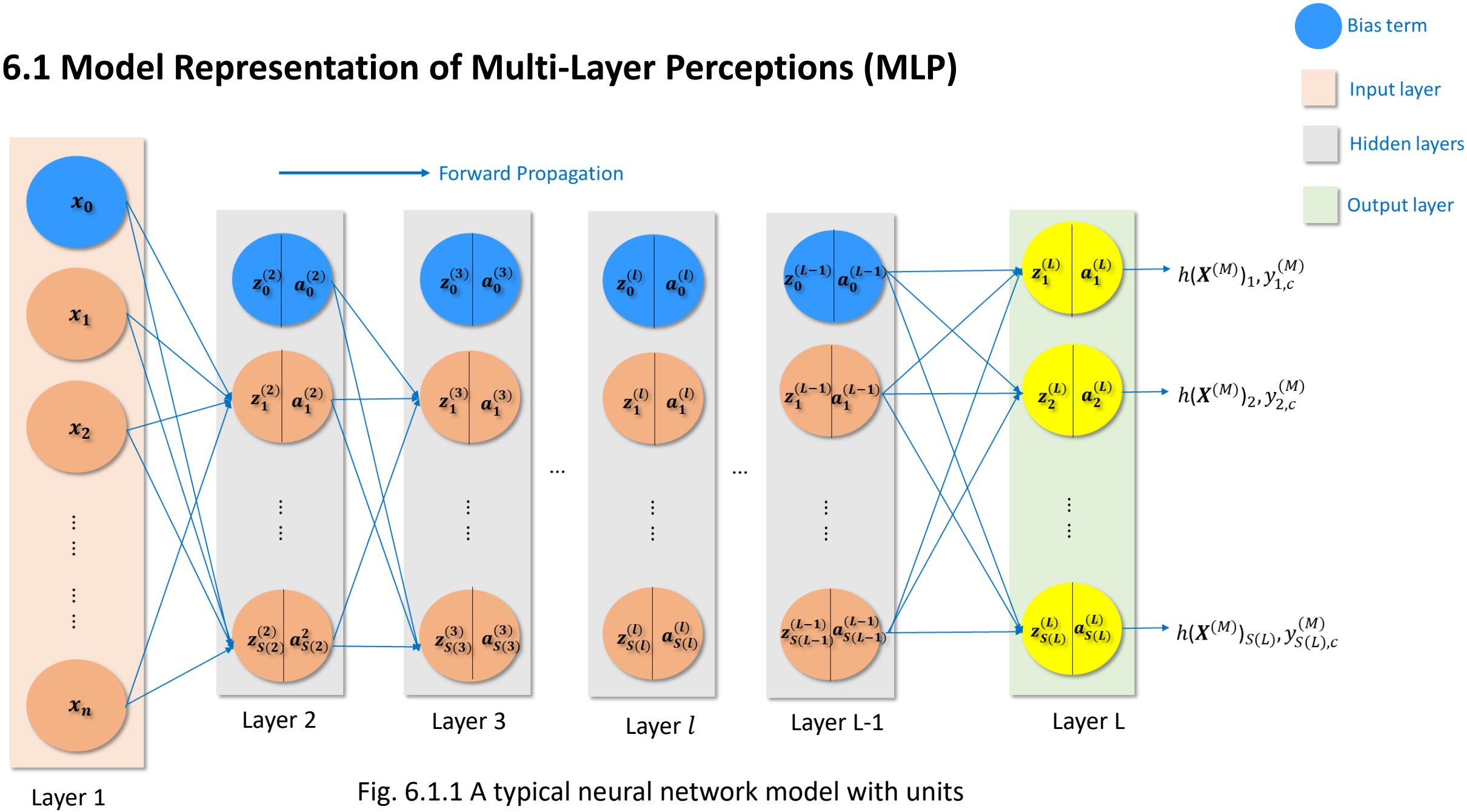


Fig. 6.1.1 A typical neural network model with units

here,

$a_j^{(l)} = \sigma(z_j^{(l)})$ = 'activation' of the j-th unit(or node) in the layer, l.

σ = sigmoid for non-exclusive categories, softmax and linear function for exclusive categories.

$\theta_{ji}^{(l)}$ = a component of the weight that maps from the i-th activation in the layer, l to the j-th 'z' function in the layer, $l+1$.

bias $\theta_{j0}^{(l-1)} related\ term$

$$z_j^{(l)} = \sum_{i=0}^{S(l-1)} \theta_{ji}^{(l-1)} \cdot a_i^{(l-1)} = \theta_{j0}^{(l-1)} \cdot a_0^{(l-1)} + \theta_{j1}^{(l-1)} \cdot a_1^{(l-1)} + \theta_{j2}^{(l-1)} \cdot a_2^{(l-1)} + \cdots\cdots + \theta_{jS(l-1)}^{(l-1)} \cdot a_{S(l-1)}^{(l-1)}$$

$$= \theta_{j1}^{(l-1)} \cdot a_1^{(l-1)} + \theta_{j2}^{(l-1)} \cdot a_2^{(l-1)} + \cdots\cdots + \theta_{jS(l-1)}^{(l-1)} \cdot a_{S(l-1)}^{(l-1)} + \theta_{j0}^{(l-1)} \cdot a_0^{(l-1)}$$

$S(l)$ = the last unit (node) number (not total number of units) in the layer, l.

L = the total number of layers (or the last layer number)

$h(\boldsymbol{X}^{(M)})$ = probability-based prediction vector for the M-th sample, $\boldsymbol{X}^{(M)}$.

$h(\boldsymbol{X}^{(M)})_j$= predicted probability that $\boldsymbol{X}^{(M)}$ belongs to the j-th class: j-th component of $h(\boldsymbol{X}^{(M)})$.

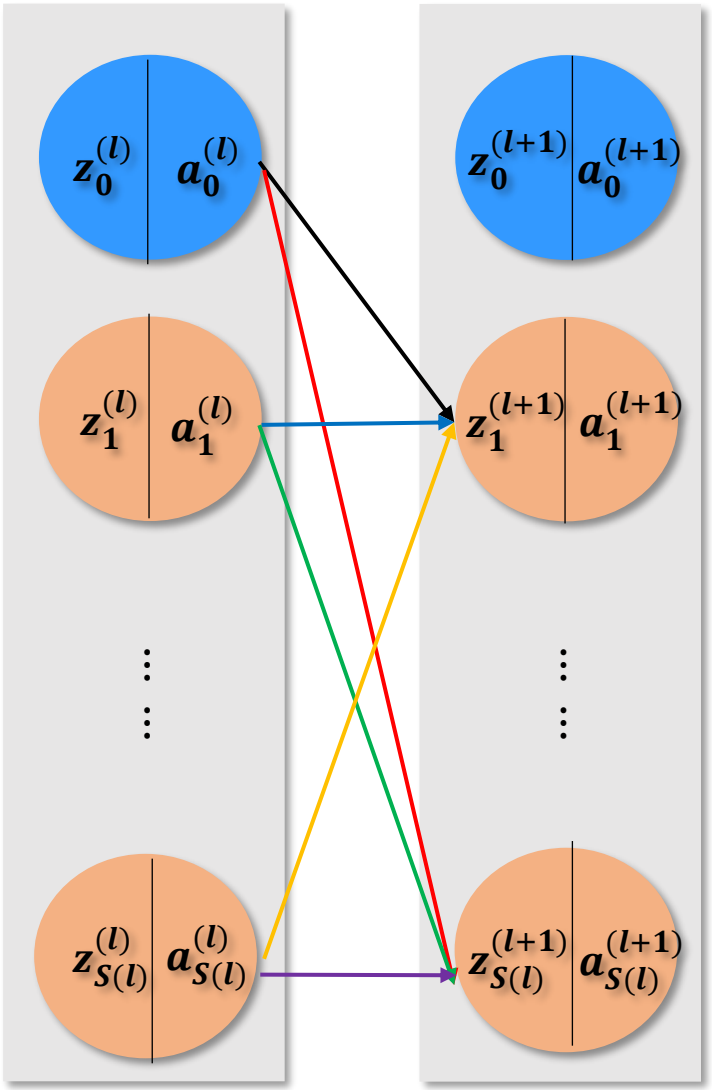
$\boldsymbol{Y}_c^{(M)}$ = probability-based actual output (ground truth) vector of the c-th class for $\boldsymbol{X}^{(M)}$:

'c' should be the class which the input sample belongs to.

$y_{j,c}^{(M)}$ = j-th component of $\boldsymbol{Y}_c^{(M)}$.

M = M-th sample in a given dataset.

c = c-th class which the M-th sample belongs to.



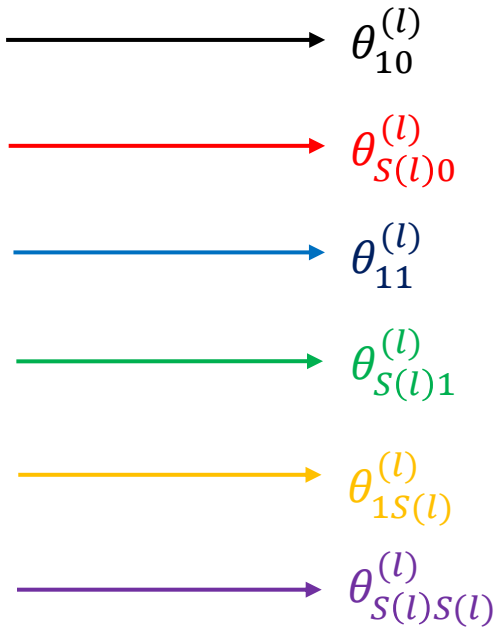


Fig. 6.1.2 A notation of nodes and matrix multiplication

z-functions

From $z_j^{(l)} = \sum_{i=0}^{S(l-1)} \theta_{ji}^{(l-1)} \cdot a_i^{(l-1)}$

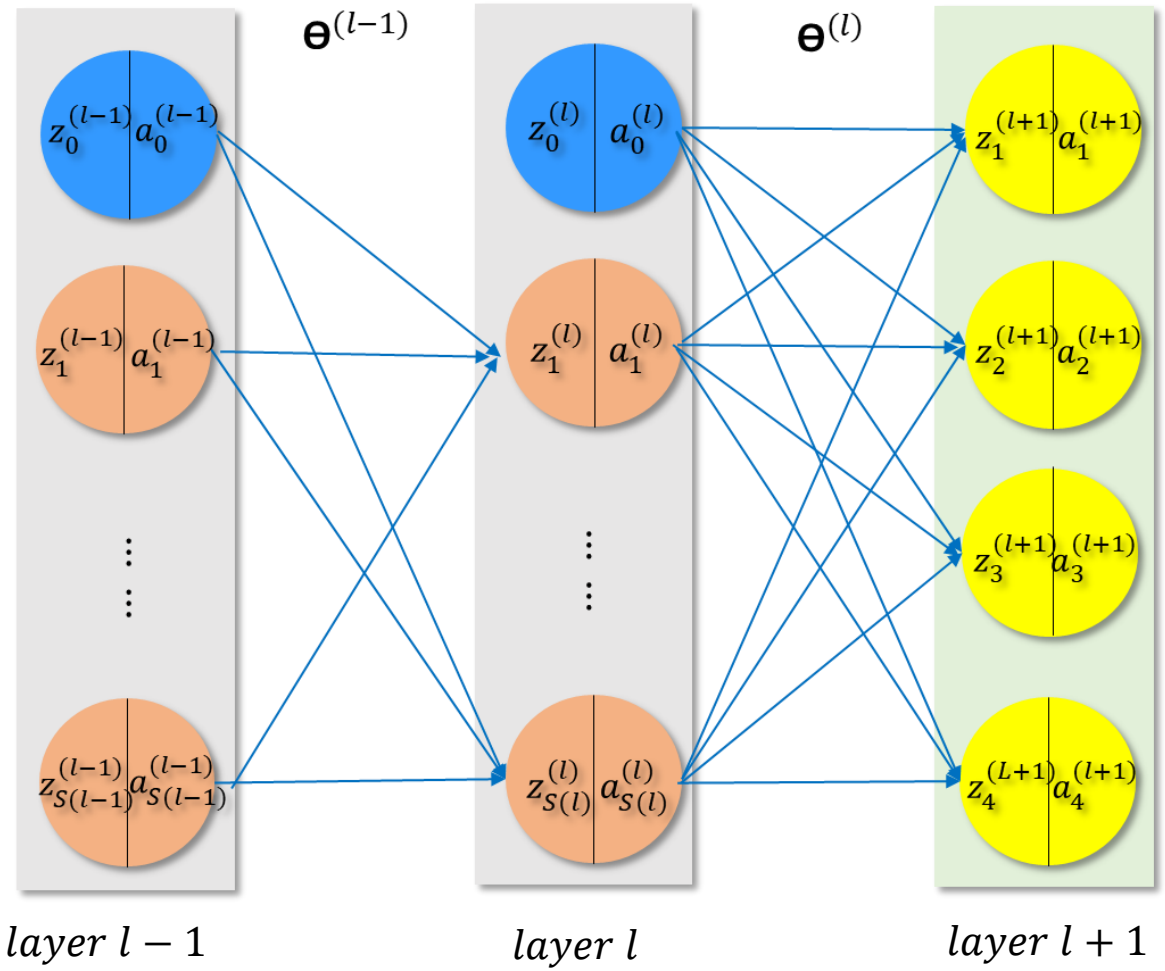


Fig. 6. A part of a neural network

$$z_1^{(l)} = \theta_{10}^{(l-1)} \cdot a_0^{(l-1)} + \theta_{11}^{(l-1)} \cdot a_1^{(l-1)} + \cdots\cdots + \theta_{1S(l-1)}^{(l-1)} \cdot a_{S(l-1)}^{(l-1)}$$

$$z_2^{(l)} = \theta_{20}^{(l-1)} \cdot a_0^{(l-1)} + \theta_{21}^{(l-1)} \cdot a_1^{(l-1)} + \cdots\cdots + \theta_{2S(l-1)}^{(l-1)} \cdot a_{S(l-1)}^{(l-1)}$$

$$\vdots$$

$$z_{S(l)}^{(l)} = \theta_{S(l)0}^{(l-1)} \cdot a_0^{(l-1)} + \theta_{S(l)1}^{(l-1)} \cdot a_1^{(l-1)} + \cdots\cdots + \theta_{S(l)S(l-1)}^{(l-1)} \cdot a_{S(l-1)}^{(l-1)}$$

bias

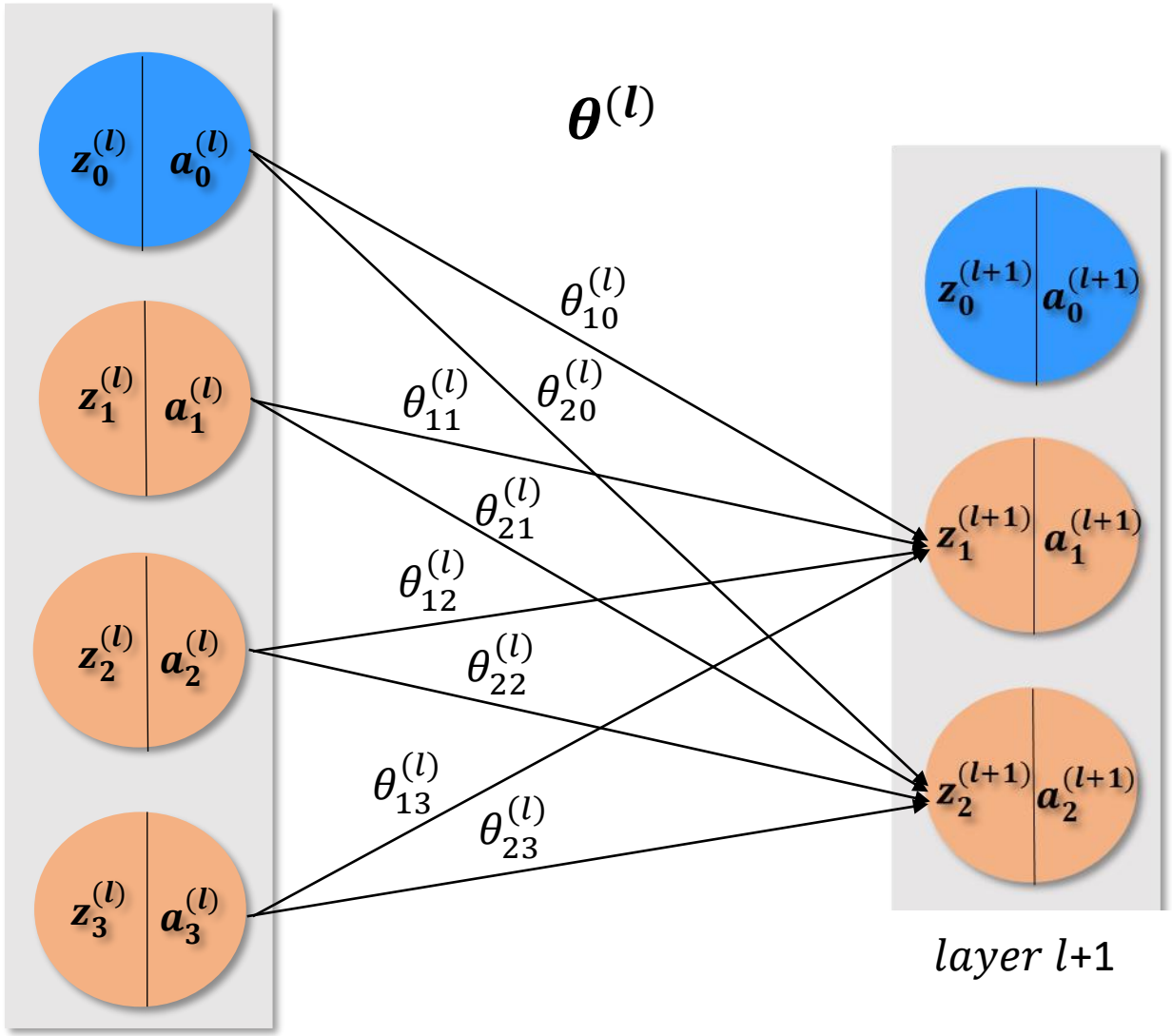
$$z_1^{(l)} = \theta_{11}^{(l-1)} \cdot a_1^{(l-1)} + \cdots\cdots\cdots + \theta_{1S(l-1)}^{(l-1)} \cdot a_{S(l-1)}^{(l-1)} + \theta_{10}^{(l-1)} \cdot a_0^{(l-1)}$$

$$z_2^{(l)} = \theta_{21}^{(l-1)} \cdot a_1^{(l-1)} + \cdots\cdots\cdots + \theta_{2S(l-1)}^{(l-1)} \cdot a_{S(l-1)}^{(l-1)} + \theta_{20}^{(l-1)} \cdot a_0^{(l-1)}$$

$$\vdots$$

$$z_{S(l)}^{(l)} = \theta_{S(l)1}^{(l-1)} \cdot a_1^{(l-1)} + \cdots\cdots + \theta_{S(l)S(l-1)}^{(l-1)} \cdot a_{S(l-1)}^{(l-1)} + \theta_{S(l)0}^{(l-1)} \cdot a_0^{(l-1)}$$

(1) Notation of multiplication of $\boldsymbol{\theta}^{(l)}$ and $\boldsymbol{A}^{(l)}$



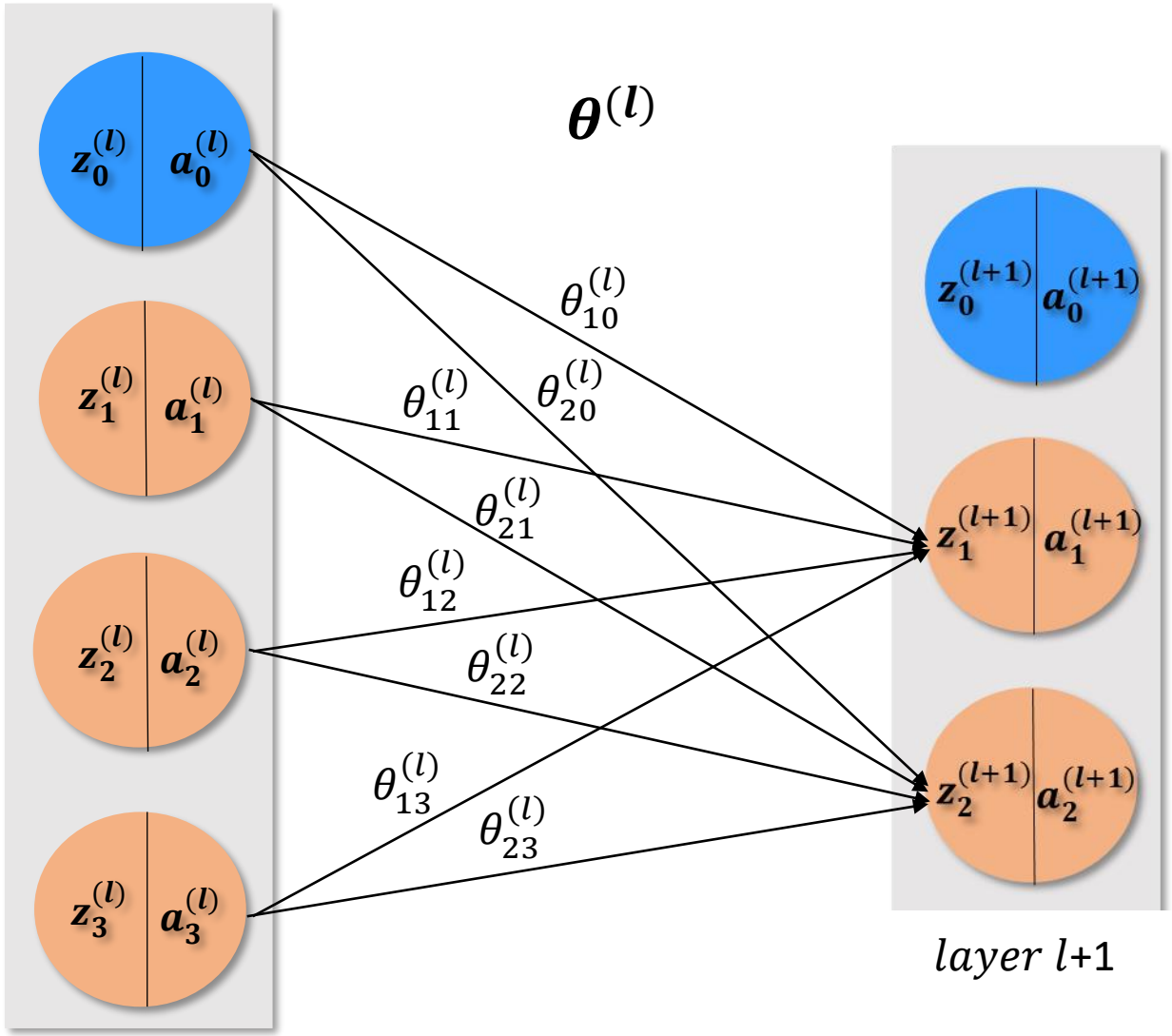
$$\boldsymbol{\theta}^{(l)} = \begin{bmatrix} \theta_{10}^{(l)} & \theta_{11}^{(l)} & \theta_{12}^{(l)} & \theta_{13}^{(l)} \\ \theta_{20}^{(l)} & \theta_{21}^{(l)} & \theta_{22}^{(l)} & \theta_{23}^{(l)} \end{bmatrix}$$

$$\boldsymbol{A}^{(l)} = \begin{Bmatrix} a_0^{(l)} \\ a_1^{(l)} \\ a_2^{(l)} \\ a_3^{(l)} \end{Bmatrix} \quad \boldsymbol{Z}^{(l+1)} = \begin{Bmatrix} z_1^{(l+1)} \\ z_2^{(l+1)} \end{Bmatrix}$$

$$\boldsymbol{Z}^{(l+1)} = \boldsymbol{\theta}^{(l)} \boldsymbol{A}^{(l)}$$

$$= \boldsymbol{A}^{(l)^T} \boldsymbol{\theta}^{(l)^T}$$

(2) Notation of multiplication of $\boldsymbol{\theta}^{(l)}$ and $\boldsymbol{A}^{(l)}$



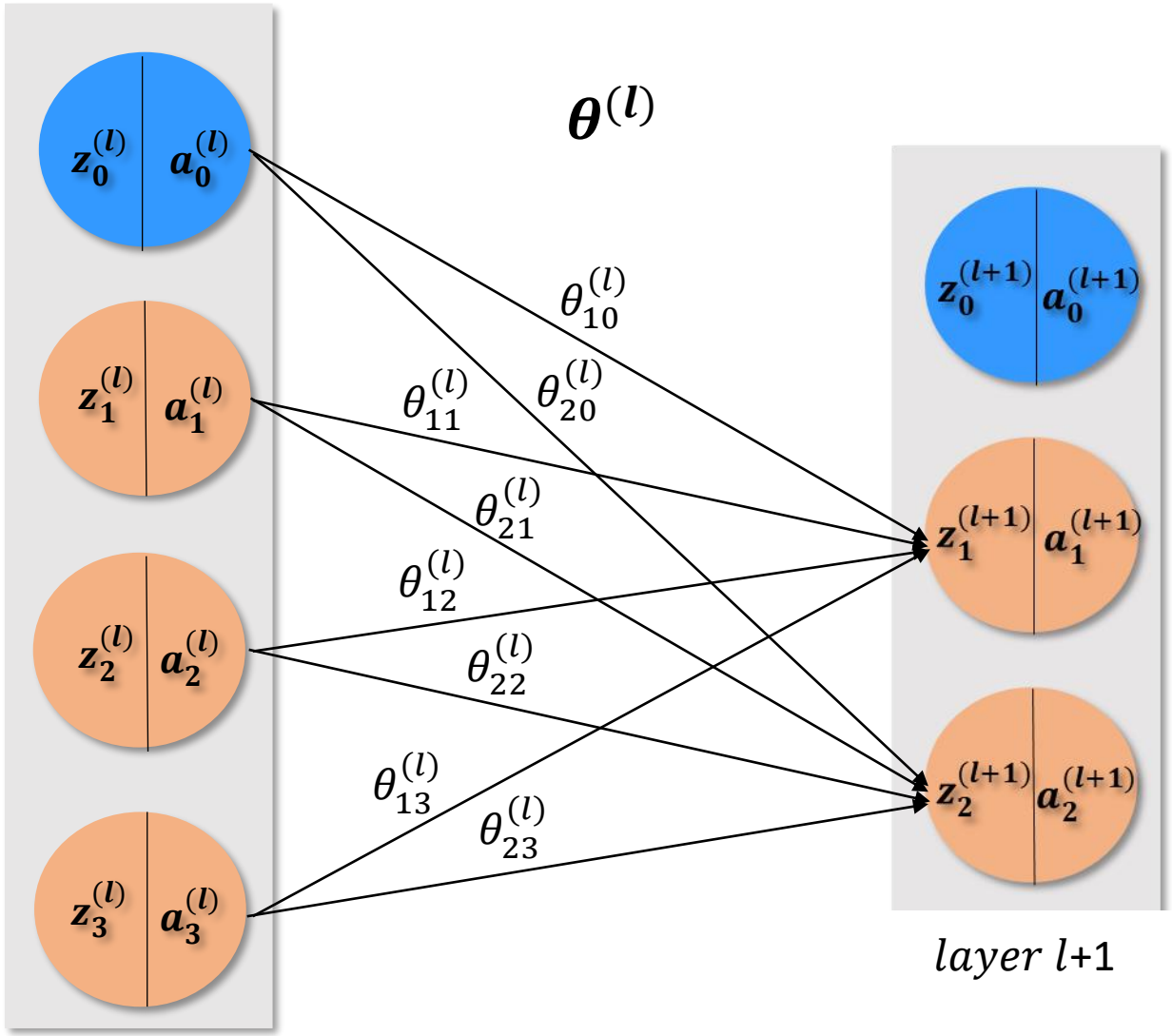
$$\boldsymbol{\theta}^{(l)} = \begin{bmatrix} \theta_{11}^{(l)} & \theta_{12}^{(l)} & \theta_{13}^{(l)} & \theta_{10}^{(l)} \\ \theta_{21}^{(l)} & \theta_{22}^{(l)} & \theta_{23}^{(l)} & \theta_{20}^{(l)} \end{bmatrix}$$

$$\boldsymbol{A}^{(l)} = \begin{Bmatrix} a_1^{(l)} \\ a_2^{(l)} \\ a_3^{(l)} \\ a_0^{(l)} \end{Bmatrix} \quad \boldsymbol{Z}^{(l+1)} = \begin{Bmatrix} z_1^{(l+1)} \\ z_2^{(l+1)} \end{Bmatrix}$$

$$\boldsymbol{Z}^{(l+1)} = \boldsymbol{\theta}^{(l)} \boldsymbol{A}^{(l)}$$

$$= \boldsymbol{A}^{(l)^T} \boldsymbol{\theta}^{(l)^T}$$

(3) Notation of multiplication of $\theta^{(l)}$ and $A^{(l)}$



$$\boldsymbol{\theta}^{(l)} = \begin{bmatrix} \theta_{11}^{(l)} & \theta_{12}^{(l)} & \theta_{13}^{(l)} \\ \theta_{21}^{(l)} & \theta_{22}^{(l)} & \theta_{23}^{(l)} \end{bmatrix}$$

$$\boldsymbol{A}^{(l)} = \begin{Bmatrix} a_1^{(l)} \\ a_2^{(l)} \\ a_3^{(l)} \end{Bmatrix} \quad \boldsymbol{B}^{(l)} = \begin{Bmatrix} \theta_{10}^{(l)} a_0^{(l)} \\ \theta_{20}^{(l)} a_0^{(l)} \end{Bmatrix} \quad \boldsymbol{Z}^{(l+1)} = \begin{Bmatrix} z_1^{(l+1)} \\ z_2^{(l+1)} \end{Bmatrix}$$

$$\boldsymbol{Z}^{(l+1)} = \boldsymbol{\theta}^{(l)} \boldsymbol{A}^{(l)} + \boldsymbol{B}^{(l)}$$

$$= \boldsymbol{A}^{(l)^T} \boldsymbol{\theta}^{(l)^T} + \boldsymbol{B}^{(l)^T}$$

What is the design of neural networks?

(1) Number of inputs: dimension of features, $\boldsymbol{X}^{(i)}$:

(2) Number of output units: It is up to the number of classes.

(3) Number and size of hidden layers: the size of matrices ($\boldsymbol{\Theta}^{(l)}$) for weights depends on the unit number of hidden layers. At least more than 1 hidden layer with the same number of hidden units is recommended.

(4) Determine hyper-parameters.

6.2 Activation Functions

What is an activation function?

In neural networks, an activation function of an unit (node) defines the output of the unit for a given set of inputs.

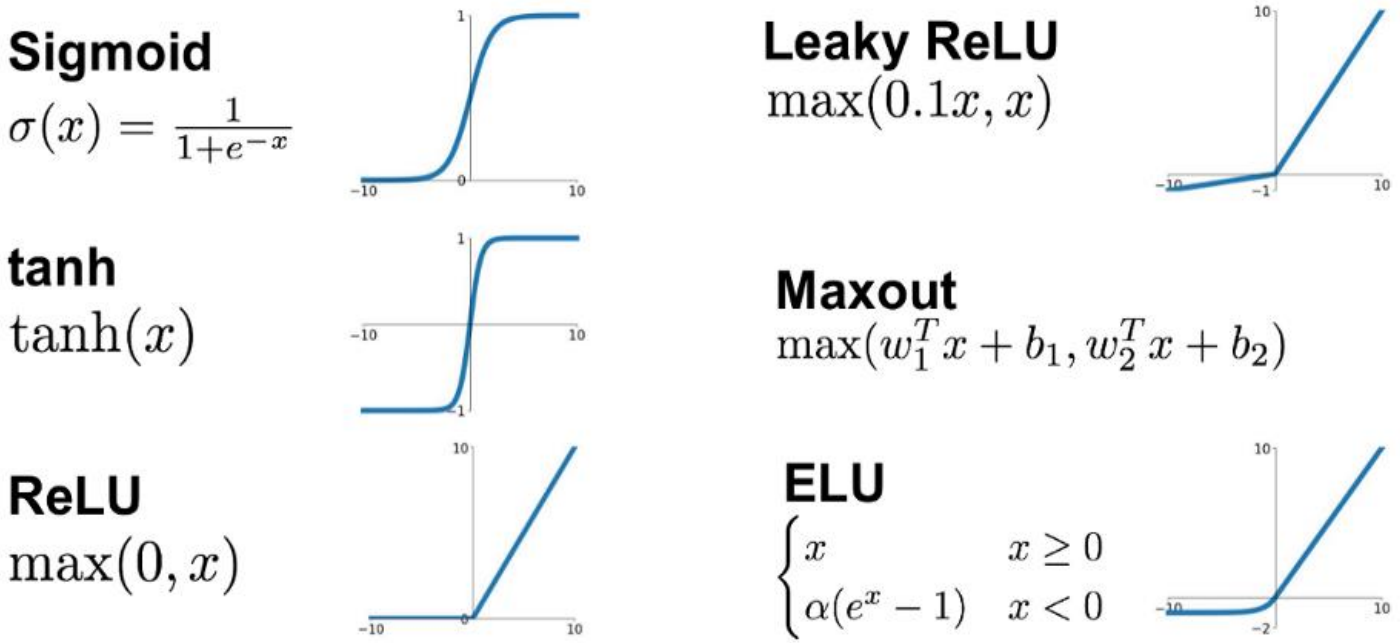


Fig. 6.2 Various activation functions

Where is the motivation for an application of the activation functions come from?

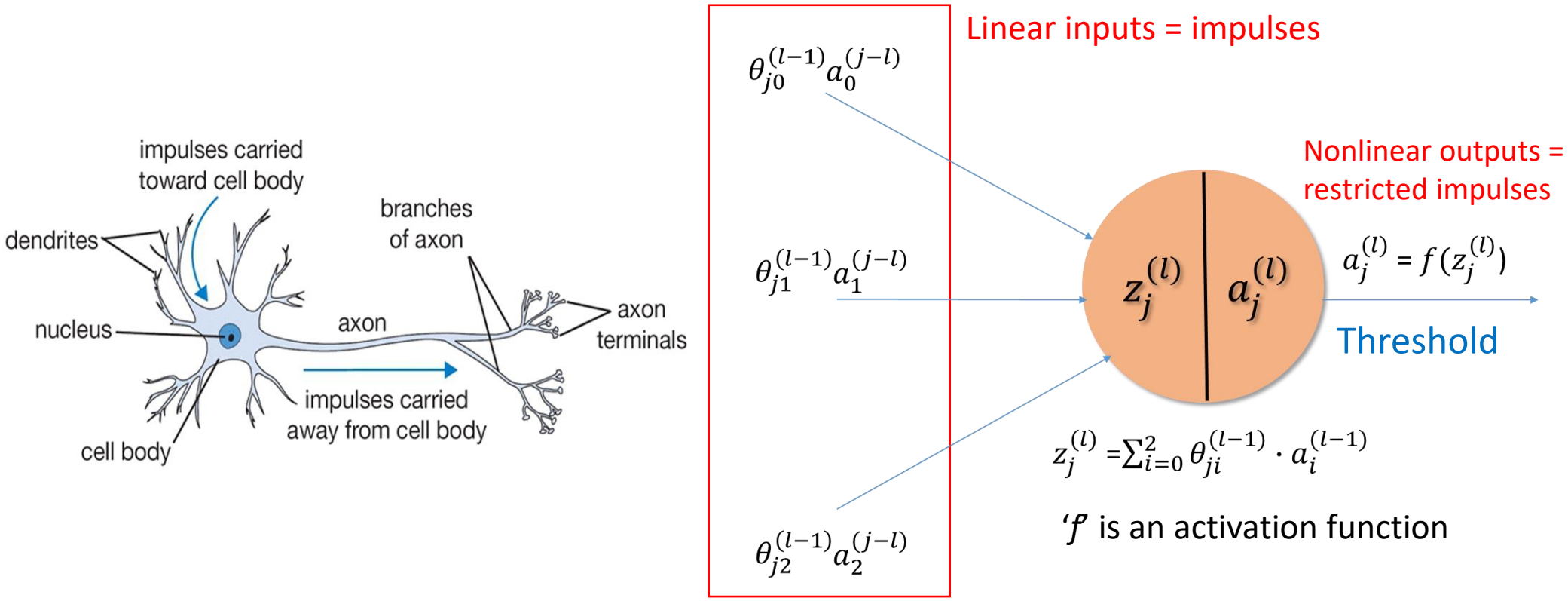


Fig. 6.3 A biological neuron and its mathematical model

What is the main roll of activation functions?

- A neuron receive signals and carries a certain limited quantity of them toward a cell body as per our requirement: A extremely weak signal (impulse) can be removed and an extremely strong signal (impulse) can be restricted by an activated neuron. It seems to look like a biological protection system because an extremely weak signal can give meaningless information, and a super strong signal can shock a human body.
- In similar way, **activation functions decide what signal (input) is to be activated to next units**.
- Activations add non-linearity into a neural network as converting linear functions into non-linear functions.

6.3 Forward Propagation

Given: $\boldsymbol{X}$ and initialized $\boldsymbol{\Theta}^{(l-1)}$ $\xrightarrow{\text{Find}}$ Unknown: $z_j^{(l)}$ & $a_j^{(l)}$ and $h(\boldsymbol{X}^{(M)})_j$

In tensor notation

$$x_j = a_j^{(1)}$$

$$z_j^{(l)} = \sum_{i=0}^{S(l-1)} \theta_{ji}^{(l-1)} \cdot a_i^{(l-1)} \rightarrow z_j^{(2)} = \sum_{i=0}^{S(1)} \theta_{ji}^{(1)} \cdot a_i^{(1)}$$

$$a_j^{(2)} = \sigma(z_j^{(2)})$$

$(ex$:

$$z_1^{(l)} = \theta_{11}^{(l-1)}\, a_1^{(l-1)} + \theta_{12}^{(l-1)}\, a_2^{(l-1)} + \cdots\cdots + \theta_{1S(1)}^{(l-1)}\, a_{S(1)}^{(l-1)} + \theta_{10}^{(l-1)}\, a_0^{(l-1)}$$

$$z_2^{(l)} = \theta_{21}^{(l-1)}\, a_1^{(l-1)} + \theta_{22}^{(l-1)}\, a_2^{(l-1)} + \cdots\cdots + \theta_{2S(1)}^{(l-1)}\, a_{S(1)}^{(l-1)} + \theta_{20}^{(l-1)}\, a_0^{(l-1)}$$

$$\vdots$$

$$z_{S(2)}^{(l)} = \theta_{s(2)1}^{(l-1)}\, a_1^{(l-1)} + \theta_{s(2)2}^{(l-1)}\, a_2^{(l-1)} + \cdots\cdots + \theta_{S(2)S(1)}^{(l-1)}\, a_{S(1)}^{(l-1)} + \theta_{s(2)0}^{(l-1)}\, a_0^{(l-1)}$$

here, n = the last feature No.. S($l = 1$) = n.

In matrix notation

$$\mathbf{X} = \boldsymbol{A}^{(1)}$$

from $\boldsymbol{Z}^{(l)} = \boldsymbol{\Theta}^{(l-1)}\boldsymbol{A}^{(l-1)} \rightarrow \boldsymbol{Z}^{(2)} = \boldsymbol{\Theta}^{(1)}\boldsymbol{A}^{(1)}$

$$\boldsymbol{A}^{(2)} = \sigma(\boldsymbol{Z}^{(2)})$$

here, $\mathbf{X} = \begin{Bmatrix} x_1 \\ x_2 \\ \cdots \\ x_n \\ x_0 \end{Bmatrix}$, $\boldsymbol{A}^{(l-1)} = \begin{Bmatrix} a_1^{(l-1)} \\ a_2^{(l-1)} \\ \cdots \\ a_{S(l-1)}^{(l-1)} \\ a_0^{(l-1)} \end{Bmatrix}$ and

$$\boldsymbol{\Theta}^{(l-1)} = \begin{bmatrix} \theta_{11}^{(l-1)} & \theta_{12}^{(l-1)} & \cdots & \theta_{1S(l-1)}^{(l-1)} & \theta_{10}^{(l-1)} \\ \theta_{21}^{(l-1)} & \theta_{22}^{(l-1)} & & \theta_{2S(l-1)}^{(l-1)} & \theta_{20}^{(l-1)} \\ \vdots & \vdots & & \vdots & \vdots \\ \theta_{S(l)1}^{(l-1)} & \theta_{S(l)2}^{(l-1)} & \cdots & \theta_{S(l)S(l-1)}^{(l-1)} & \theta_{S(l)0}^{(l-1)} \end{bmatrix}$$

In tensor notation

$$z_j^{(3)} = \sum_{i=0}^{S(2)} \theta_{ji}^{(2)} \cdot a_i^{(2)}$$

$$a_j^{(3)} = \sigma(z_j^{(3)})$$

$$\vdots$$

$$z_j^{(L)} = \sum_{i=0}^{S(L-1)} \theta_{ji}^{(L-1)} \cdot a_i^{(L-1)}$$

$$a_j^{(L)} = \sigma(z_j^{(L)})$$

therefore,

$$h(\boldsymbol{X}^{(M)})_j = a_j^{(L)}$$

In matrix notation

$$\boldsymbol{Z}^{(3)} = \boldsymbol{\Theta}^{(2)} \boldsymbol{A}^{(2)}$$

$$\boldsymbol{A}^{(3)} = \sigma(\boldsymbol{Z}^{(3)})$$

$$\vdots$$

$$\boldsymbol{Z}^{(L)} = \boldsymbol{\Theta}^{(L-1)} \boldsymbol{A}^{(L-1)}$$

$$\boldsymbol{A}^{(L)} = \sigma(\boldsymbol{Z}^{(L)})$$

therefore,

$$\mathrm{H}(\mathbf{X}) = \boldsymbol{A}^{(L)}$$

Example: AND function

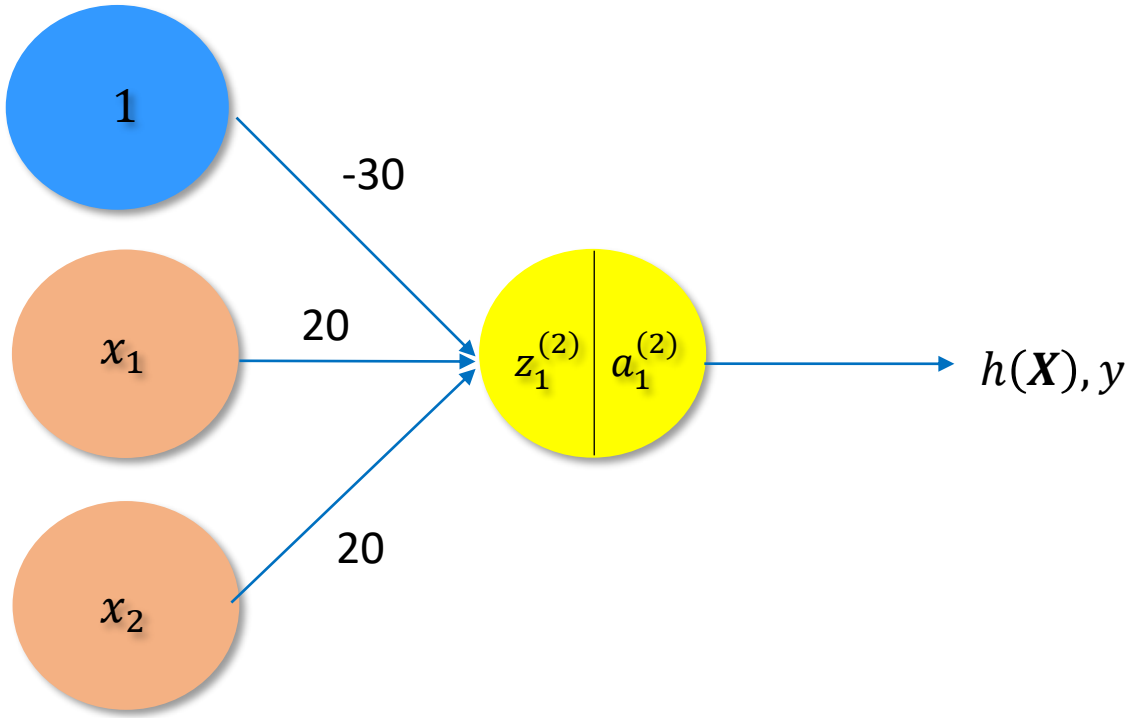


Fig. 6.2 A neural network with a AND function

$$\boldsymbol{X} = \boldsymbol{A}^{(1)} = \{x_1, x_2, 1\}^T$$

$$\boldsymbol{Z}^{(2)} = \boldsymbol{\Theta}^{(1)} \boldsymbol{A}^{(1)}$$

here, $\boldsymbol{\Theta}^{(1)} = \{\theta_{11}^{(1)}, \theta_{12}^{(1)}, \theta_{10}^{(1)}\} = \{20, 20, -30\}$

$$h(\boldsymbol{X}) = \boldsymbol{A}^{(2)} = \sigma(\boldsymbol{Z}^{(2)}) = \sigma\ (20\ x_1 + 20\ x_2 - 30)$$

x_1	x_2	$h(\boldsymbol{X})$
0	0	σ(-30)≈0
0	1	σ(-10) ≈0
1	0	σ(-10) ≈0
1	1	σ(+10) ≈1

Here, σ is sigmoid.

Example: OR function

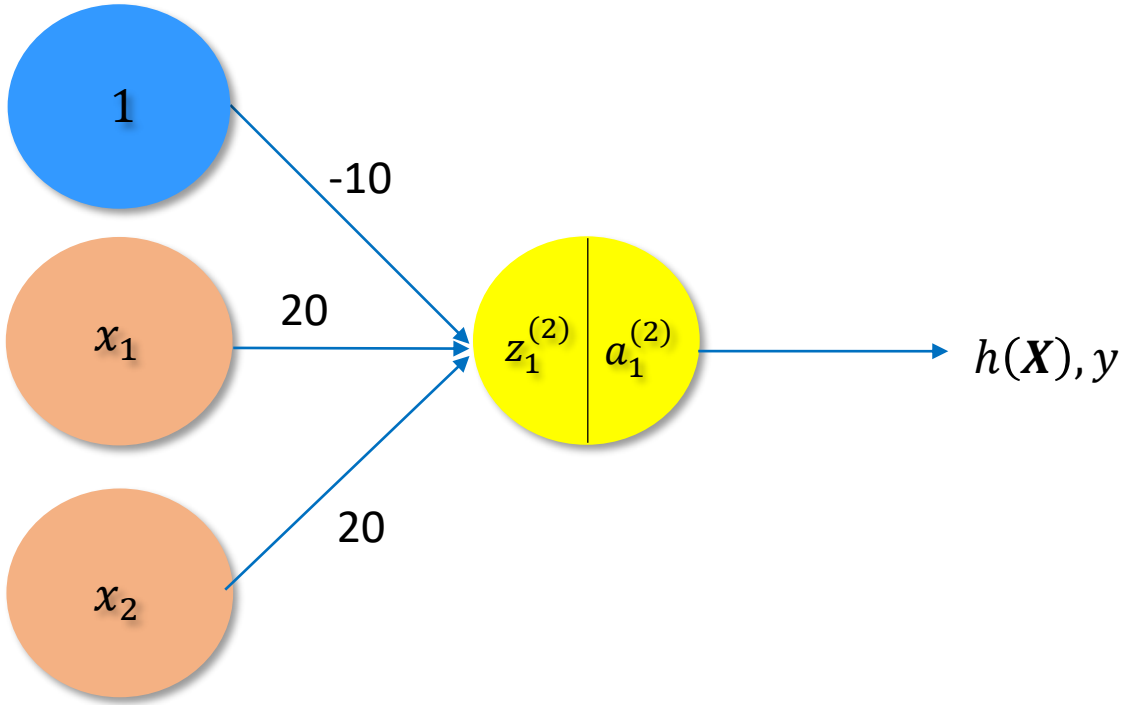


Fig. 6.3 A neural network with a OR function

$$X = A^{(1)} = \{x_1, x_2, 1\}^T$$

$$Z^{(2)} = \Theta^{(1)} A^{(1)}$$

here, $\Theta^{(1)} = \{\theta_{11}^1, \theta_{12}^1, \theta_{10}^1\} = \{20, 20, -10\}$

$$h(X) = A^{(2)} = \sigma(Z^{(2)}) = \sigma\,(20\,x_1 + 20\,x_2 - 10)$$

x_1	x_2	$h(X)$
0	0	σ(-10)$\approx$ 0
0	1	σ(+10) $\approx$ 1
1	0	σ(+10) $\approx$ 1
1	1	σ(+30) $\approx$ 1

Multi-class classification:





Fig. 6.4 An input dataset with 4 classes and 5 examples

- Suppose that each picture has $n \times n + 1$ number of pixels (gray color) and a required neural network should have $n \times n$ number of features.
- In many real classification cases, each picture has $3 \times n \times n$ dimension if they are RGB images.

No. of pictures (M)	X_0	X_1	……	$X_{n\times n}$	Y	Class (c)
1	$x_0^{(1)}$	$x_1^{(1)}$	……	$x_{n\times n}^{(1)}$	$\boldsymbol{Y}_1^{(1)}$	1st (person)
2	$x_0^{(2)}$	$x_1^{(2)}$	……	$x_{n\times n}^{(2)}$	$\boldsymbol{Y}_2^{(2)}$	2nd (car)
3	$x_0^{(3)}$	$x_1^{(3)}$	……	$x_{n\times n}^{(3)}$	$\boldsymbol{Y}_1^{(3)}$	1st (person)
4	$x_0^{(4)}$	$x_1^{(4)}$	……	$x_{n\times n}^{(4)}$	$\boldsymbol{Y}_3^{(4)}$	3rd (motor-cycle)
5	$x_0^{(5)}$	$x_1^{(5)}$	……	$x_{n\times n}^{(5)}$	$\boldsymbol{Y}_4^{(5)}$	4th (truck)

here, $\boldsymbol{Y}_1^{(M)} = \begin{Bmatrix}1\\0\\0\\0\end{Bmatrix}$: person. $\boldsymbol{Y}_2^{(M)} = \begin{Bmatrix}0\\1\\0\\0\end{Bmatrix}$: car. $\boldsymbol{Y}_3^{(M)} = \begin{Bmatrix}0\\0\\1\\0\end{Bmatrix}$: motorcycle. $\boldsymbol{Y}_4^{(M)} = \begin{Bmatrix}0\\0\\0\\1\end{Bmatrix}$: truck.

$$\boldsymbol{X}^{(M)} = \left\{x_0^{(M)}, x_1^{(M)}, \dots, x_{n\times n}^{(M)}\right\} = \left\{x_1^{(M)}, \dots, x_{n\times n}^{(M)}, x_0^{(M)}\right\}$$

A training dataset for given features: $\{\boldsymbol{X}^{(M)}, \boldsymbol{Y}_c^{(M)}\} = \{(\boldsymbol{X}^{(1)}, \boldsymbol{Y}_1^{(1)}), (\boldsymbol{X}^{(2)}, \boldsymbol{Y}_2^{(2)}), (\boldsymbol{X}^{(3)}, \boldsymbol{Y}_1^{(3)}), (\boldsymbol{X}^{(4)}, \boldsymbol{Y}_3^{(4)}), (\boldsymbol{X}^{(5)}, \boldsymbol{Y}_4^{(5)})\}$

M = M-th example. c = c-th class which M-th sample belongs to. m = 5 and the dataset has 4 classes.

Neural Network

Goal: Do forward propagation to get $h(\boldsymbol{X}^{(M)})_j \approx y_{j,c}^{(M)}$

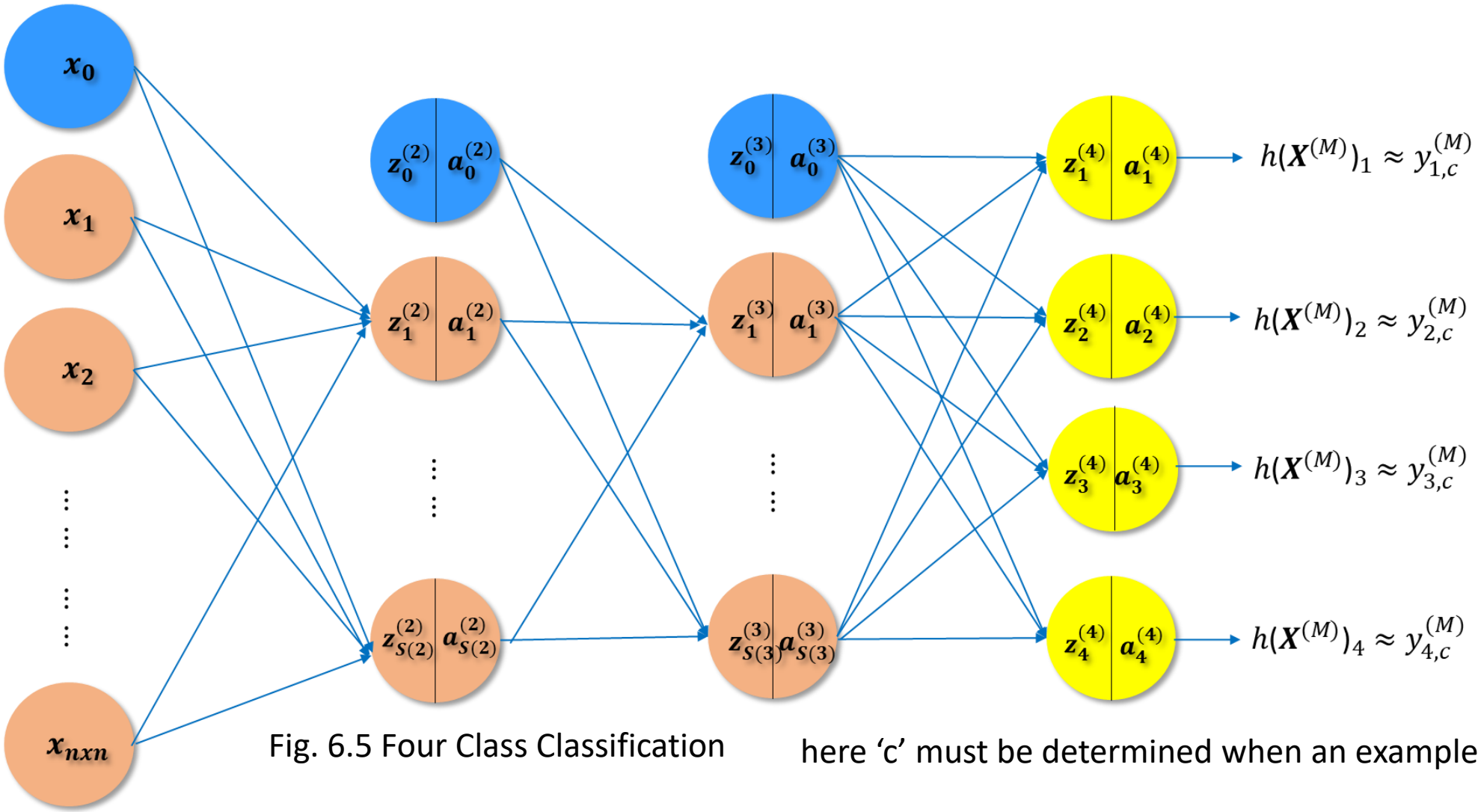


Fig. 6.5 Four Class Classification

here ‘c’ must be determined when an example is picked up.

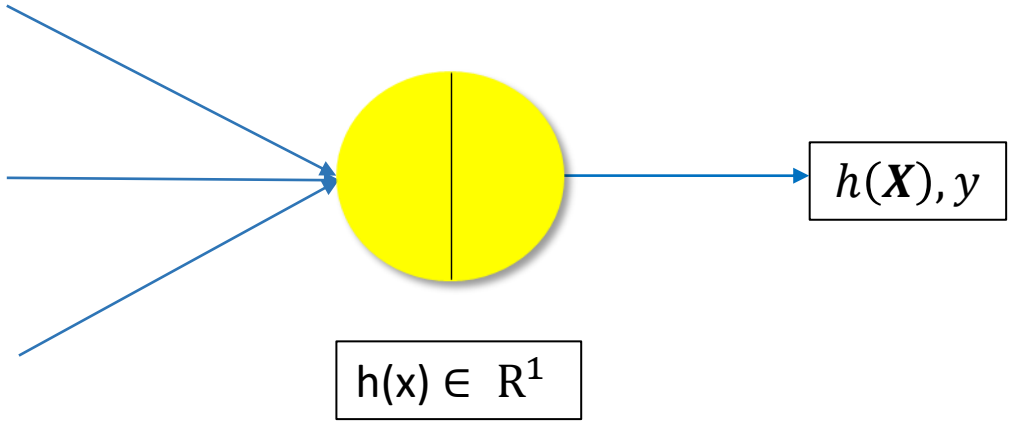


Fig. 6.6 Binary classification (y = 0 or 1)

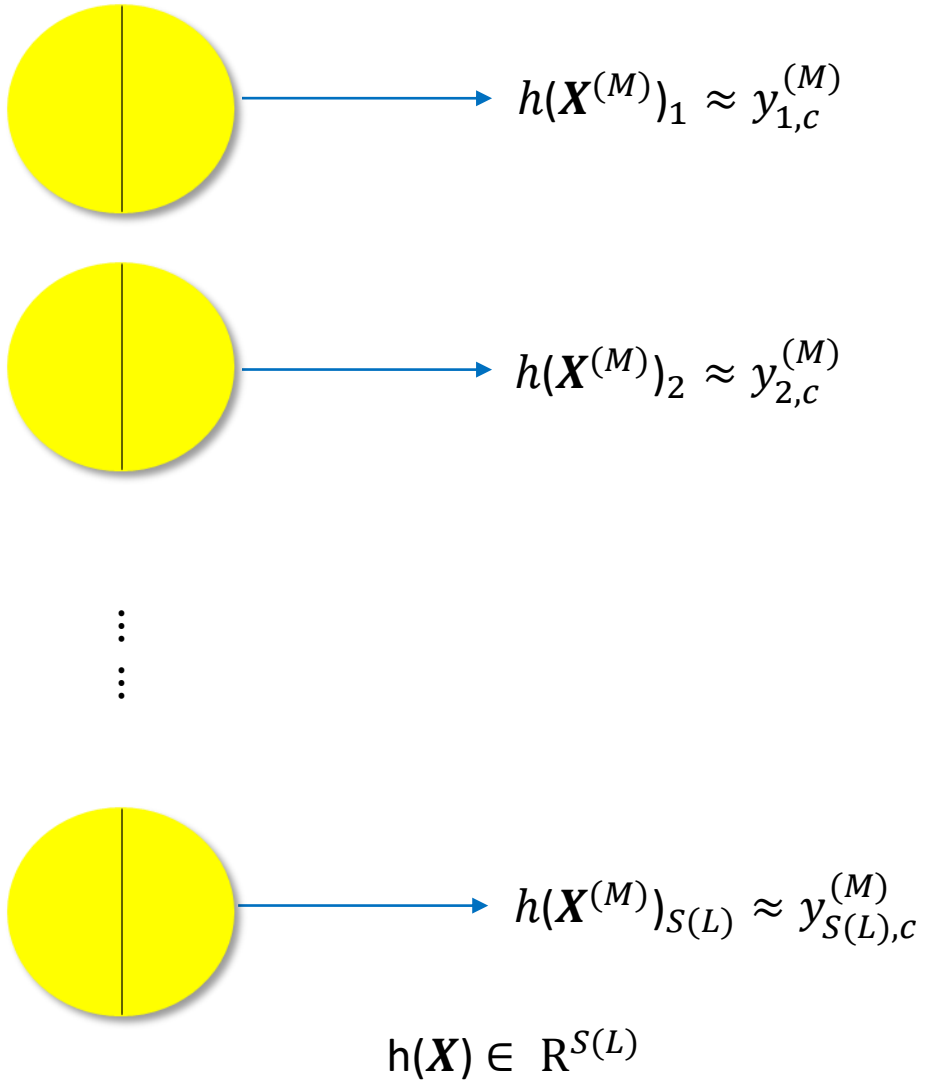


Fig. 6.7 Multi-class classification

6.4 Cost Function

Logistic regression: Cross-Entropy loss

$$J(\Theta) = -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log\left(h\left(\mathbf{X}^{(i)}\right)\right) + (1- y^{(i)}) \log(1-h\left(\mathbf{X}^{(i)}\right))] + \frac{\lambda}{2m}\sum_{j=1}^{n} \theta_j^2$$

Logistic regression for neural network: Cross-Entropy loss

$$J(\Theta) = -\frac{1}{m}[\sum_{M=1}^{m}\sum_{j=1}^{S(L)} y_{j,c}^{(M)} \log\left(h(\boldsymbol{X}^{(M)})_j\right) + (1 -y_{j,c}^{(M)}) \log(1-h(\boldsymbol{X}^{(M)})_j)] + \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{S(l)}\sum_{j=1}^{S(l+1)} (\theta_{ji}^{l})^2$$

6.5 Back Propagation

Forward propagation

Given: $\boldsymbol{X}$ and initialized $\boldsymbol{\Theta}^{(l-1)}$ Find Unknown: $z_j^{(l)}$ & $a_j^{(l)}$ and $h(\boldsymbol{X}^{(M)})_j$

$$\boldsymbol{Y}_1^{(M)} = \begin{Bmatrix} 1 \\ 0 \end{Bmatrix}: \ \boldsymbol{Y}_2^{(M)} = \begin{Bmatrix} 0 \\ 1 \end{Bmatrix}$$

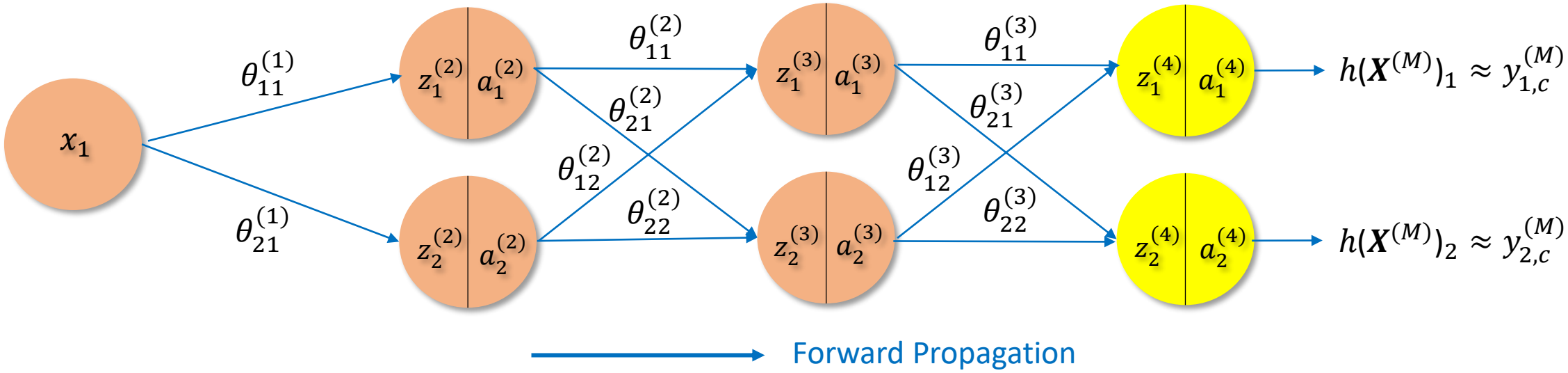


Fig. 6.8 A forward propagation example

Backpropagation

Given: $z_j^{(l)}$ & $a_i^{(l-1)}$ and $y_{j,c}^{(M)}$ Find Unknown: $\delta_j^{(l)}$ & $\frac{\partial J(\theta)}{\partial \theta_{ji}^{(l)}}$ and update $\theta_{ji}^{(l)}$

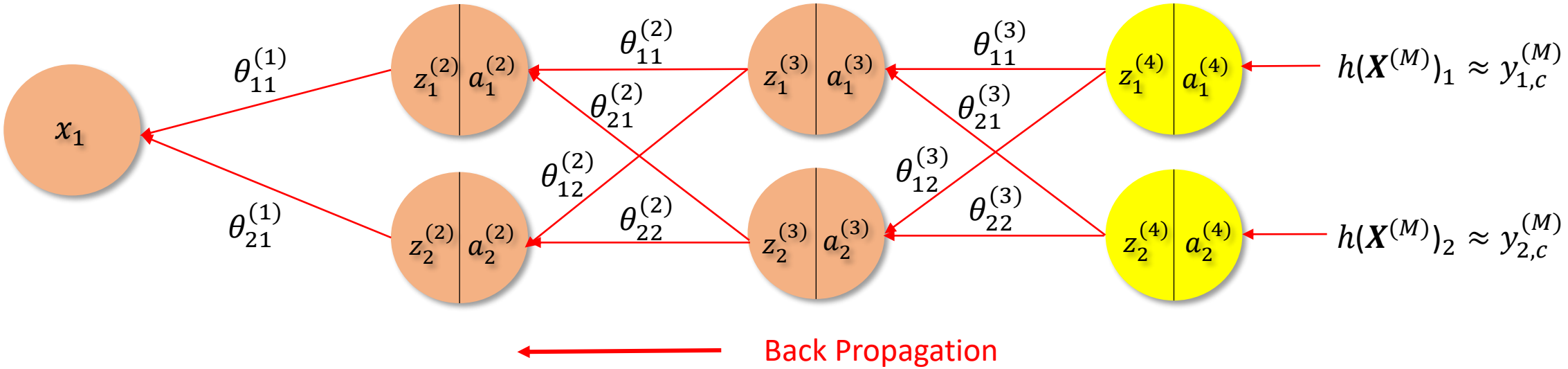


Fig. 6.9.1 A back propagation example

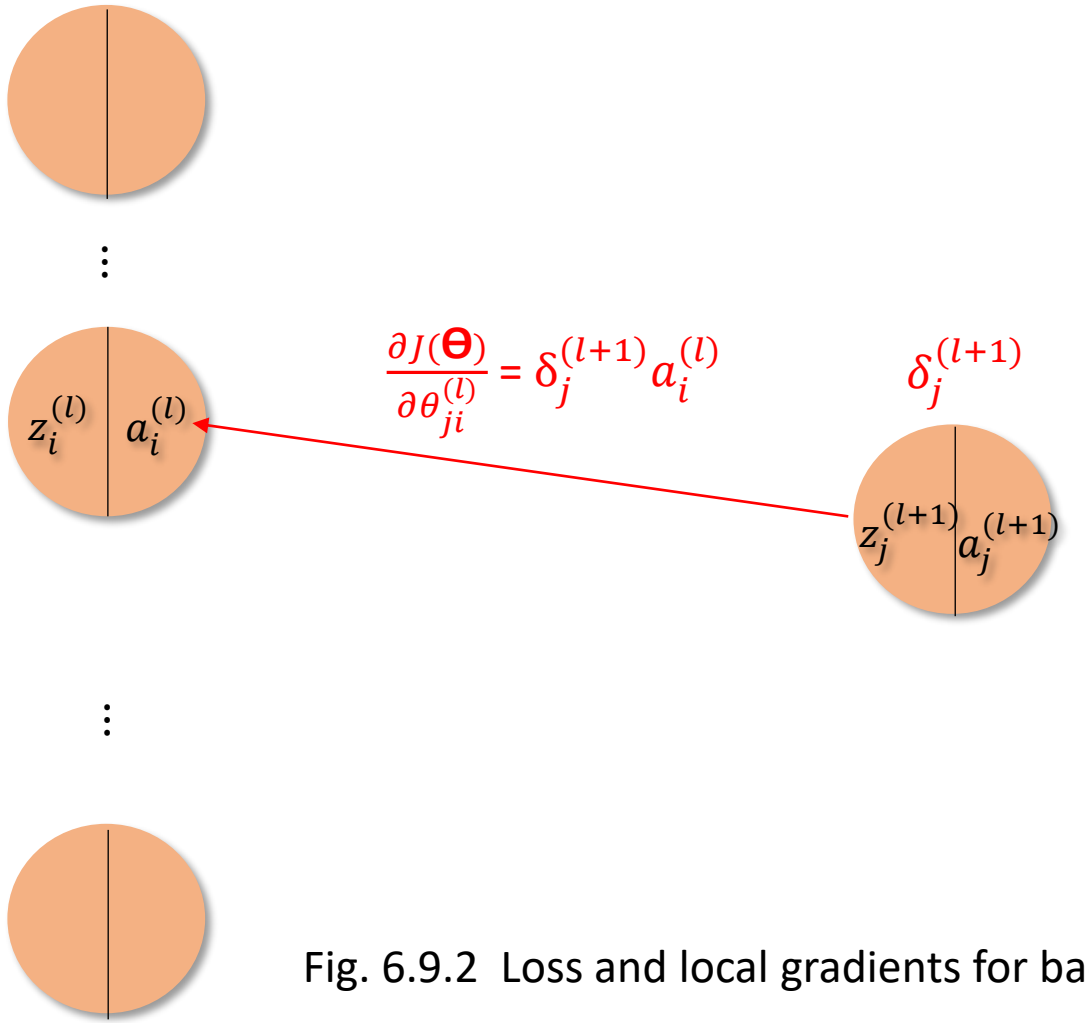


Fig. 6.9.2 Loss and local gradients for backpropagation

- Back Propagation computes the gradient of a given cost function w.r.t weights which are supposed to be turned based on the errors at output. It makes the errors smaller during the repetition of back and forward propagation.

- Compute the Gradient Descent using back propagation algorithm for Fig. 6.9 when the network has 2 classes:

  Gradient Descent: $\theta_{ji}^{(l)} := \theta_{ji}^{(l)} - \alpha \frac{\partial J(\boldsymbol{\Theta})}{\partial \theta_{ji}^{(l)}}$

Take the simplest $J(\boldsymbol{\Theta})$

$$J(\boldsymbol{\Theta}) = \frac{1}{2}\sum_{j=1}^{2}(a_j^{(4)} - y_{j,c}^{(M)})^2 = \frac{1}{2}\sum_{j=1}^{2}(z_j^{(4)} - y_{j,c}^{(M)})^2$$

'M' and 'c' are given in the dataset table. Assume $a_j^{(4)} = z_j^{(4)}$ at output which means that a sigmoid function is not applied to the last output units. For updating $\theta_{ji}^{(3)}$

$$\frac{\partial J(\boldsymbol{\Theta})}{\partial \theta_{ji}^{(3)}} = \frac{\partial J(\boldsymbol{\Theta})}{\partial z_j^{(4)}} \frac{\partial z_j^{(4)}}{\partial \theta_{ji}^{(3)}} = \delta_j^{(4)} a_i^{(3)}$$

here,

$$\frac{\partial J(\boldsymbol{\Theta})}{\partial z_j^{(4)}} = (a_j^{(4)} - y_{j,c}^{(M)}) = (z_j^{(4)} - y_{j,c}^{(M)}) = \delta_j^{(4)},\ z_j^{(4)} = \sum_{i=1}^{S(3)} \theta_{ji}^{(3)} a_i^{(3)} \text{ and } \frac{\partial z_j^{(4)}}{\partial \theta_{ji}^{(3)}} = \frac{\partial}{\partial \theta_{ji}^{(3)}} (\theta_{ji}^{(3)} a_i^{(3)}) = a_i^{(3)}$$

For updating $\theta_{ji}^{(2)}$

$$\frac{\partial J(\boldsymbol{\Theta})}{\partial \theta_{ji}^{(2)}} = \frac{\partial J(\boldsymbol{\Theta})}{\partial z_j^{(3)}} \frac{\partial z_j^{(3)}}{\partial \theta_{ji}^{(2)}} = \frac{\partial J(\boldsymbol{\Theta})}{\partial z_i^{(4)}} \frac{\partial z_i^{(4)}}{\partial z_j^{(3)}} \frac{\partial z_j^{(3)}}{\partial \theta_{ji}^{(2)}} = \delta_j^{(3)} a_i^{(2)}$$

here,

$$z_i^{(4)} = \sum_{i=1}^{S(l=4)} (\theta_{ji}^{(3)})^T a_j^{(3)} = \sum_{i=1}^{S(4)} (\theta_{ji}^{(3)})^T \sigma\left(z_j^{(3)}\right) \text{ when } a_j^{(3)} = \sigma\left(z_j^{(3)}\right)$$

$$\frac{\partial z_i^{(4)}}{\partial z_j^{(3)}} = \sum_{i=1}^{S(4)} (\theta_{ji}^{(3)})^T \sigma\left(z_j^{(3)}\right)\left(1 - \sigma\left(z_j^{(3)}\right)\right) = \sum_{i=1}^{S(4)} (\theta_{ji}^{(3)})^T \sigma'(z_j^{(3)})$$

$$\frac{\partial J(\boldsymbol{\Theta})}{\partial z_j^{(3)}} = \frac{\partial J(\boldsymbol{\Theta})}{\partial z_i^{(4)}} \frac{\partial z_i^{(4)}}{\partial z_j^{(3)}} = \delta_i^{(4)} \sum_{i=1}^{S(4)} (\theta_{ji}^{(3)})^T \sigma'(z_j^{(3)}) = \sigma'(z_j^{(3)}) \sum_{i=1}^{S(4)} (\theta_{ji}^{(3)})^T \delta_i^{(4)} = \delta_j^{(3)}$$

The subscripts *j* in $(\theta_{ji}^{(3)})^T$ and *i* in $\delta_i^{(4)}$ must be identical.

$$z_j^{(3)} = \sum_{i=1}^{S(2)} \theta_{ji}^{(2)} a_i^{(2)} \text{ and } \frac{\partial z_j^{(3)}}{\partial \theta_{ji}^{(2)}} = \frac{\partial}{\partial \theta_{ji}^{(2)}} (\theta_{ji}^{(2)} a_i^{(2)}) = a_i^{(2)}$$

For updating $\theta_{ji}^{(1)}$

$$\frac{\partial J(\boldsymbol{\Theta})}{\partial \theta_{ji}^{(1)}} = \frac{\partial J(\boldsymbol{\Theta})}{\partial z_j^{(2)}} \frac{\partial z_j^{(2)}}{\partial \theta_{ji}^{(1)}} = \frac{\partial J(\boldsymbol{\Theta})}{\partial z_i^{(3)}} \frac{\partial z_i^{(3)}}{\partial z_j^{(2)}} \frac{\partial z_j^{(2)}}{\partial \theta_{ji}^{(1)}} = \delta_j^{(2)} a_i^{(1)}$$

here,

$$z_i^{(3)} = \sum_{i=1}^{S(l=3)} (\theta_{ji}^{(2)})^T a_j^{(2)} = \sum_{i=1}^{S(3)} (\theta_{ji}^{(2)})^T \sigma\left(z_j^{(2)}\right) \text{ when } a_j^{(2)} = \sigma\left(z_j^{(2)}\right)$$

$$\frac{\partial z_i^{(3)}}{\partial z_j^{(2)}} = \sum_{i=1}^{S(3)} (\theta_{ji}^{(2)})^T \sigma\left(z_j^{(2)}\right)\left(1 - \sigma\left(z_j^{(2)}\right)\right) = \sum_{i=1}^{S(3)} (\theta_{ji}^{(2)})^T \sigma'(z_j^{(2)})$$

$$\frac{\partial J(\boldsymbol{\Theta})}{\partial z_j^{(2)}} = \frac{\partial J(\boldsymbol{\Theta})}{\partial z_i^{(3)}} \frac{\partial z_i^{(3)}}{\partial z_j^{(2)}} = \delta_i^{(3)} \sum_{i=1}^{(3)} (\theta_{ji}^{(2)})^T \sigma'(z_j^{(2)}) = \sigma'(z_j^{(2)}) \sum_{i=1}^{S(3)} (\theta_{ji}^{(2)})^T \delta_i^{(3)} = \delta_j^{(2)}$$

The subscripts *j* in $(\theta_{ji}^{(2)})^T$ and *i* in $\delta_i^{(3)}$ must be identical.

$$z_j^{(2)} = \sum_{i=1}^{S(1)} \theta_{ji}^{(1)} a_i^{(1)} \text{ and } \frac{\partial z_j^{(2)}}{\partial \theta_{ji}^{(1)}} = \frac{\partial}{\partial \theta_{ji}^{(1)}} (\theta_{ji}^{(1)} a_i^{(1)}) = a_i^{(1)}$$

Generalization of the all formulas gives

$$\delta_j^{(4)} = (a_j^{(4)} - y_{j,c}^{(M)}) = (z_j^{(4)} - y_{j,c}^{(M)})$$

$$\delta_j^{(3)} = \sigma'(z_j^{(3)}) \sum_{i=1}^{S(4)} (\theta_{ji}^{(3)})^T \delta_i^{(4)}$$

$$\delta_j^{(2)} = \sigma'(z_j^{(2)}) \sum_{i=1}^{S(3)} (\theta_{ji}^{(2)})^T \delta_i^{(3)} \; (\text{N}\sigma\, \delta_j^{(1)})$$

Loss gradient: it is a loss gradient (node error).

Local gradient to update the weight, $\theta_{ji}^{(l)}$

therefore,

$\delta_j^{(L)} = (z_j^{(L)} - y_{j,c}^{(M)})$ for the output layer

$\delta_j^{(l)} = \frac{\partial J(\boldsymbol{\Theta})}{\partial z_j^{(l)}} = \sigma'(z_j^{(l)}) \sum_{i=1}^{S(l+1)} (\theta_{ji}^{(l)})^T \delta_i^{(l+1)}$ for the hidden layer: $l \in \{2, 3, 4, \dots, L-1\}$

$\frac{\partial J(\boldsymbol{\Theta})}{\partial \theta_{ji}^{(l)}} = \delta_j^{(l+1)} a_i^{(l)}$ → Update

$$\theta_{ji}^{(l)} := \theta_{ji}^{(l)} - \alpha \frac{\partial J(\boldsymbol{\Theta})}{\partial \theta_{ji}^{(l)}}$$

A whole process regarding tuning $\theta_{ji}^{(l)}$ is called ‘training’ of a neural network or ‘learning’

What is a training process of a neural network?

(1) Symmetric breaking random initialization of $\theta_{ji}^{(l)}$.

(2) Implementation of forward propagation to find $\boldsymbol{Z}^{(l)}$ & $\boldsymbol{A}^{(l)}$ and $h(\boldsymbol{X}^{(M)})$ based on given $\mathbf{X}$ and initialized $\boldsymbol{\Theta}^{(l-1)}$.

(3) Implementation of the code to compute $J(\boldsymbol{\Theta})$ for each example or batch.

(4) Implementation of back propagation to compute $\frac{\partial J(\boldsymbol{\Theta})}{\partial \theta_{ji}^{(l)}}$ and $\delta_j^{(l+1)}$, and update $\theta_{ji}^{(l)}$ based on $z_j^{(l)}$ & $a_i^{(l-1)}$ and $y_{j,c}^{(M)}$.

(5) Minimization of $J(\boldsymbol{\Theta})$ to find optimized $\theta_{ji}^{(l)}$ during backpropagations.

Example:

Implement the 1st iteration and the 2nd forward propagation and then compare the cost function values at output. Applied a sigmoid activation function for a hidden layer: $a_j^{(2)} = \sigma\left(z_j^{(2)}\right)$, and assume $a_1^{(3)} = z_1^{(3)}$. Actual output is $y_{1,1}^{(1)}$=1.0 for one example (M = 1) and one class(c = 1).

Cost function: $J(\boldsymbol{\Theta}) = \frac{1}{2}(a_1^{(3)} - y_{1,1}^{(1)})^2$. here, α = 0.5.

$\mathbf{X} = \begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix} = \begin{Bmatrix} 2.0 \\ 3.0 \end{Bmatrix}$, initial weights are

$\boldsymbol{\Theta}^{(1)} = \begin{bmatrix} \theta_{11}^{(1)} & \theta_{12}^{(1)} \\ \theta_{21}^{(1)} & \theta_{22}^{(1)} \end{bmatrix} = \begin{bmatrix} 0.11 & 0.12 \\ 0.21 & 0.08 \end{bmatrix}$

and $\boldsymbol{\Theta}^{(2)} = \{\theta_{11}^{(2)}, \theta_{12}^{(2)}\} = \{0.14, 0.15\}$

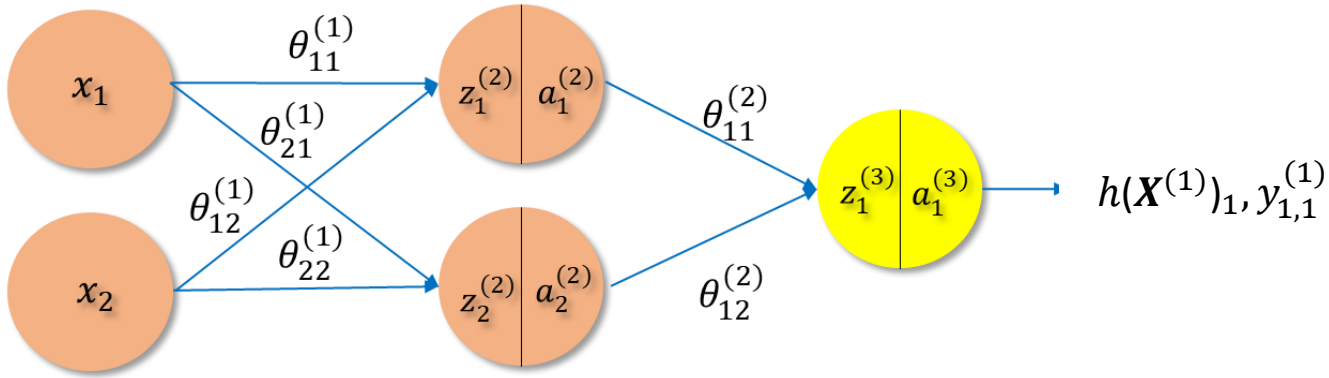


Fig. 6.10 A neural network model

Step 1: implement the 1st forward propagation with inputs and initial weights.

$\boldsymbol{A}^{(1)} = \mathbf{X} = \begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix} = \begin{Bmatrix} 2.0 \\ 3.0 \end{Bmatrix}$

$\boldsymbol{Z}^{(2)} = \boldsymbol{\Theta}^{(1)}\, \boldsymbol{A}^{(1)} = \begin{bmatrix} 0.11 & 0.12 \\ 0.21 & 0.08 \end{bmatrix} \begin{Bmatrix} 2.0 \\ 3.0 \end{Bmatrix} = \begin{Bmatrix} 0.58 \\ 0.66 \end{Bmatrix}$

$\boldsymbol{A}^{(2)} = \sigma(\boldsymbol{Z}^{(2)}) = \begin{Bmatrix} 0.641067 \\ 0.659260 \end{Bmatrix}$

$\boldsymbol{Z}^{(3)} = \boldsymbol{\Theta}^{(2)}\, \boldsymbol{A}^{(2)} = \{0.14, 0.15\} \begin{Bmatrix} 0.641067 \\ 0.659260 \end{Bmatrix} = 0.188638$

$\boldsymbol{A}^{(3)} = \boldsymbol{Z}^{(3)} = 0.188638$

Error checking using: $J(\boldsymbol{\Theta}) = \frac{1}{2}(a_1^{(3)} - y_{1,1}^{(1)})^2 = \frac{1}{2}(0.188638 - 1.0)^2 = 0.329154$

Step 2: implement the 1st back propagation in tensor notation.

$\delta_1^{(3)} = (z_1^{(3)} - y_{1,1}^{(1)}) = (0.188638 - 1.0) = -0.811362$

$\delta_1^{(2)} = \sigma'(z_1^{(2)}) \sum_{i=1}^{S(3)=1} (\theta_{1i}^{(2)})^T \delta_i^{(3)} = \sigma'(z_1^{(2)})(\theta_{11}^{(2)})^T \delta_1^{(3)} = \sigma'(0.58)\,(0.14)(-0.811362) = -0.0261372$

$\delta_2^{(2)} = \sigma'(z_2^{(2)}) \sum_{i=1}^{1} (\theta_{2i}^{(2)})^T \delta_i^{(3)} = \sigma'(z_2^{(2)})(\theta_{21}^{(2)})^T \delta_1^{(3)} = \sigma'(0.66)\,(0.15)(-0.811362) = -0.0273392$

Error checking using:

$\frac{\partial J(\boldsymbol{\Theta})}{\partial \theta_{11}^{(2)}} = \delta_1^{(3)} a_1^{(2)} = -0.811362 \times 0.641067 = -0.520137$

$\frac{\partial J(\boldsymbol{\Theta})}{\partial \theta_{12}^{(2)}} = \delta_1^{(3)} a_2^{(2)} = -0.811362 \times 0.659260 = -0.534899$

$\frac{\partial J(\boldsymbol{\Theta})}{\partial \theta_{11}^{(1)}} = \delta_1^{(2)} \, a_1^{(1)} = -0.0261372 \times 2.0 = -0.0522744$

$\frac{\partial J(\boldsymbol{\Theta})}{\partial \theta_{12}^{(1)}} = \delta_1^{(2)} \, a_2^{(1)} = -0.0261372 \times 3.0 = -0.0784116$

$\frac{\partial J(\boldsymbol{\Theta})}{\partial \theta_{21}^{(1)}} = \delta_2^{(2)} \, a_1^{(1)} = -0.0273392 \times 2.0 = -0.0546784$

$\frac{\partial J(\boldsymbol{\Theta})}{\partial \theta_{22}^{(1)}} = \delta_2^{(2)} \, a_2^{(1)} = -0.0273392 \times 3.0 = -0.0820176$

Update $\theta_{ji}^{(l)}$ using GD

$\theta_{11}^{(2)} := \theta_{11}^{(2)} - 0.5 \, \frac{\partial J(\boldsymbol{\Theta})}{\partial \theta_{11}^{(2)}} = 0.14 - 0.5 \times -0.520137 = 0.400069$

$\theta_{12}^{(2)} := \theta_{12}^{(2)} - 0.5 \, \frac{\partial J(\boldsymbol{\Theta})}{\partial \theta_{12}^{(2)}} = 0.15 - 0.5 \times -0.534899 = 0.417450$

$\theta_{11}^{(1)} := \theta_{11}^{(1)} - 0.5 \frac{\partial J(\boldsymbol{\Theta})}{\partial \theta_{11}^{(1)}} = 0.11 - 0.5 \times -0.0522744 = 0.136137$

$\theta_{12}^{(1)} := \theta_{12}^{(1)} - 0.5 \frac{\partial J(\boldsymbol{\Theta})}{\partial \theta_{12}^{(1)}} = 0.12 - 0.5 \times -0.0784116 = 0.159206$

$\theta_{21}^{(1)} := \theta_{21}^{(1)} - 0.5 \frac{\partial J(\boldsymbol{\Theta})}{\partial \theta_{21}^{(1)}} = 0.21 - 0.5 \times -0.0546784 = 0.237339$

$\theta_{22}^{(1)} := \theta_{22}^{(1)} - 0.5 \frac{\partial J(\boldsymbol{\Theta})}{\partial \theta_{22}^{(1)}} = 0.08 - 0.5 \times -0.0820176 = 0.121009$

Therefore, new $\theta_{ji}^{(l)}$ are

$\boldsymbol{\Theta}_{new}^{(1)} = \begin{bmatrix} 0.136137 & 0.159206 \\ 0.237339 & 0.121009 \end{bmatrix}$ and $\boldsymbol{\Theta}_{new}^{(2)} = \{0.400069, 0.417450\}$

Step 3: implement the 2nd forward propagation with inputs and new weights.

$$\boldsymbol{A}^{(1)} = \mathbf{X} = \begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix} = \begin{Bmatrix} 2.0 \\ 3.0 \end{Bmatrix}$$

$$\boldsymbol{Z}^{(2)}_{new} = \boldsymbol{\Theta}^{(1)}_{new}\, \boldsymbol{A}^{(1)} = \begin{bmatrix} 0.136137 & 0.159206 \\ 0.237339 & 0.121009 \end{bmatrix} \begin{Bmatrix} 2.0 \\ 3.0 \end{Bmatrix} = \begin{Bmatrix} 0.749892 \\ 0.837705 \end{Bmatrix}$$

$$\boldsymbol{A}^{(2)}_{new} = \sigma(\boldsymbol{Z}^{(2)}_{new}) = \begin{Bmatrix} 0.679155 \\ 0.697982 \end{Bmatrix}$$

$$\boldsymbol{Z}^{(3)}_{new} = \boldsymbol{\Theta}^{(2)}_{new}\, \boldsymbol{A}^{(2)}_{new} = \{0.400069,\ 0.417450\} \begin{Bmatrix} 0.679155 \\ 0.697982 \end{Bmatrix} = 0.563081$$

$$\boldsymbol{A}^{(3)}_{new} = \boldsymbol{Z}^{(3)}_{new} = 0.563081$$

Error checking using: $\boldsymbol{J}_{new}(\boldsymbol{\Theta}) = \frac{1}{2}(a^{(3)}_{1,new} - y^{(1)})^2 = \frac{1}{2}(0.563081 - 1.0)^2 = 0.0954491$

$J(\boldsymbol{\Theta}) = 0.329154 > \boldsymbol{J}_{new}(\boldsymbol{\Theta}) = 0.0954491$

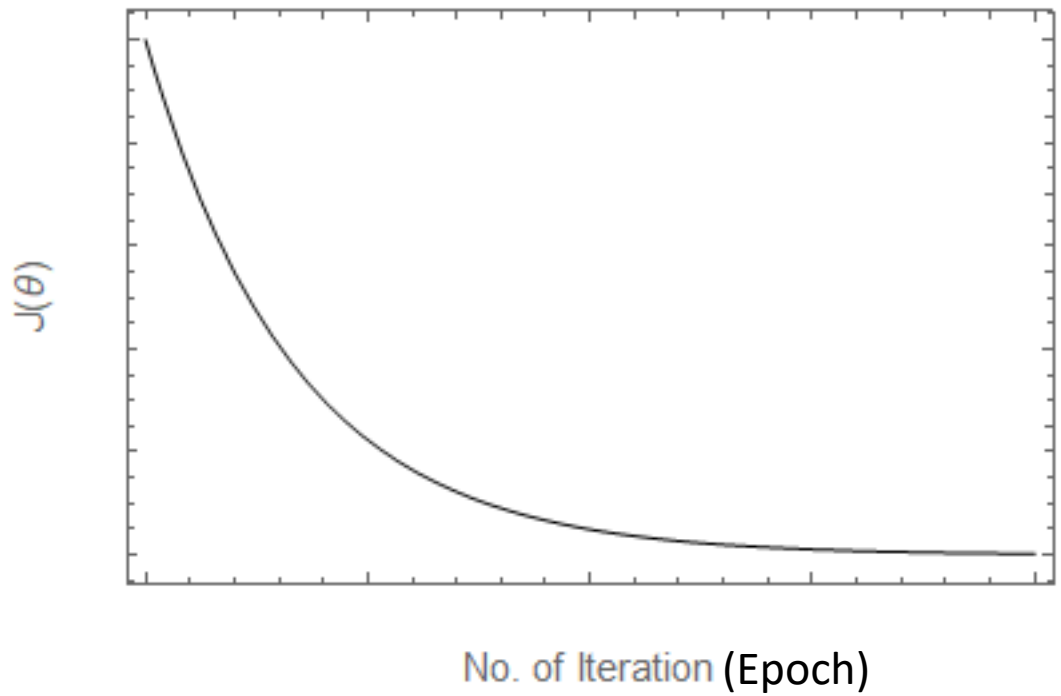


Fig. 6.11 An error at output decreases over epochs

6.6 Initialization

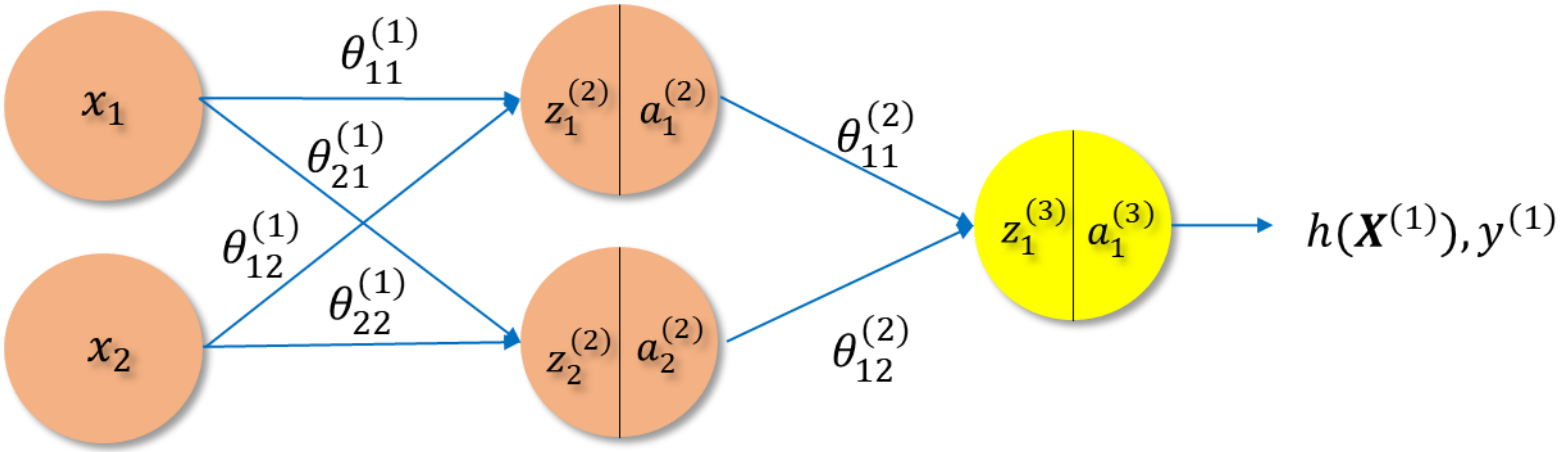


Fig. 6.12 A neural network model

- ❑ The initial values of **Θ** are needed for the 1st forward propagation.
- ❑ Bad initialization gives terrible results in the optimization of **Θ** (learning process): vanishing or exploding gradient problems

Bad Initialization Cases

(1) Zero or symmetric Initialization

for all $\theta_{ji}^{(l)}$ = 0, all the activation function produce same values, $a_1^{(2)} = a_2^{(2)}$ for forward propagations:

$z_1^{(2)} = \theta_{11}^{(1)} a_1^{(1)} + \theta_{12}^{(1)} a_2^{(1)} = 0$

$a_1^{(2)} = \sigma(z_1^{(2)}) = 0.5$

$z_2^{(2)} = \theta_{21}^{(1)} a_1^{(1)} + \theta_{22}^{(1)} a_2^{(1)} = 0$

$a_2^{(2)} = \sigma(z_2^{(2)}) = 0.5$

for all $\theta_{ji}^{(l)}$ = 0, no updating of $\theta_{ji}^{(l)}$ will be made, $\delta_1^{(2)} = \delta_2^{(2)} = 0$ for back propagations:

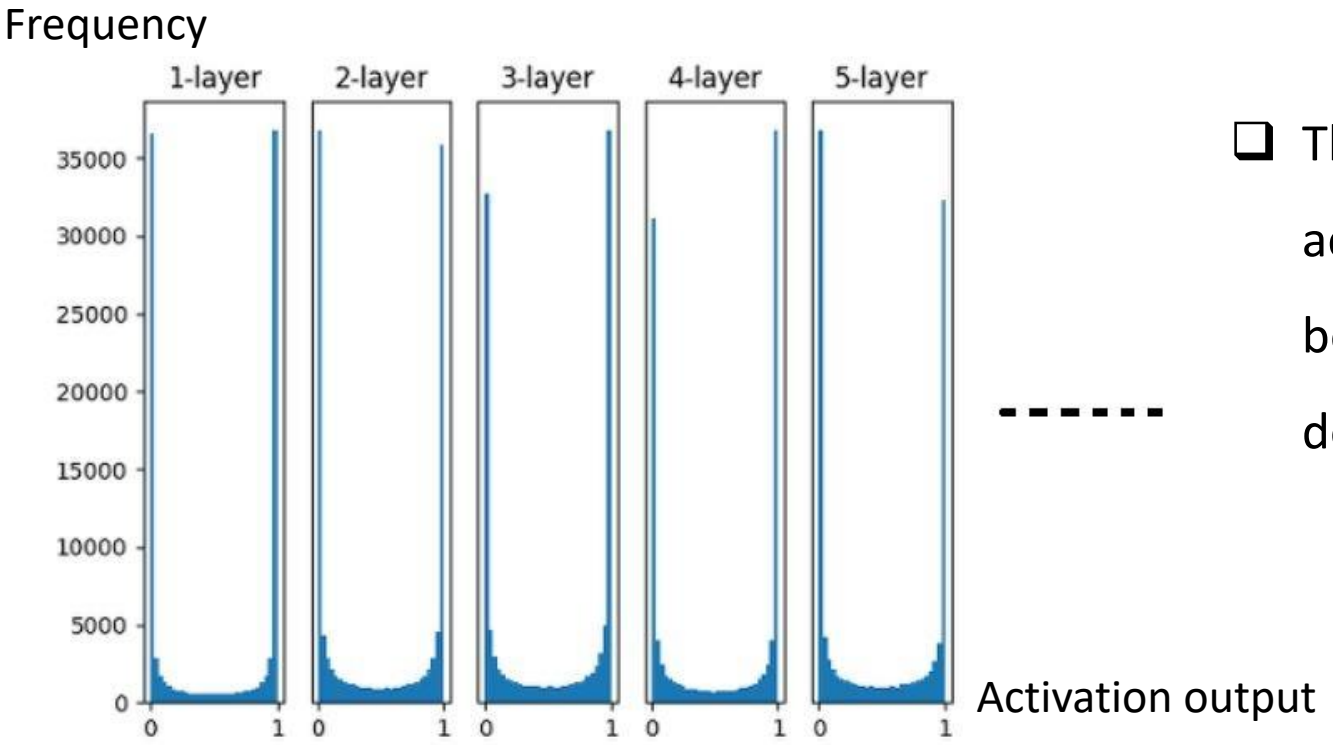
$\delta_1^{(2)} = \sigma'(z_1^{(2)}) (\theta_{11}^{(2)})^T \delta_1^{(3)} = \sigma'(z_1^{(2)}) \times 0 \times \delta_1^{(3)} = 0$

$\delta_2^{(2)} = \sigma'(z_2^{(2)}) (\theta_{21}^{(2)})^T \delta_1^{(3)} = \sigma'(z_2^{(2)}) \times 0 \times \delta_1^{(3)} = 0$

When some machine learning models have weights all initialized to the same value (symmetric initialization), it can be difficult or impossible for the weights to differ as the model is trained and the symmetric breaking is required.

Random initialization is required to prevent dying ReLU from backpropagation when ReLU inputs are negative (1st derivative of ReLU must be '0' when its inputs are negative): Initialize all $\theta_{ji}^{(l)}$ to have asymmetric values in $-\varepsilon \leq \theta_{ji}^{(l)} \leq \varepsilon$ or $0 < \theta_{ji}^{(l)} \leq 1.0$.

(2) Initialization following a normal distribution with a mean of 0 and a standard deviation of 1



- This initialization makes most of sigmoid activation outputs fall around ‘0’ and ‘1’ because of a too large standard deviation.

Fig. 6.13 Sigmoid activation outputs after normal distribution initialization

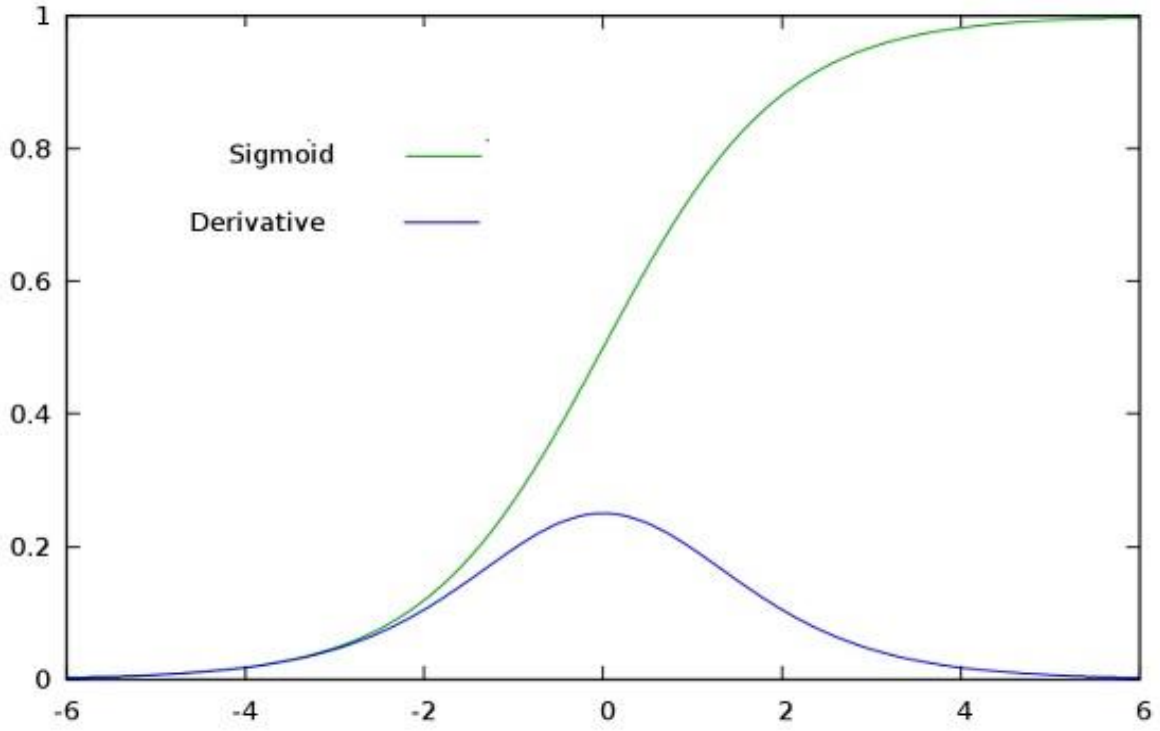
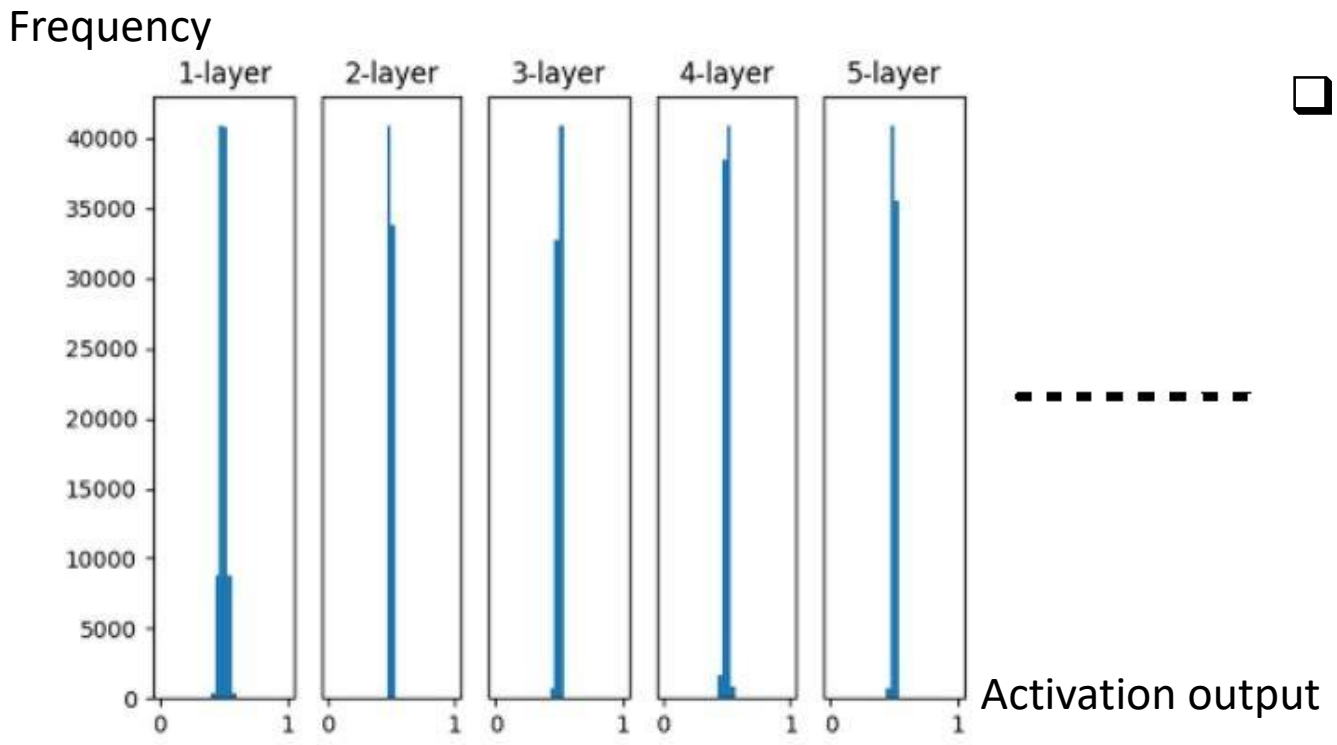


Fig. 6.14 A sigmoid function and its1st derivative

- The activation outputs around ‘0’ and ‘1’ make the 1st derivative values of the activation have almost zero, in other words, a vanishing problem arises: gradient will be vanishingly small.
- We should change both the initial values of **ϴ** and the activation function to solve the vanishing problem caused by the wrong initialization.

(3) Initialization following a normal distribution with a mean of 0 and a small standard deviation value (0.01)



- For this initialization, most of sigmoid activation outputs are squeezed into around '0.5' because of a too small standard deviation. It is the better case than the 2nd case, but it still causes problem.

Fig. 6.15 Sigmoid activation outputs after normal distribution initialization

Frequency

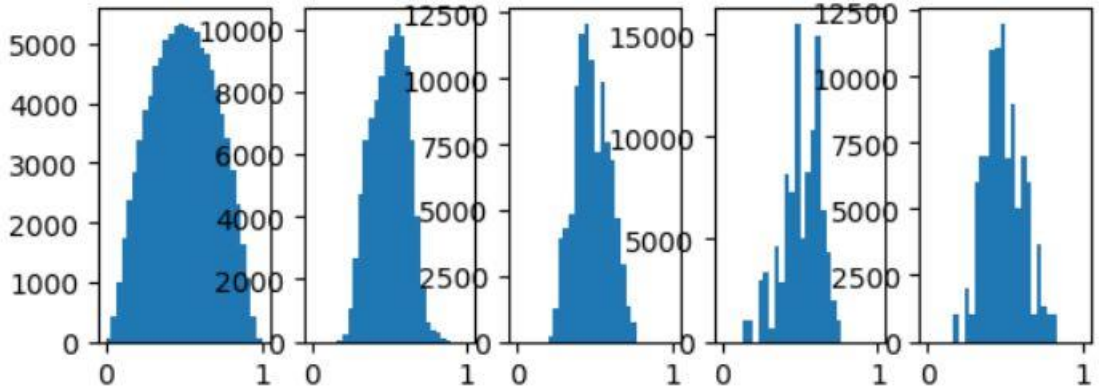


Fig. 6.16.1 Sigmoid activation outputs after Xavier initialization

- Tanh (hyperbolic tangent) activation performs better than Sigmoid activation when using Xavier initialization .

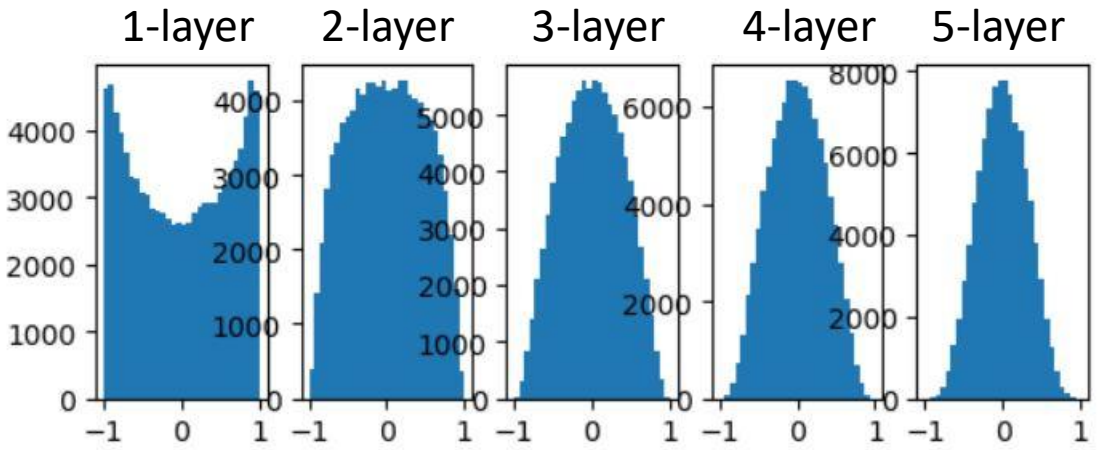


Fig. 6.16.2 Tanh activation outputs after Xavier initialization

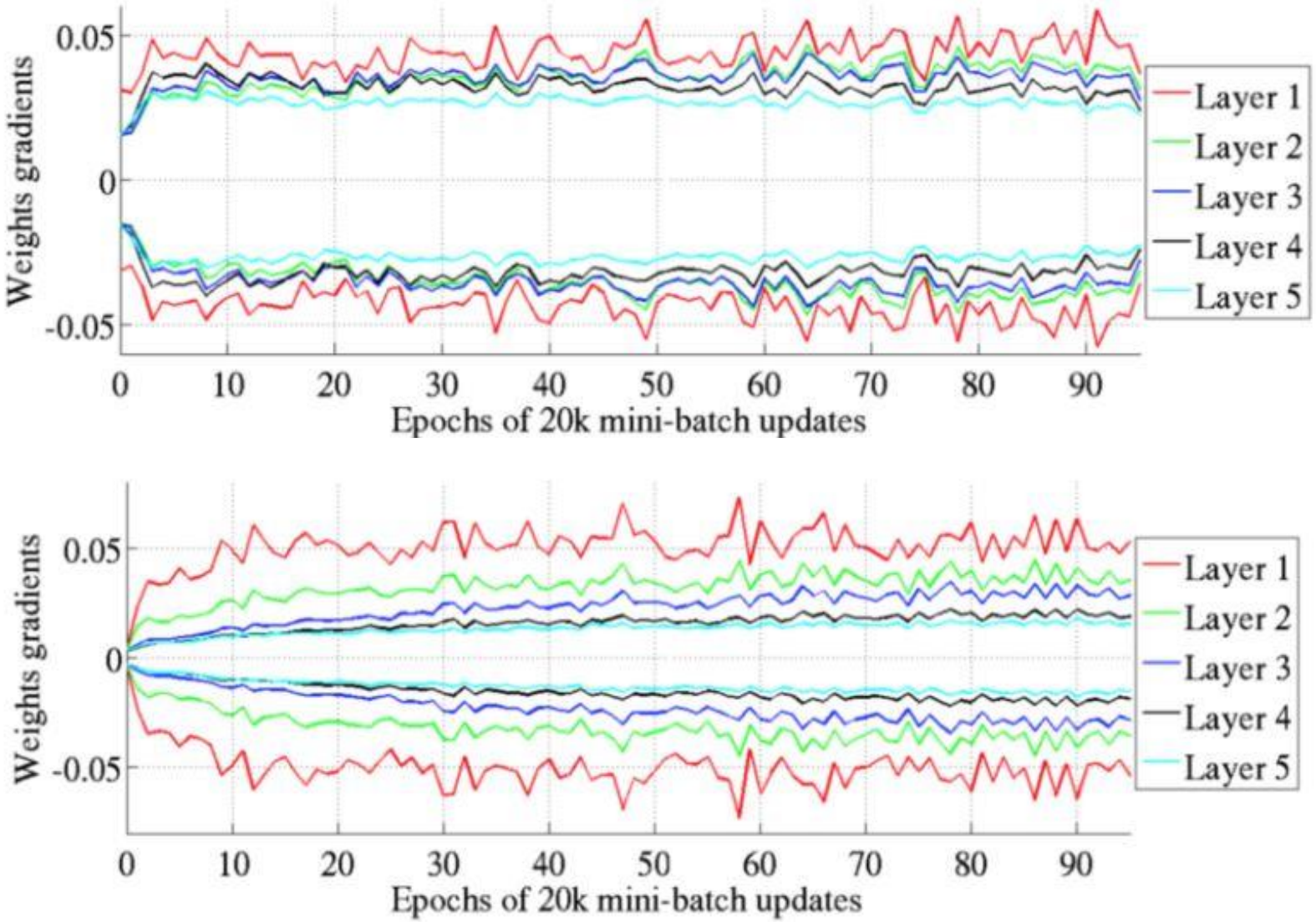


Fig. 6.17 With Xavier nor. init. (upper) and with standard normal init. (bottom)

- Glorot and Bengio shows that for a tanh network, Xavier initialization would maintain the back-propagated gradients all the way up or down. Xavier initialization enables a 5-layer network to maintain near identical variances of its weight gradients across layers.
- While the standard normal initialization shows that the gradients at its top-most layers are approaching zero: gradient vanishing.

Xavier Initialization

(1) Xavier normal initialization:

$N(0, Var(\boldsymbol{\Theta}^{(l)}) = \sigma^2)$ for $\sigma = \sqrt{\frac{2}{n_{in}+n_{out}}}$

(2) Xavier uniform initialization

$U(-\sqrt{\frac{6}{n_{in}+n_{out}}}, \sqrt{\frac{6}{n_{in}+n_{out}}})$

He Initialization

(1) Kaiming He normal initialization:

$N(0, Var(\boldsymbol{\Theta}^{(l)}) = \sigma^2)$ for $\sigma = \sqrt{\frac{2}{n_{in}}}$

(2) Kaiming He uniform initialization

$U(-\sqrt{\frac{6}{n_{in}}}, \sqrt{\frac{6}{n_{in}}})$

where n_{in} is the number of incoming network connections, or "fan-in," to the layer, and n_{out} is the number of outgoing network connections from that layer, also known as the "fan-out."

Good combinations of the type of activation functions and initialization

Activation function	Initialization	
	Normal distribution	Uniform distribution (-r, r)
Hyperbolic Tangent	$\sigma = \sqrt{\frac{2}{n_{in}+n_{out}}}$ (Xavier)	$r = \sqrt{\frac{6}{n_{in}+n_{out}}}$ (Xavier)
Sigmoid	$\sigma = 4\sqrt{\frac{2}{n_{in}+n_{out}}}$	$r = 4\sqrt{\frac{6}{n_{in}+n_{out}}}$
ReLU	$\sigma = \sqrt{\frac{12}{n_{in}+n_{out}}}$	$r = \sqrt{\frac{12}{n_{in}+n_{out}}}$

Kaiming He initialization + ReLU is one of another good cases.

Bias also must be initialized. In many cases, zero initialization is common but sometimes ReLU requires the initialization with small value like 0.01.