Neural Networks
Part 2

Yingyu Liang

yliang@cs.wisc.edu

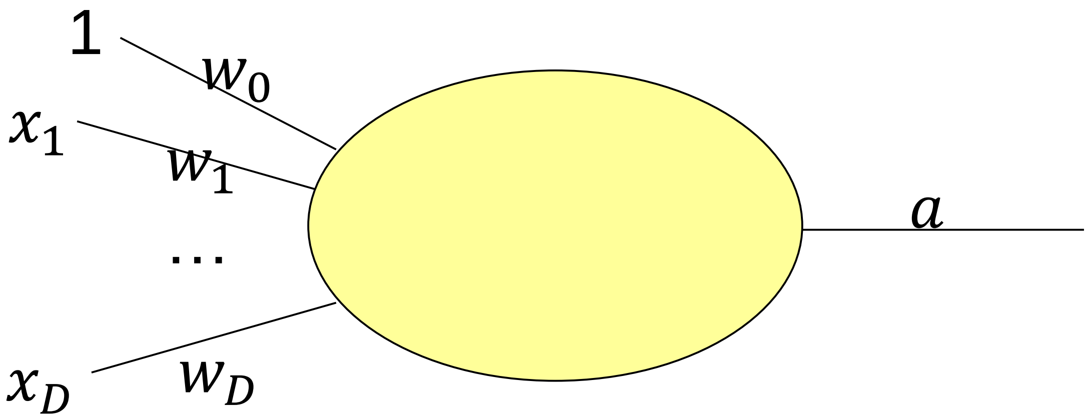
Computer Sciences Department

University of Wisconsin, Madison

[Based on slides from Jerry Zhu, Mohit Gupta]

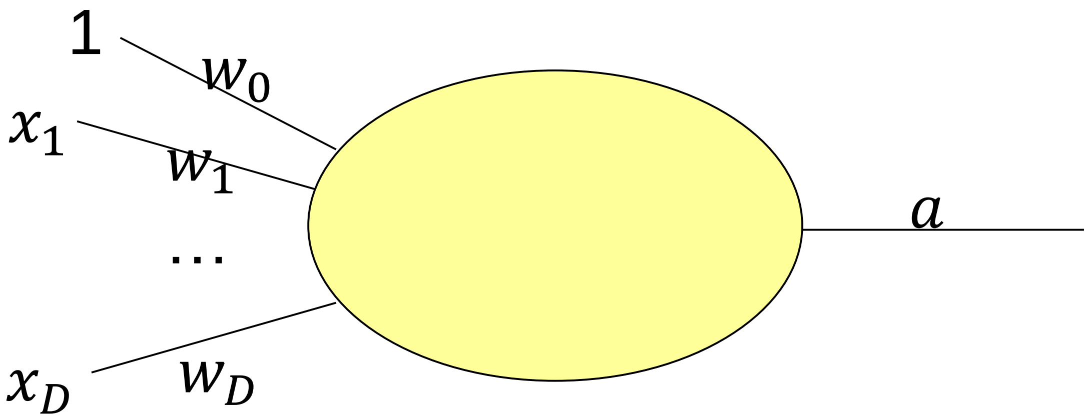
Limited power of one single neuron

- Perceptron: $a = g(\sum_d w_d x_d)$
- Activation function g: linear, step, sigmoid

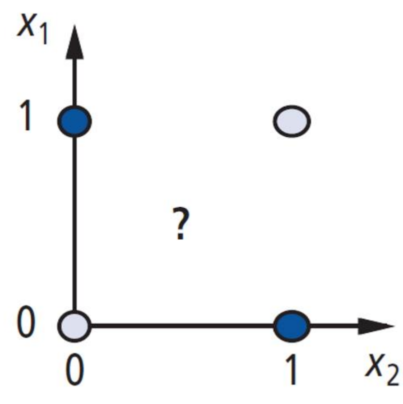


Limited power of one single neuron

- Perceptron: $a = g(\sum_d w_d x_d)$
- Activation function g: linear, step, sigmoid

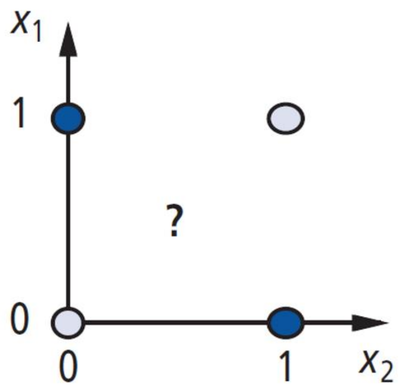


- Decision boundary linear even for nonlinear g
- XOR problem



Limited power of one single neuron

- XOR problem

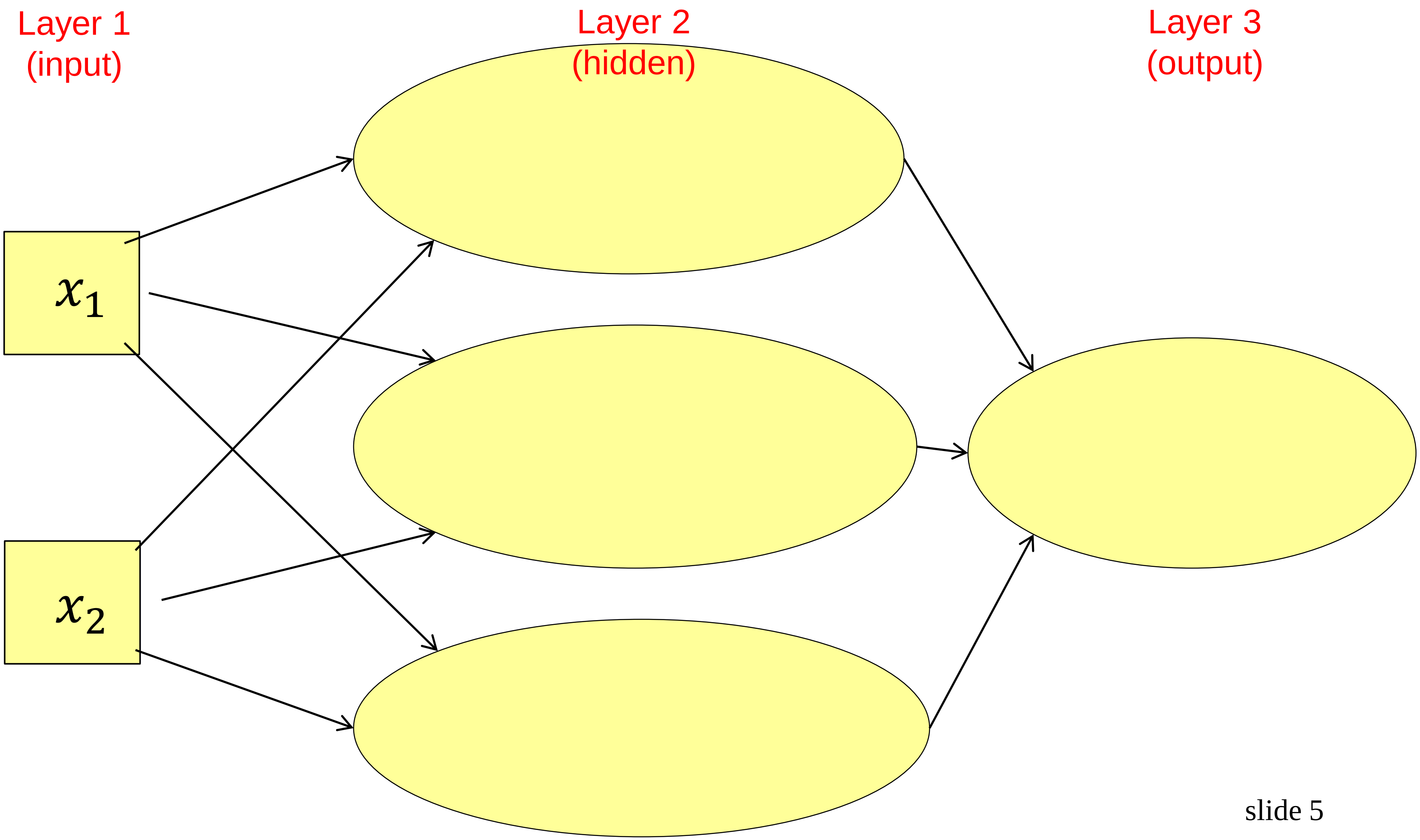


- Wait! If one can represent AND, OR, NOT, one can represent any logic circuit (including XOR), by connecting them

Question: how to?

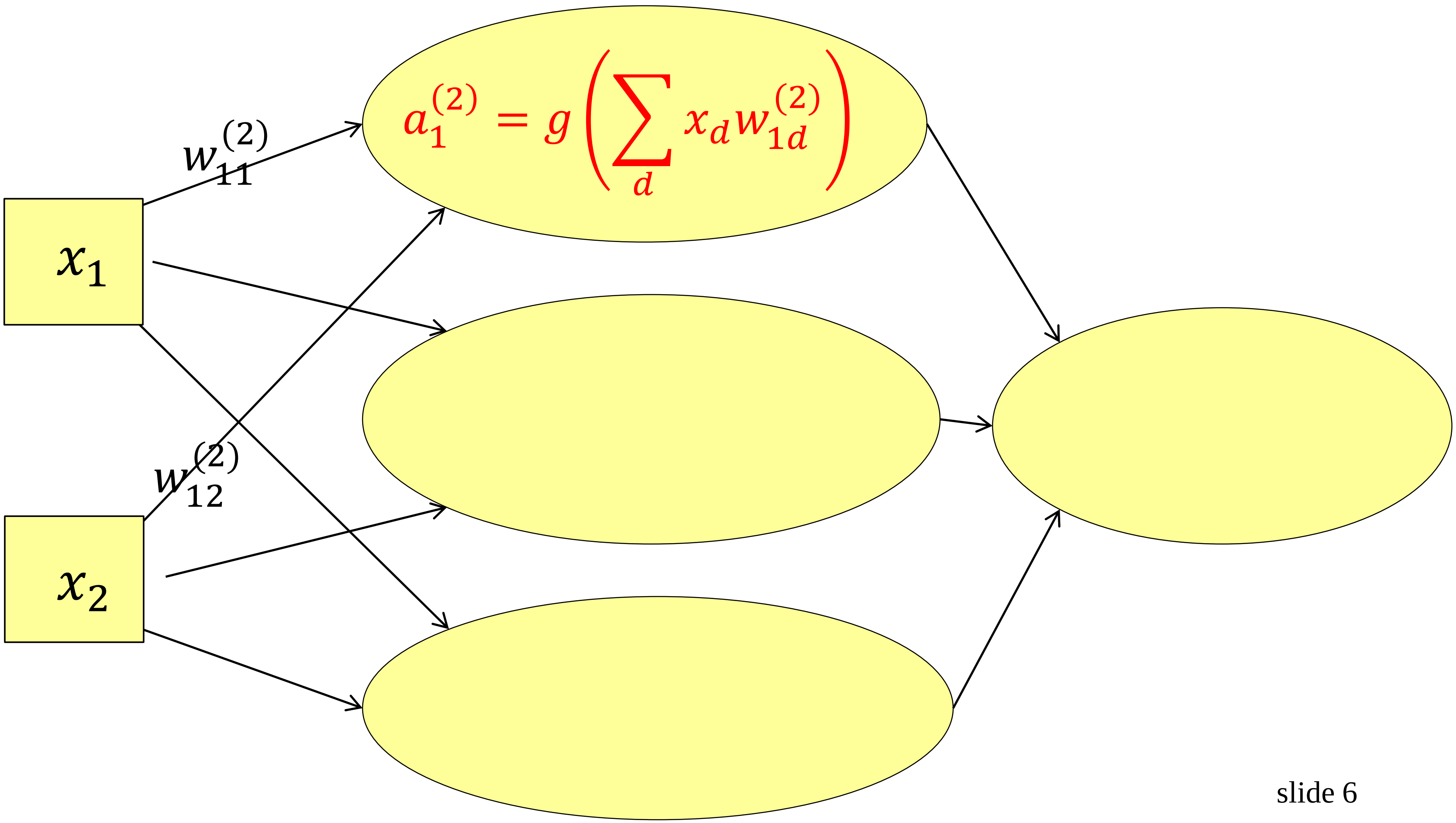
Multi-layer neural networks

- Standard way to connect Perceptrons
- Example: 1 hidden layer, 1 output layer



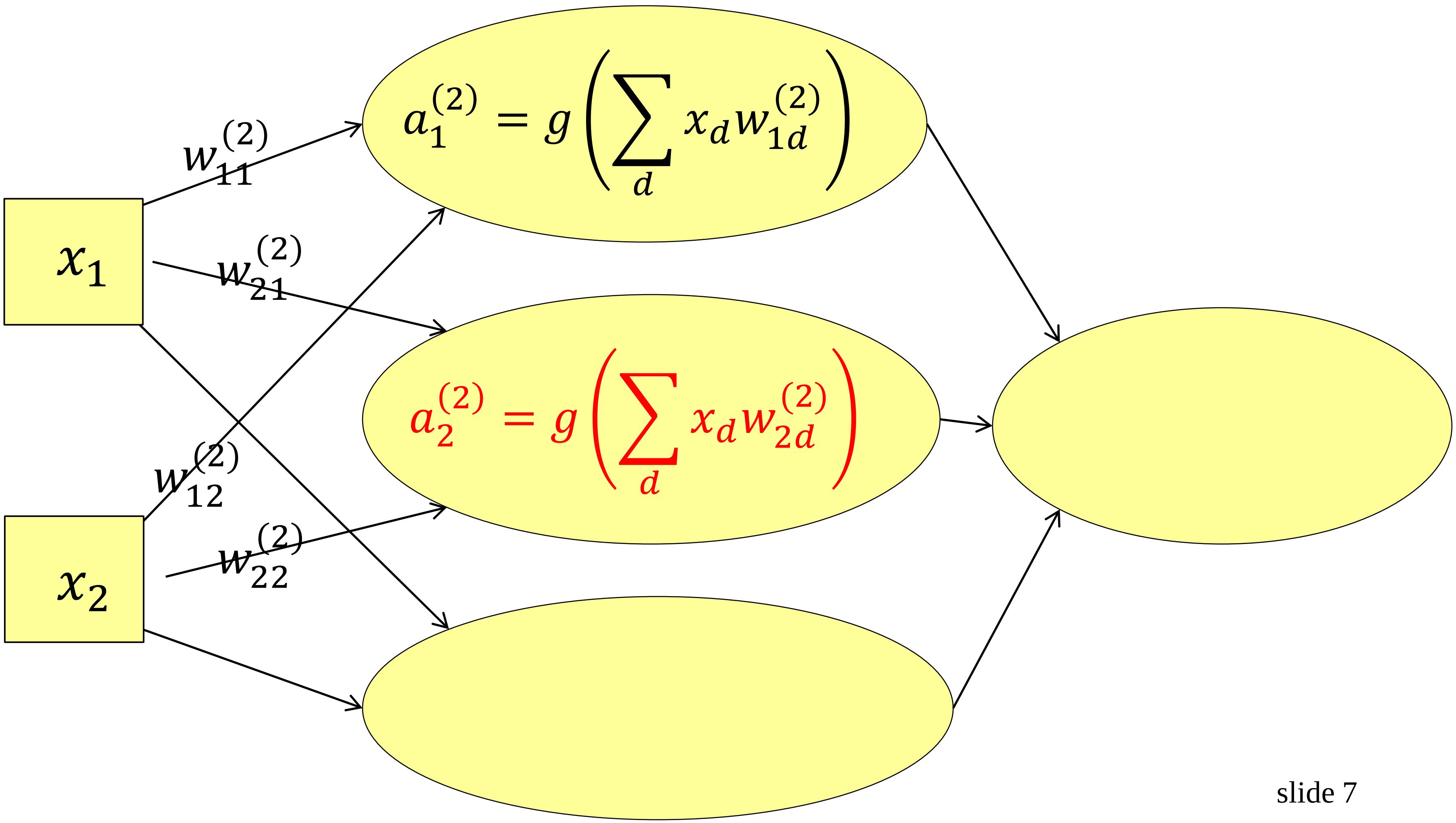
Multi-layer neural networks

- Standard way to connect Perceptrons
- Example: 1 hidden layer, 1 output layer



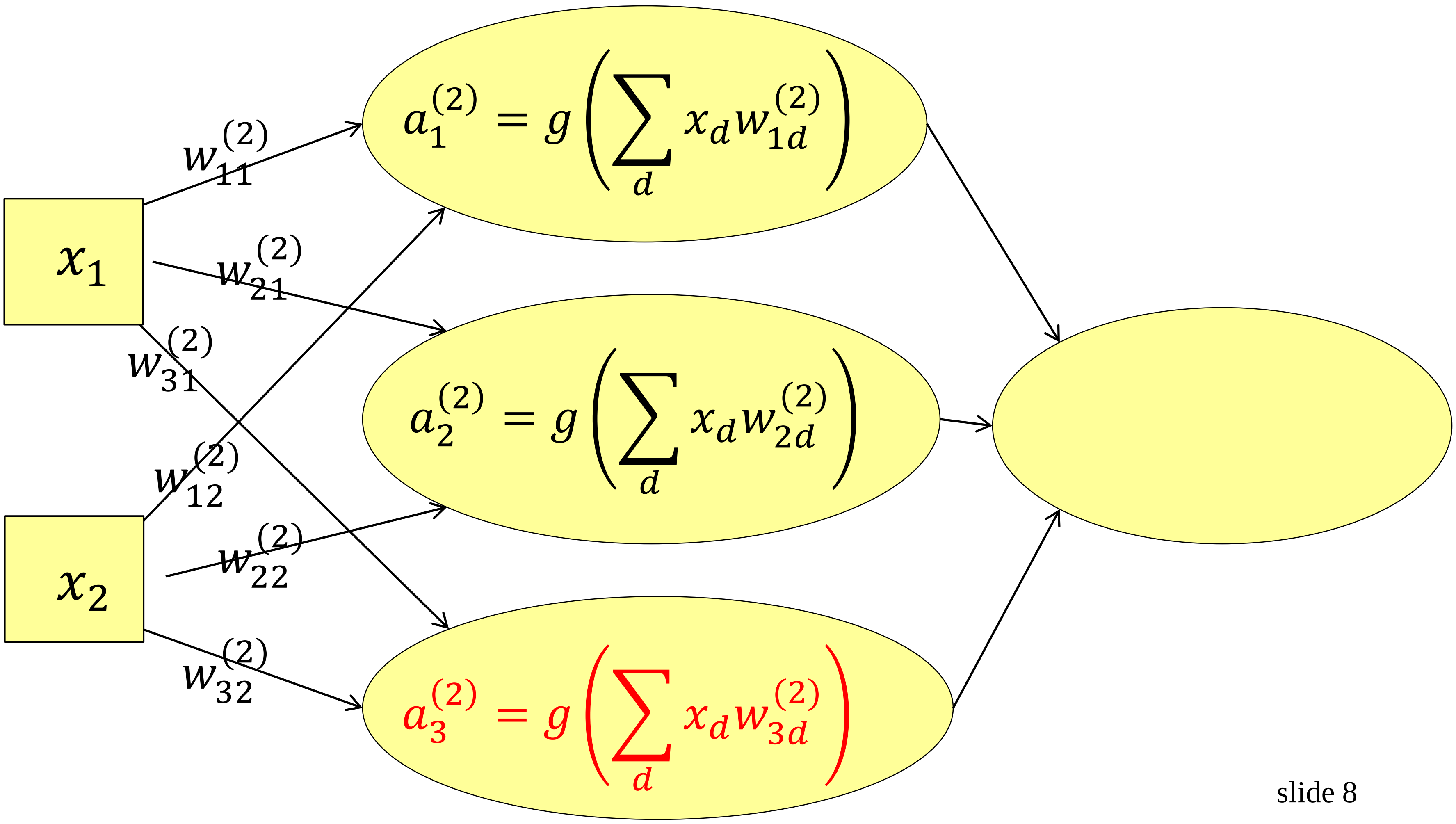
Multi-layer neural networks

- Standard way to connect Perceptrons
- Example: 1 hidden layer, 1 output layer



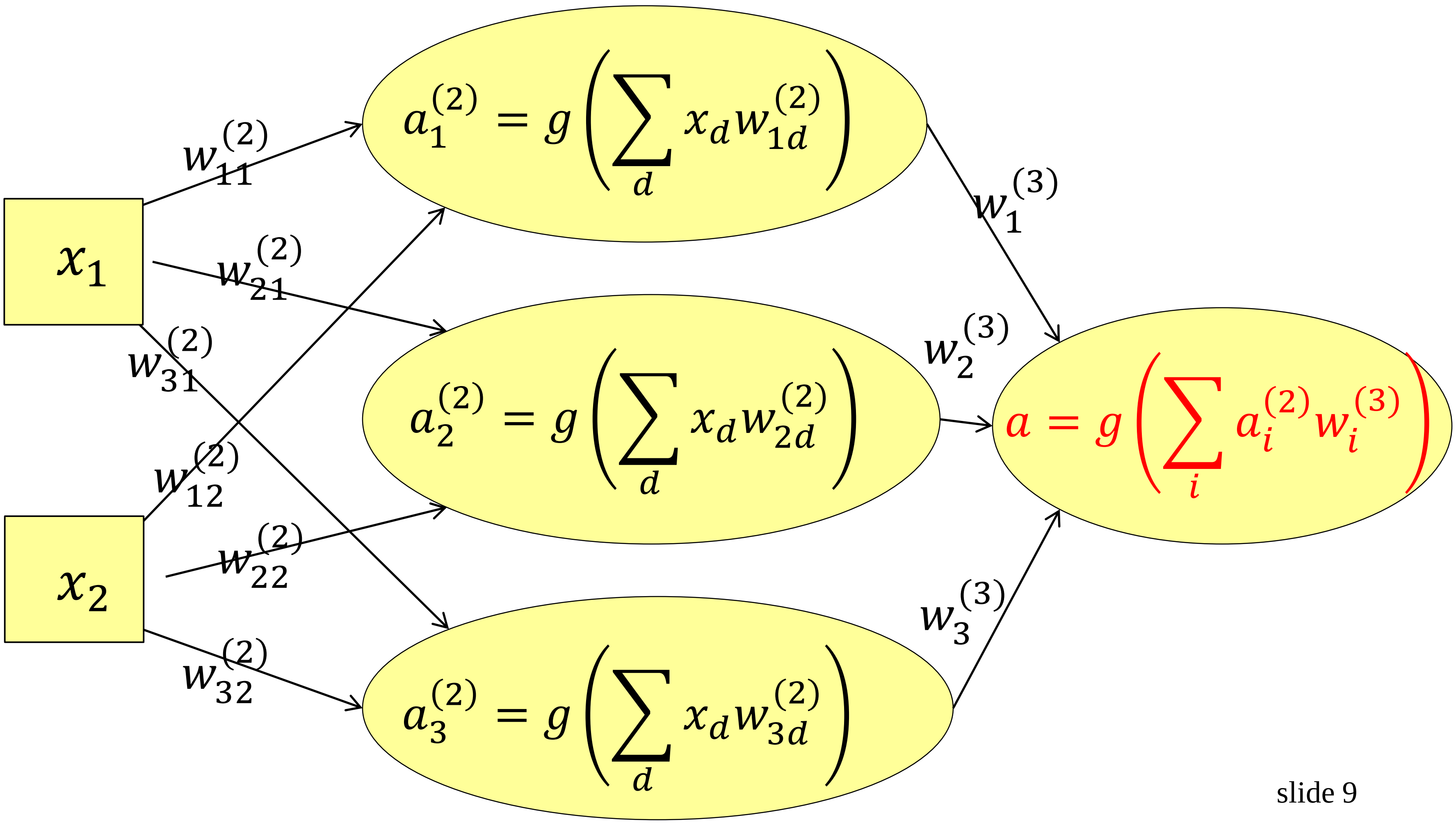
Multi-layer neural networks

- Standard way to connect Perceptrons
- Example: 1 hidden layer, 1 output layer



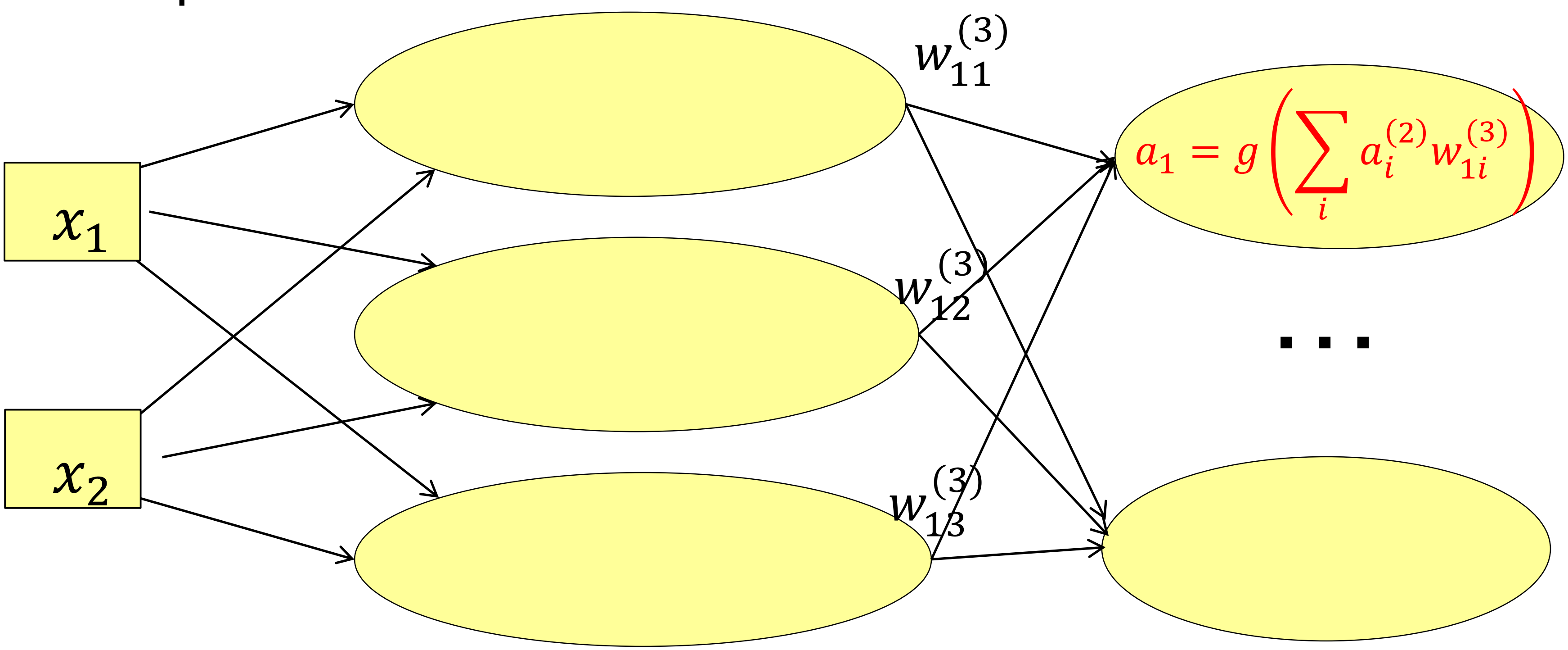
Multi-layer neural networks

- Standard way to connect Perceptrons
- Example: 1 hidden layer, 1 output layer



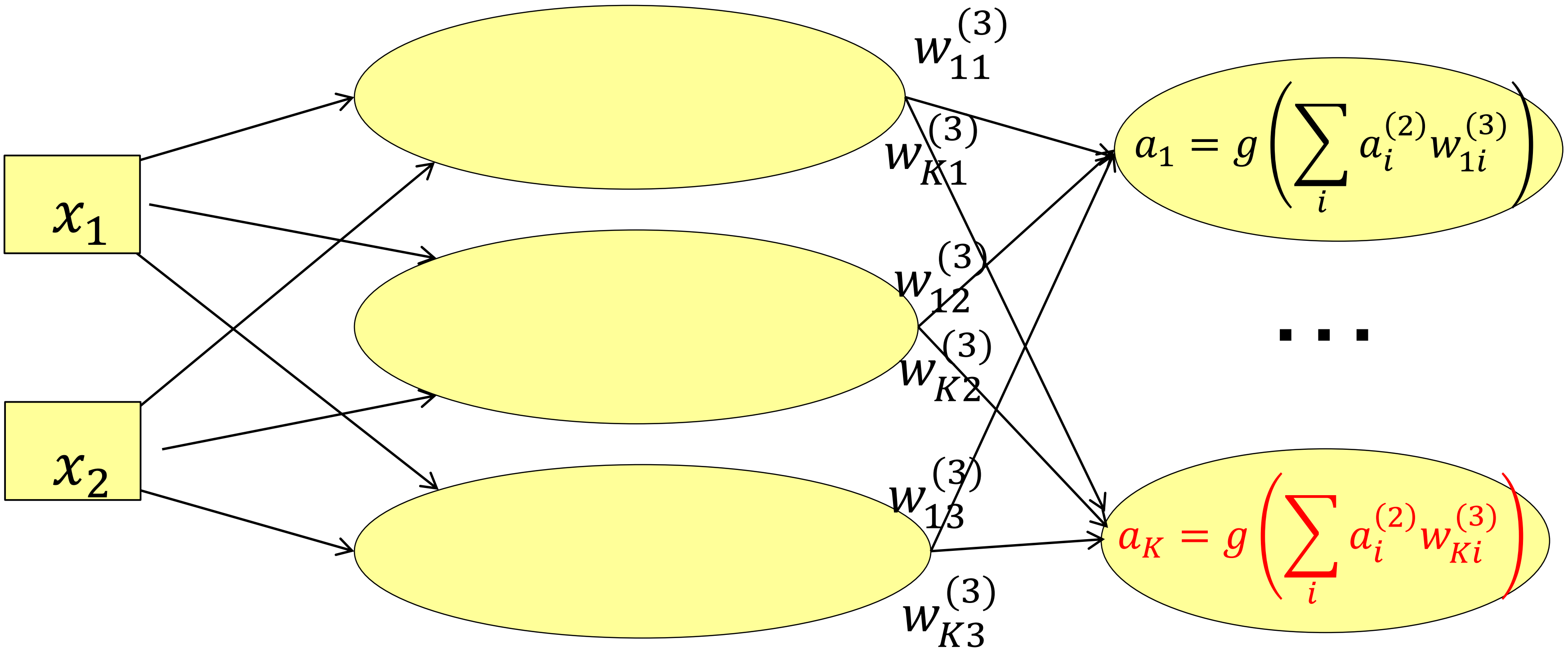
Neural net for K-way classification

- Use K output units
- Training: encode a label y by an indicator vector
  - class1=(1,0,0,…,0), class2=(0,1,0,…,0) etc.
- Test: choose the class corresponding to the largest output unit



Neural net for K-way classification

- Use K output units
- Training: encode a label y by an indicator vector
  - class1=(1,0,0,…,0), class2=(0,1,0,…,0) etc.
- Test: choose the class corresponding to the largest output unit



The (unlimited) power of neural network

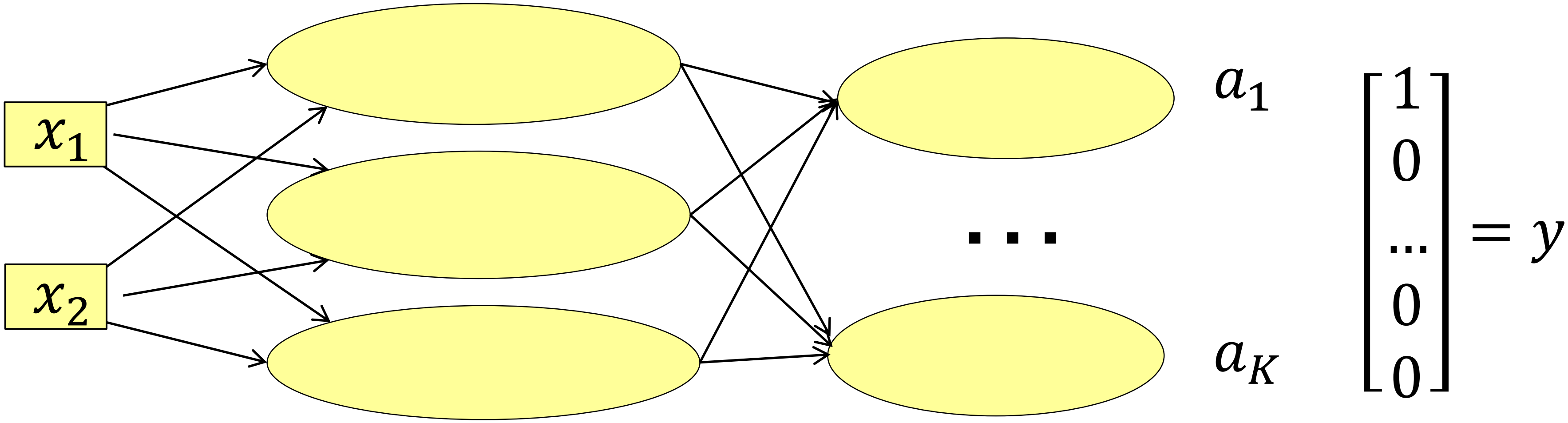
- In theory
  - we don’t need too many layers:
  - 1-hidden-layer net with enough hidden units can represent any continuous function of the inputs with arbitrary accuracy
  - 2-hidden-layer net can even represent discontinuous functions

Learning in neural network

- Again we will minimize the error (K outputs):

$$E = \frac{1}{2}\sum_{x \in D} E_x , \qquad E_x = \|y - a\|^2 = \sum_{c=1}^{K} (a_c - y_c)^2$$

- x: one training point in the training set D
- a_c: the c-th output for the training point x
- y_c: the c-th element of the label indicator vector for x



Learning in neural network

- Again we will minimize the error (K outputs):

$$E = \frac{1}{2}\sum_{x \in D} E_x, \qquad E_x = \|y - a\|^2 = \sum_{c=1}^{K}(a_c - y_c)^2$$

- x: one training point in the training set D
- a_c: the c-th output for the training point x
- y_c: the c-th element of the label indicator vector for x
- Our variables are all the weights w on all the edges
  - Apparent difficulty: we don't know the 'correct' output of hidden units

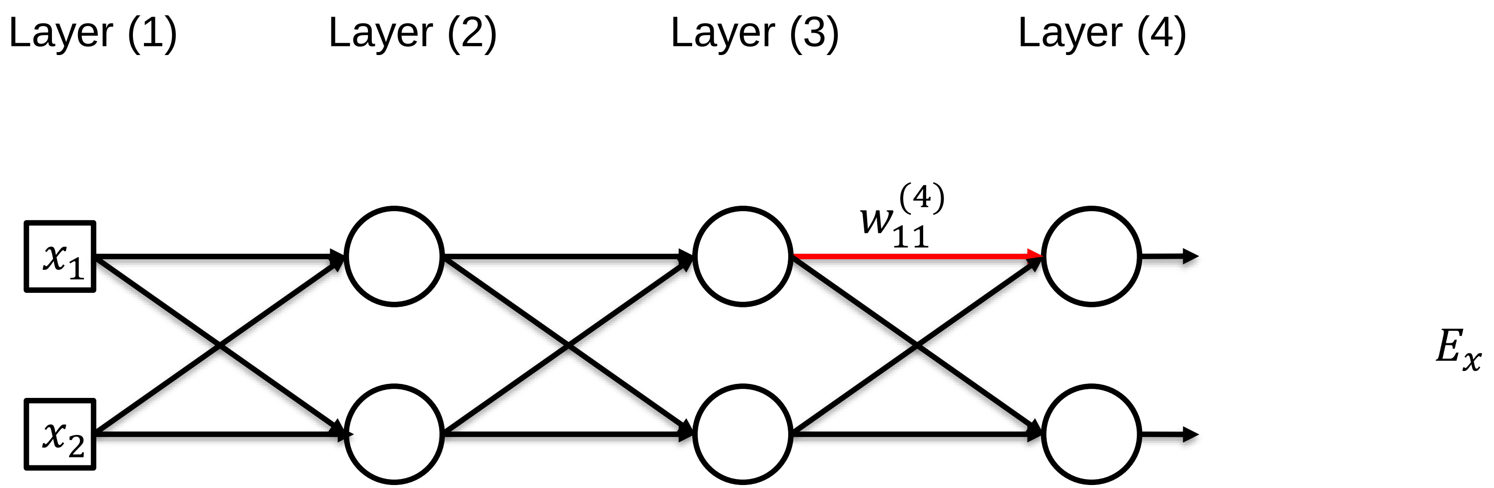
Learning in neural network

- Again we will minimize the error (K outputs):

$$E = \frac{1}{2}\sum_{x \in D} E_x , \qquad E_x = \|y - a\|^2 = \sum_{c=1}^{K} (a_c - y_c)^2$$

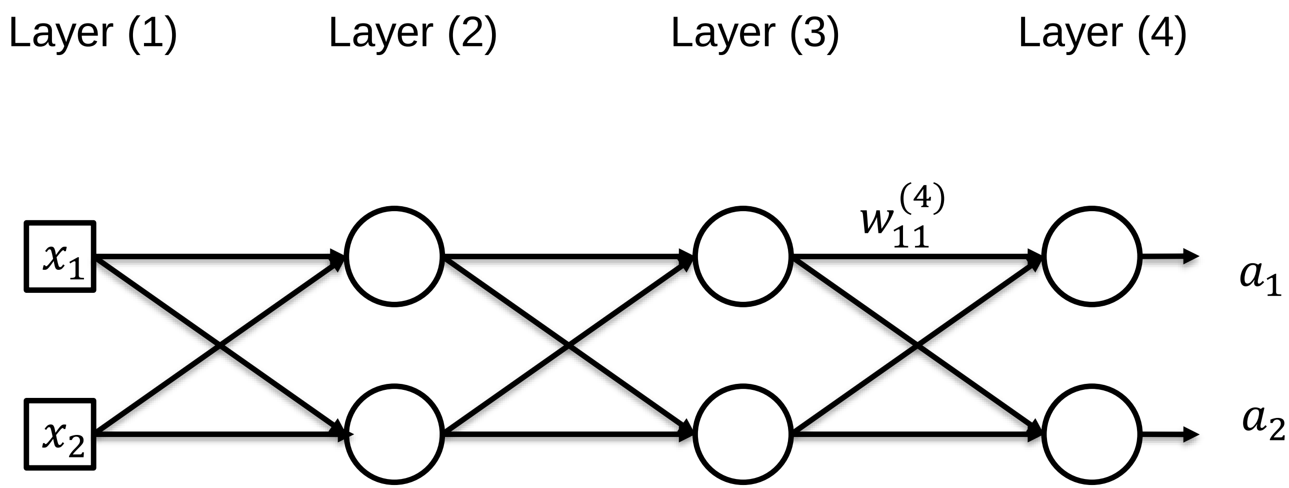
- x: one training point in the training set D
- a_c: the c-th output for the training point x
- y_c: the c-th element of the label indicator vector for x
- Our variables are all the weights w on all the edges
  - Apparent difficulty: we don't know the 'correct' output of hidden units
  - It turns out to be OK: we can still do gradient descent. The trick you need is the chain rule
  - The algorithm is known as **back-propagation**

Gradient (on one data point)



want to compute $\frac{\partial E_x}{\partial w_{11}^{(4)}}$

Gradient (on one data point)

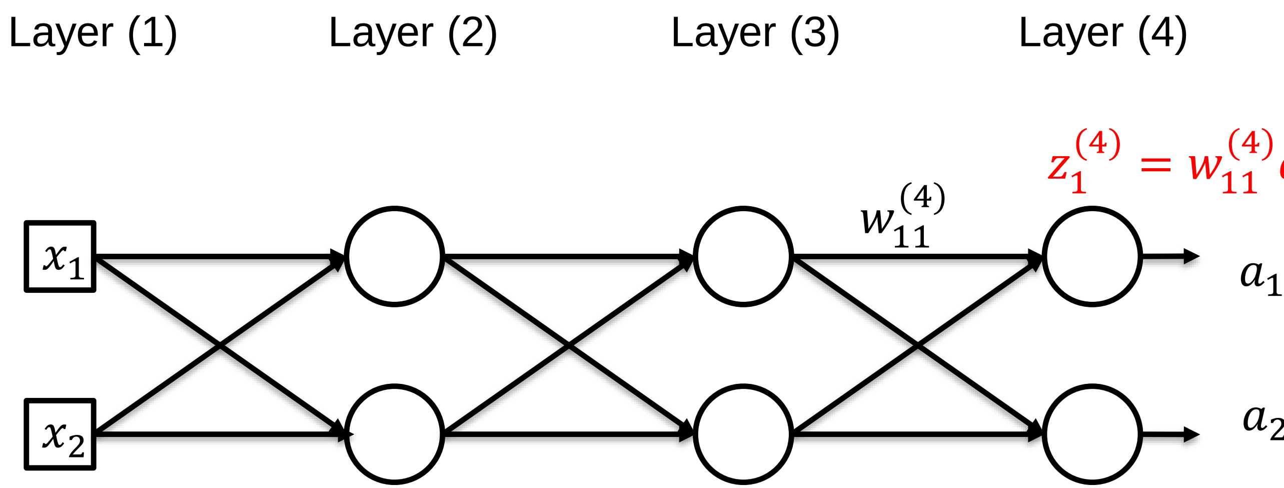


$$E_x = \|y - a\|^2$$

$$a_1 \xrightarrow{\|y - a\|^2} E_x$$

Gradient (on one data point)

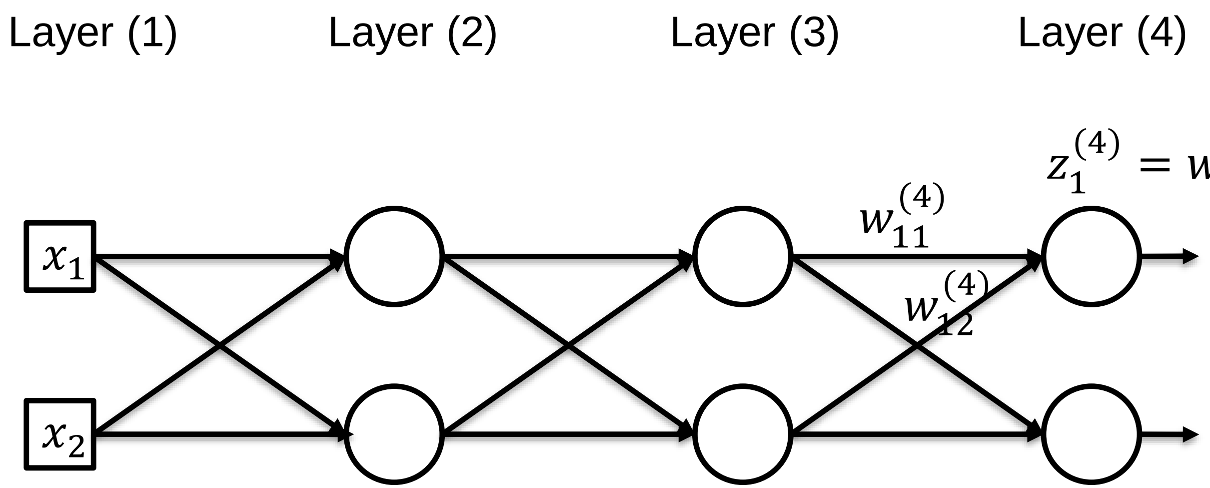
$$z_1^{(4)} = w_{11}^{(4)} a_1^{(3)} + w_{12}^{(4)} a_2^{(3)}$$



$$E_x = \|y - a\|^2$$

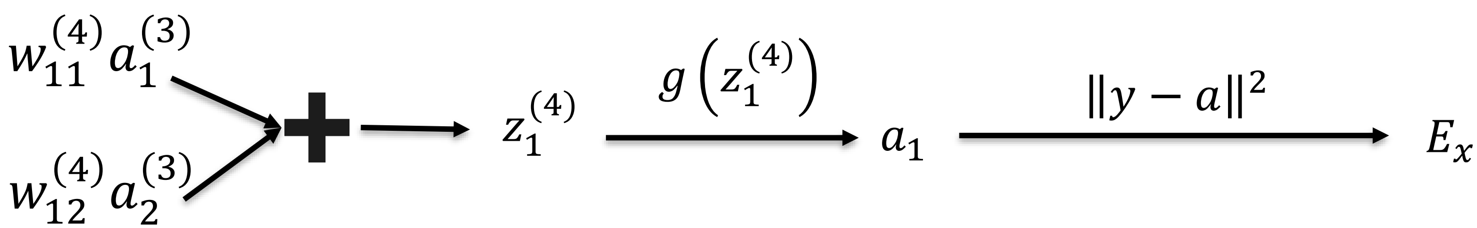
$$z_1^{(4)} \xrightarrow{\quad g\left(z_1^{(4)}\right) \quad} a_1 \xrightarrow{\qquad \|y - a\|^2 \qquad} E_x$$

Gradient (on one data point)

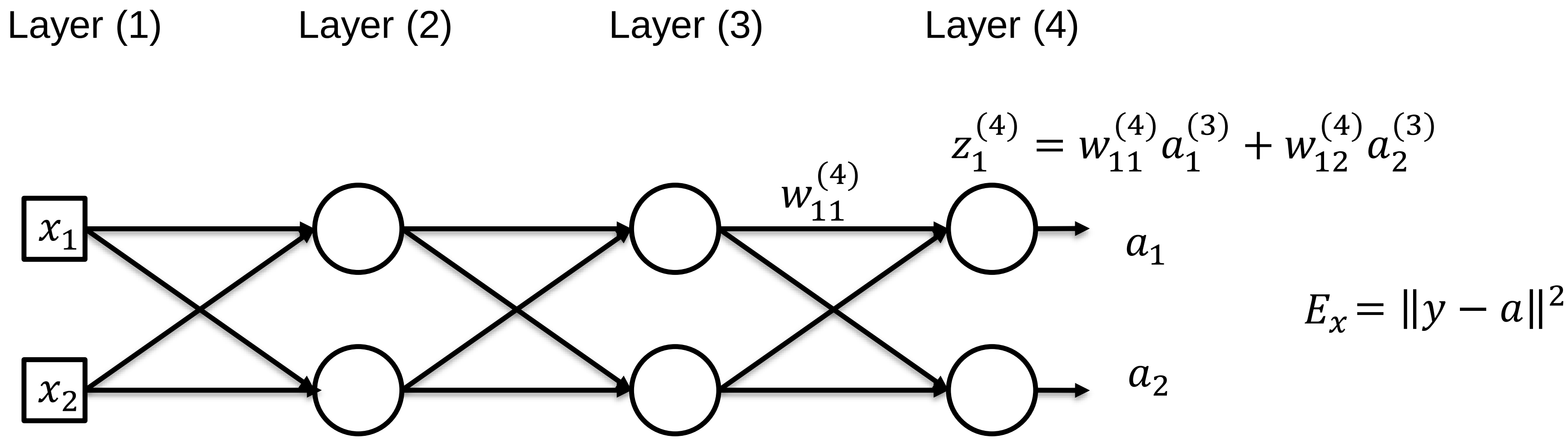


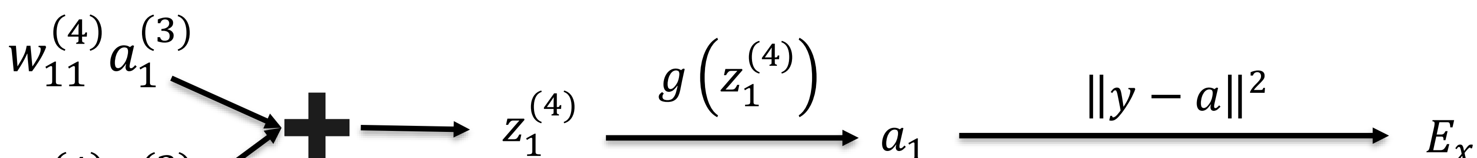
$$z_1^{(4)} = w_{11}^{(4)} a_1^{(3)} + w_{12}^{(4)} a_2^{(3)}$$

$$E_x = \|y - a\|^2$$



Gradient (on one data point)





$$\frac{\partial a_1}{\partial z_1^{(4)}} = g'\left(z_1^{(4)}\right) \qquad \frac{\partial E_{\boldsymbol{x}}}{\partial a_1} = 2(a_1 - y_1)$$

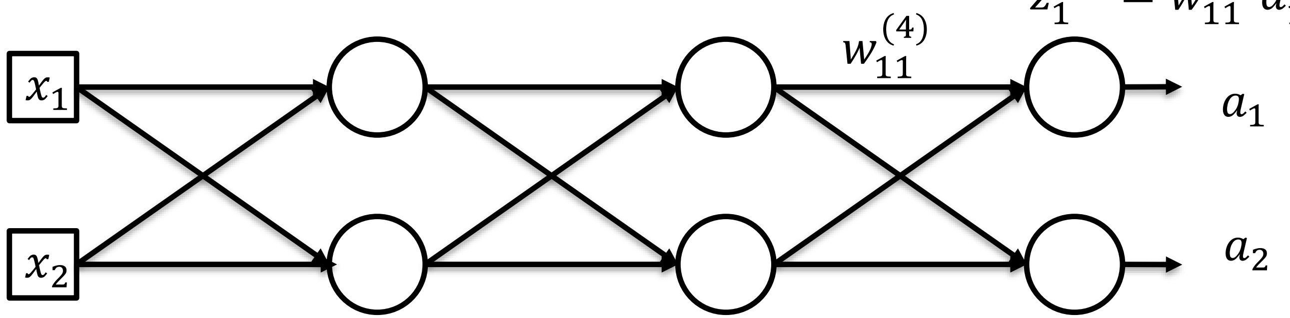
By Chain Rule:

$$\frac{\partial E_{\boldsymbol{x}}}{\partial w_{11}^{(4)}} = \frac{\partial E_{\boldsymbol{x}}}{\partial a_1} \frac{\partial a_1}{\partial z_1^{(4)}} \frac{\partial z_1^{(4)}}{\partial w_{11}^{(4)}}$$

Gradient (on one data point)

Layer (1) Layer (2) Layer (3) Layer (4)

$$z_1^{(4)} = w_{11}^{(4)} a_1^{(3)} + w_{12}^{(4)} a_2^{(3)}$$



$$E_x = \|y - a\|^2$$

$$w_{11}^{(4)} a_1^{(3)},\quad w_{12}^{(4)} a_2^{(3)} \longrightarrow + \longrightarrow z_1^{(4)} \xrightarrow{g\left(z_1^{(4)}\right)} a_1 \xrightarrow{\|y-a\|^2} E_x$$

$$\frac{\partial a_1}{\partial z_1^{(4)}} = g'\left(z_1^{(4)}\right) \qquad \frac{\partial E_{\boldsymbol{x}}}{\partial a_1} = 2(a_1 - y_1)$$

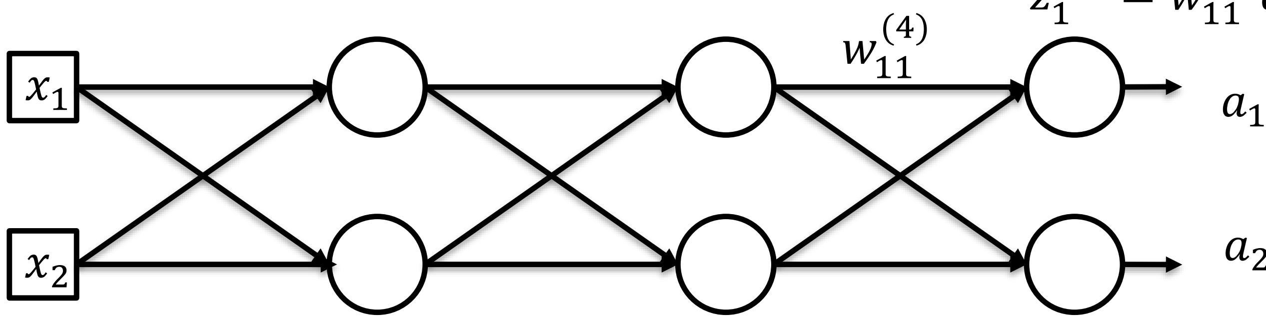
By Chain Rule:

$$\frac{\partial E_{\boldsymbol{x}}}{\partial w_{11}^{(4)}} = 2(a_1 - y_1) g'\left(z_1^{(4)}\right) \frac{\partial z_1^{(4)}}{\partial w_{11}^{(4)}}$$

Gradient (on one data point)

Layer (1) Layer (2) Layer (3) Layer (4)

$$z_1^{(4)} = w_{11}^{(4)} a_1^{(3)} + w_{12}^{(4)} a_2^{(3)}$$



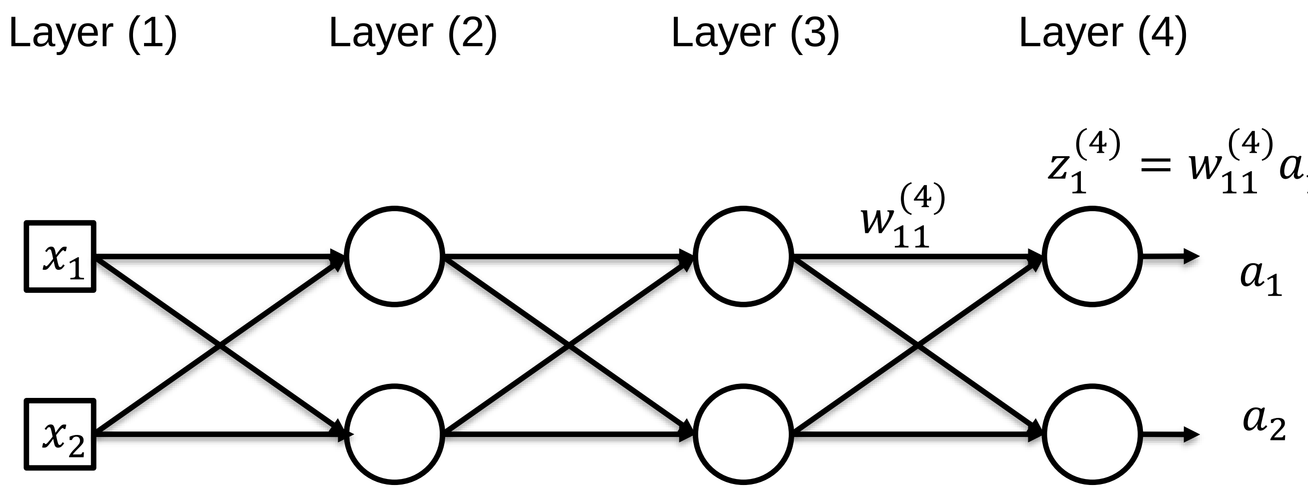
$$E_x = \|y - a\|^2$$

$$w_{11}^{(4)} a_1^{(3)},\ w_{12}^{(4)} a_2^{(3)} \rightarrow + \longrightarrow z_1^{(4)} \xrightarrow{g\left(z_1^{(4)}\right)} a_1 \xrightarrow{\|y-a\|^2} E_x$$

$$\frac{\partial a_1}{\partial z_1^{(4)}} = g'\left(z_1^{(4)}\right) \qquad \frac{\partial E_{\boldsymbol{x}}}{\partial a_1} = 2(a_1 - y_1)$$

By Chain Rule:
$$\frac{\partial E_{\boldsymbol{x}}}{\partial w_{11}^{(4)}} = 2(a_1 - y_1) g'\left(z_1^{(4)}\right) a_1^{(3)}$$

Gradient (on one data point)



$$z_1^{(4)} = w_{11}^{(4)} a_1^{(3)} + w_{12}^{(4)} a_2^{(3)}$$

$$E_x = \|y - a\|^2$$

$$w_{11}^{(4)} a_1^{(3)}, \; w_{12}^{(4)} a_2^{(3)} \longrightarrow + \longrightarrow z_1^{(4)} \xrightarrow{g\left(z_1^{(4)}\right)} a_1 \xrightarrow{\|y-a\|^2} E_x$$

$$\frac{\partial a_1}{\partial z_1^{(4)}} = g'\left(z_1^{(4)}\right) \qquad \frac{\partial E_{\boldsymbol{x}}}{\partial a_1} = 2(a_1 - y_1)$$

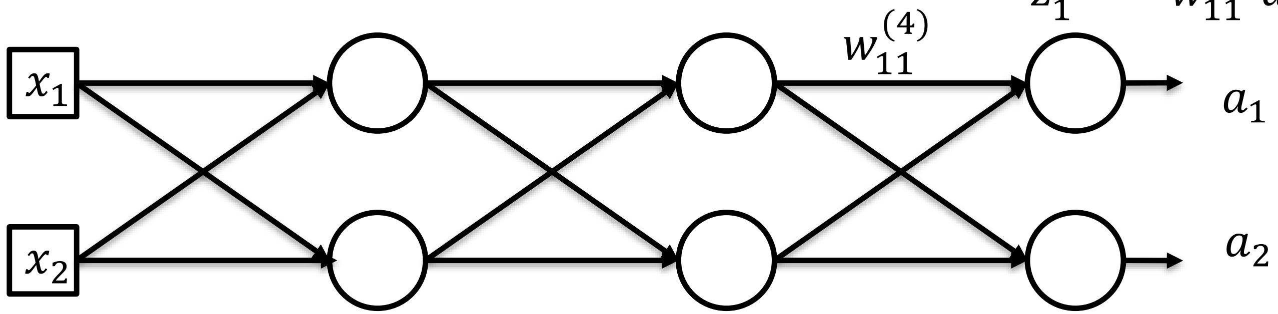
By Chain Rule:

$$\frac{\partial E_{\boldsymbol{x}}}{\partial w_{11}^{(4)}} = 2(a_1 - y_1) g\left(z_1^{(4)}\right)\left(1 - g\left(z_1^{(4)}\right)\right) a_1^{(3)}$$

Gradient (on one data point)

Layer (1) Layer (2) Layer (3) Layer (4)

$$z_1^{(4)} = w_{11}^{(4)} a_1^{(3)} + w_{12}^{(4)} a_2^{(3)}$$



$$E_x = \|y - a\|^2$$

$$w_{11}^{(4)} a_1^{(3)} \;+\; w_{12}^{(4)} a_2^{(3)} \longrightarrow z_1^{(4)} \xrightarrow{g\left(z_1^{(4)}\right)} a_1 \xrightarrow{\|y - a\|^2} E_x$$

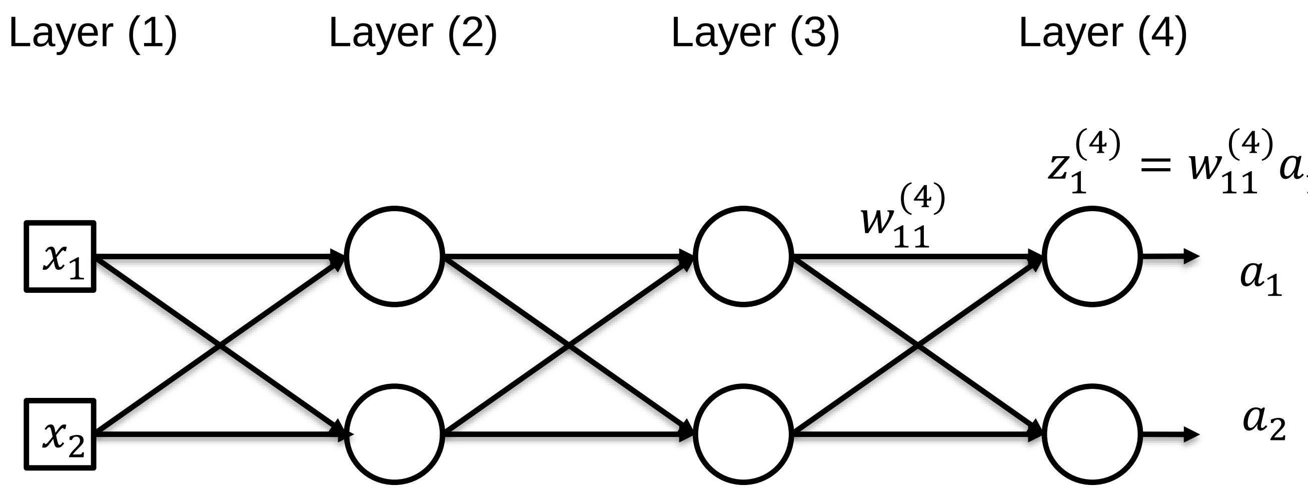
$$\frac{\partial a_1}{\partial z_1^{(4)}} = g'\left(z_1^{(4)}\right) \qquad \frac{\partial E_{\boldsymbol{x}}}{\partial a_1} = 2(a_1 - y_1)$$

By Chain Rule:

$$\frac{\partial E_{\boldsymbol{x}}}{\partial w_{11}^{(4)}} = 2(a_1 - y_1) a_1 (1 - a_1) a_1^{(3)}$$

Gradient (on one data point)

$$z_1^{(4)} = w_{11}^{(4)} a_1^{(3)} + w_{12}^{(4)} a_2^{(3)}$$



$$E_x = \|y - a\|^2$$

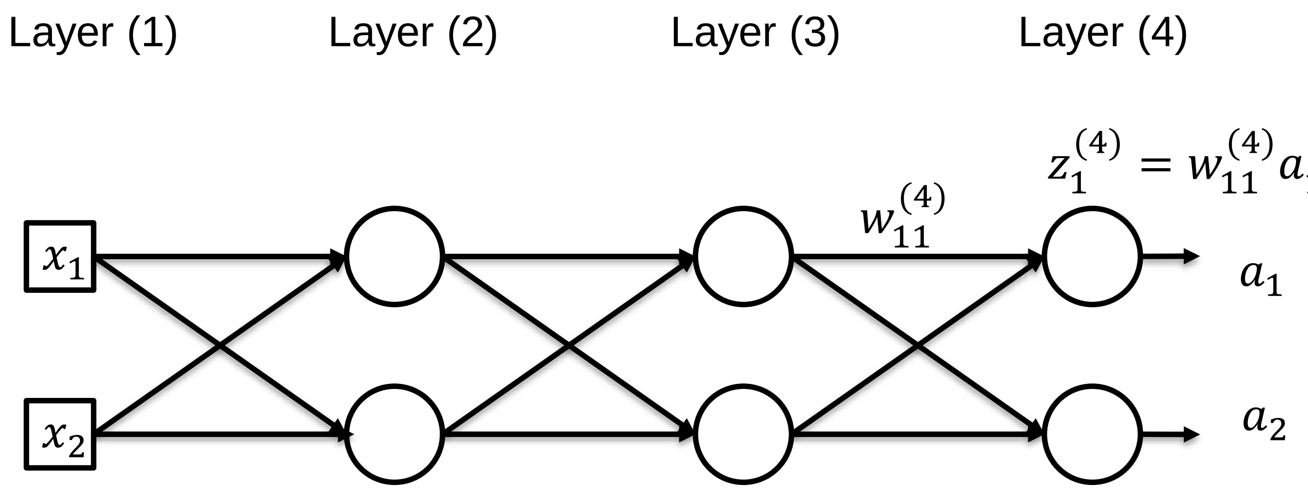
$$w_{11}^{(4)} a_1^{(3)},\ w_{12}^{(4)} a_2^{(3)} \xrightarrow{+} z_1^{(4)} \xrightarrow{g\left(z_1^{(4)}\right)} a_1 \xrightarrow{\|y-a\|^2} E_x$$

$$\frac{\partial a_1}{\partial z_1^{(4)}} = g'\left(z_1^{(4)}\right) \qquad \frac{\partial E_{\boldsymbol{x}}}{\partial a_1} = 2(a_1 - y_1)$$

By Chain Rule: $$\frac{\partial E_{\boldsymbol{x}}}{\partial w_{11}^{(4)}} = 2(a_1 - y_1)a_1(1 - a_1)a_1^{(3)}$$

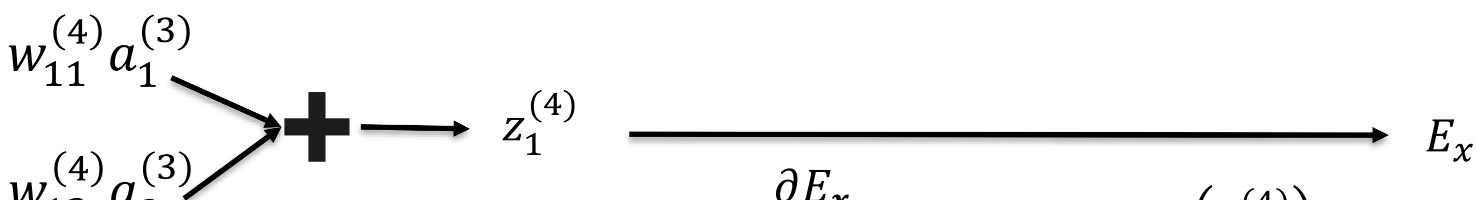
Can be computed by network activation

Backpropagation



$$z_1^{(4)} = w_{11}^{(4)} a_1^{(3)} + w_{12}^{(4)} a_2^{(3)}$$

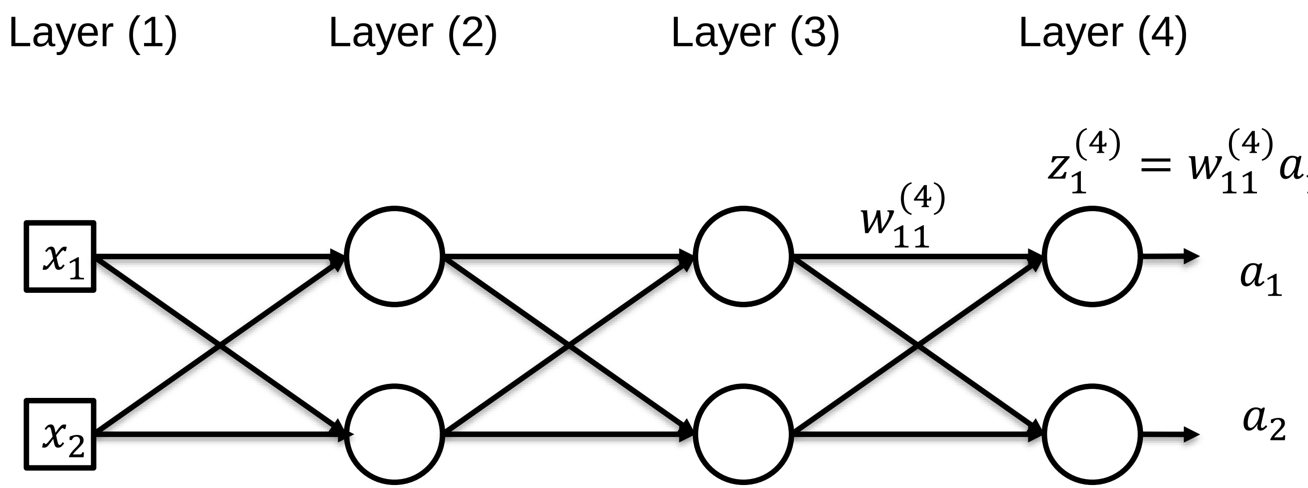
$$E_x = \|y - a\|^2$$



$$\frac{\partial E_{\boldsymbol{x}}}{\partial z_1^{(4)}} = 2(a_1 - y_1) g'\left(z_1^{(4)}\right)$$

By Chain Rule: $$\frac{\partial E_{\boldsymbol{x}}}{\partial w_{11}^{(4)}} = 2(a_1 - y_1) a_1 (1 - a_1) a_1^{(3)}$$

Backpropagation



$$z_1^{(4)} = w_{11}^{(4)} a_1^{(3)} + w_{12}^{(4)} a_2^{(3)}$$

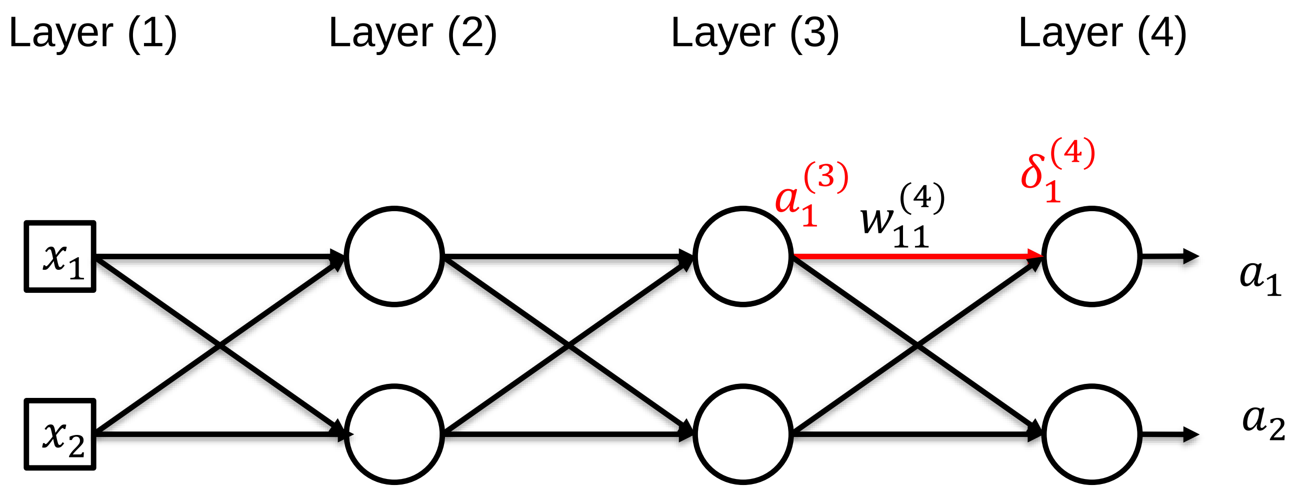
$$E_x = \|y - a\|^2$$

$$\left.\begin{matrix} w_{11}^{(4)} a_1^{(3)} \\ w_{12}^{(4)} a_2^{(3)} \end{matrix}\right\} \rightarrow + \longrightarrow z_1^{(4)} \longrightarrow E_x$$

$$\color{red}{\delta_1^{(4)}} = \frac{\partial E_{\boldsymbol{x}}}{\partial z_1^{(4)}} = 2(a_1 - y_1) g'\left(z_1^{(4)}\right)$$

By Chain Rule: $$\frac{\partial E_{\boldsymbol{x}}}{\partial w_{11}^{(4)}} = \boxed{2(a_1 - y_1) a_1 (1 - a_1)} a_1^{(3)}$$

Backpropagation



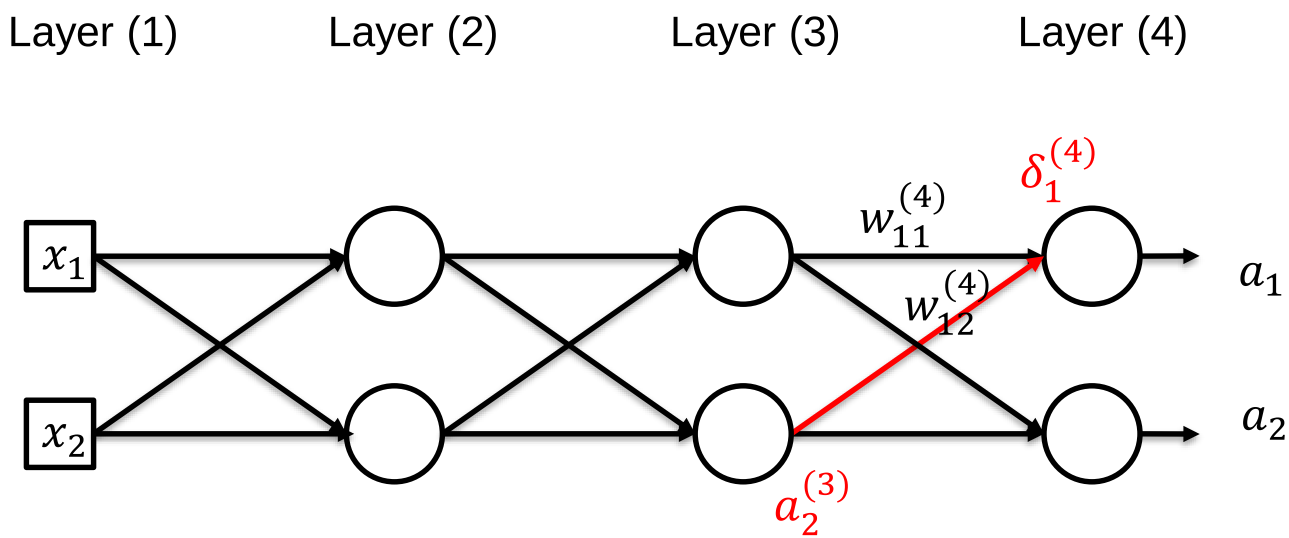
$$E_x = \|y - a\|^2$$

$$w_{11}^{(4)} a_1^{(3)} + w_{12}^{(4)} a_2^{(3)} \longrightarrow z_1^{(4)} \longrightarrow E_x$$

$$\delta_1^{(4)} = \frac{\partial E_{\boldsymbol{x}}}{\partial z_1^{(4)}} = 2(a_1 - y_1)g'\left(z_1^{(4)}\right)$$

By Chain Rule: $$\frac{\partial E_{\boldsymbol{x}}}{\partial w_{11}^{(4)}} = \delta_1^{(4)} a_1^{(3)}$$

Backpropagation



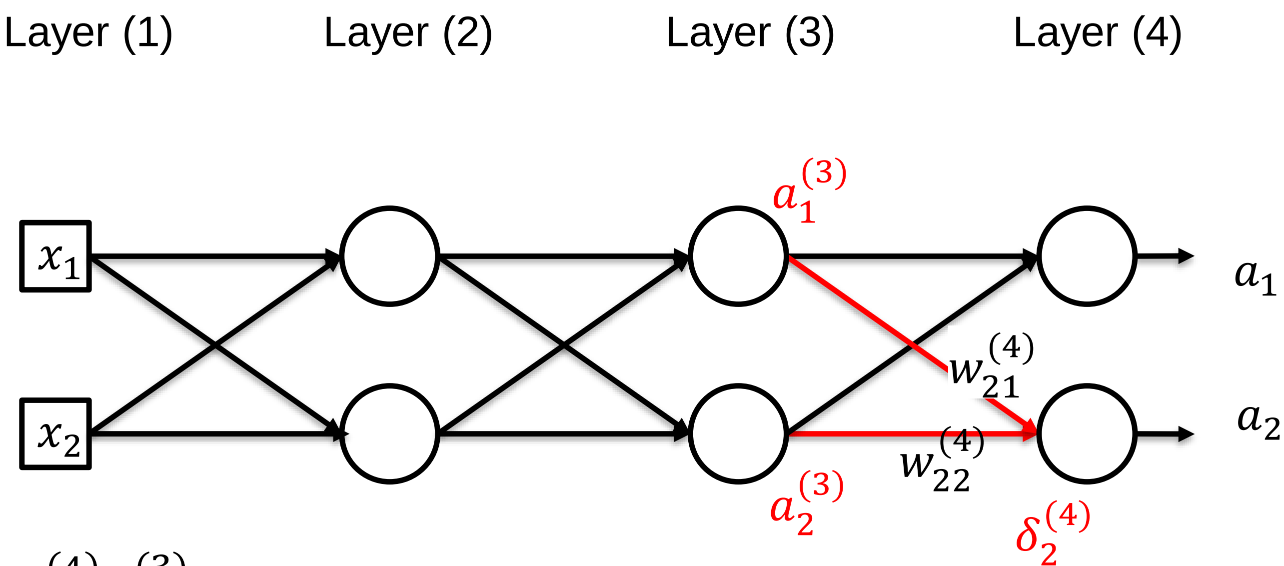
$$E_x = \|y - a\|^2$$

$$w_{11}^{(4)} a_1^{(3)},\quad w_{12}^{(4)} a_2^{(3)} \rightarrow + \rightarrow z_1^{(4)} \longrightarrow E_x$$

$$\delta_1^{(4)} = \frac{\partial E_{\boldsymbol{x}}}{\partial z_1^{(4)}} = 2(a_1 - y_1)g'\left(z_1^{(4)}\right)$$

By Chain Rule:
$$\frac{\partial E_{\boldsymbol{x}}}{\partial w_{11}^{(4)}} = \delta_1^{(4)} a_1^{(3)}, \qquad \frac{\partial E_{\boldsymbol{x}}}{\partial w_{12}^{(4)}} = \delta_1^{(4)} a_2^{(3)}$$

Backpropagation



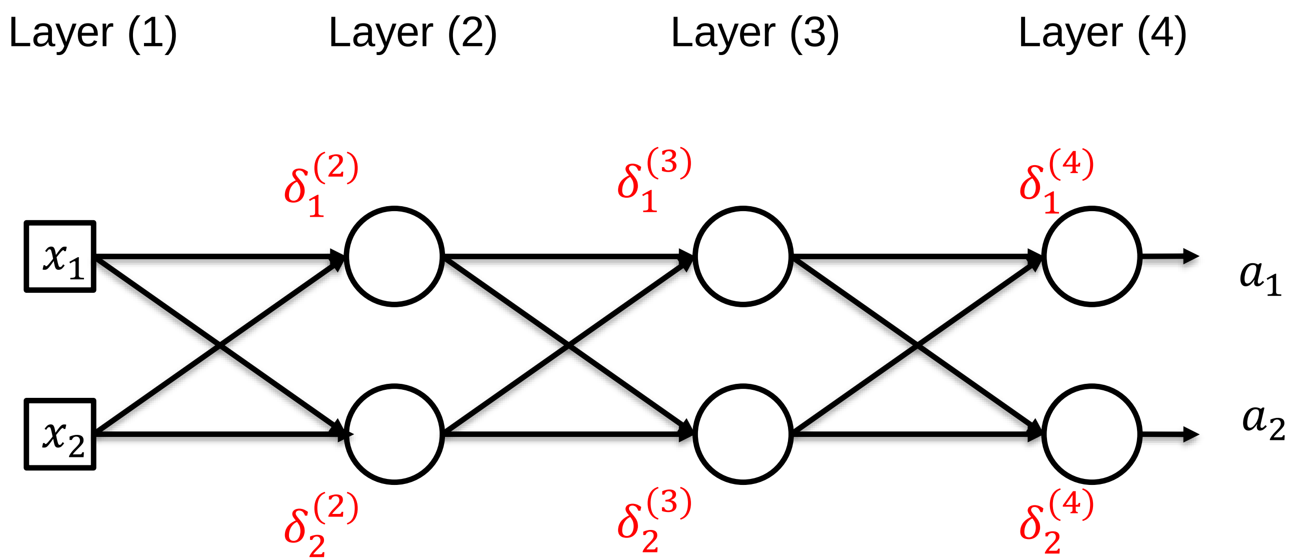
$$E_x = \|y - a\|^2$$

$$\left.\begin{array}{l} w_{11}^{(4)} a_1^{(3)} \\ w_{12}^{(4)} a_2^{(3)} \end{array}\right\} + \longrightarrow z_1^{(4)} \longrightarrow E_x$$

$$\delta_2^{(4)} = \frac{\partial E_{\boldsymbol{x}}}{\partial z_2^{(4)}} = 2(a_2 - y_2) g'\left(z_2^{(4)}\right)$$

By Chain Rule: $$\frac{\partial E_{\boldsymbol{x}}}{\partial w_{21}^{(4)}} = \delta_2^{(4)} a_1^{(3)}, \qquad \frac{\partial E_{\boldsymbol{x}}}{\partial w_{22}^{(4)}} = \delta_2^{(4)} a_2^{(3)}$$

Backpropagation



$$E_x = \|y - a\|^2$$

Thus, for any weight in the network:

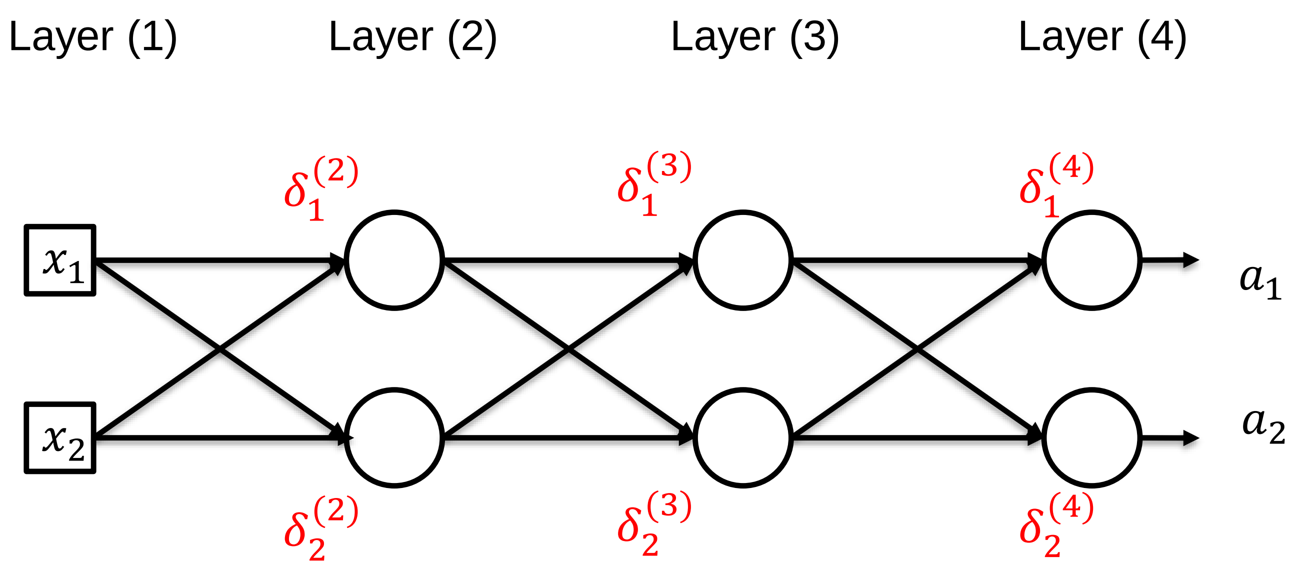
$$\boxed{\frac{\partial E_x}{\partial w_{jk}^{(l)}} = \delta_j^{(l)} a_k^{(l-1)}}$$

$\delta_j^{(l)}$: δ of j^{th} neuron in Layer l

$a_k^{(l-1)}$: Activation of k^{th} neuron in Layer $l-1$

$w_{jk}^{(l)}$: Weight from k^{th} neuron in Layer $l-1$ to j^{th} neuron in Layer l

Exercise



$$E_x = \|y - a\|^2$$

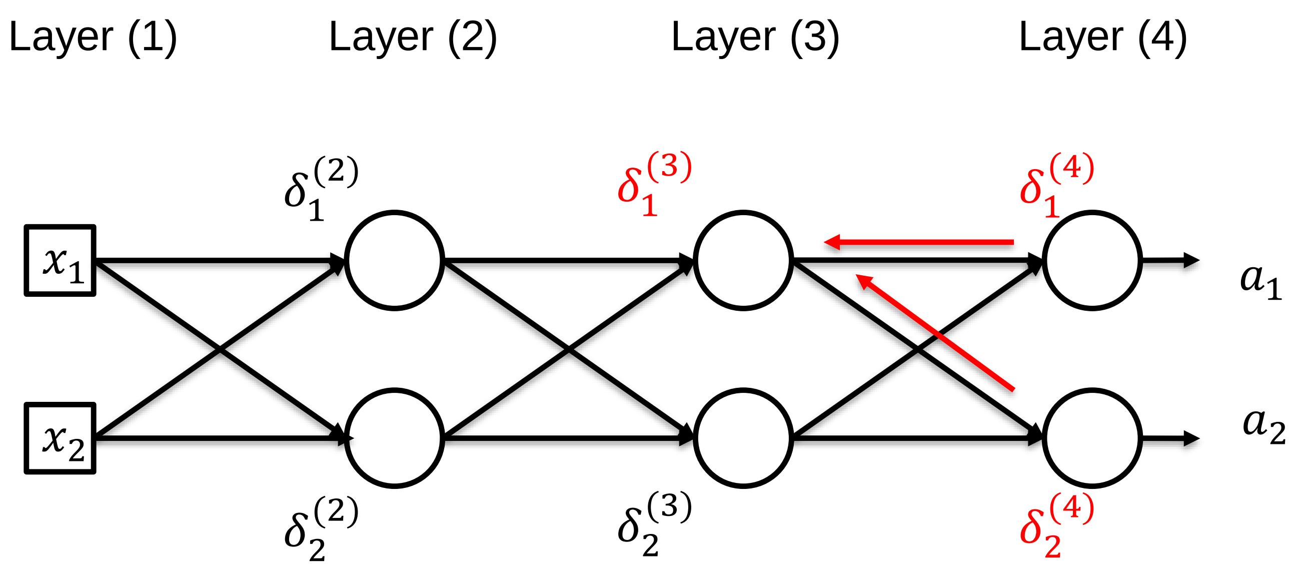
Show that for any bias in the network:

$$\boxed{\frac{\partial E_x}{\partial b_j^{(l)}} = \delta_j^{(l)}}$$

$\delta_j^{(l)}$: δ of j^{th} neuron in Layer l

$b_j^{(l)}$: bias for the j^{th} neuron in Layer l, i.e., $z_j^{(l)} = \sum_k w_{jk}^{(l)} a_k^{(l-1)} + b_j^{(l)}$

Backpropagation of δ



$$E_x = \|y - a\|^2$$

Thus, for any neuron in the network:

$$\boxed{\delta_j^{(l)} = \sum_k \delta_k^{(l+1)} w_{kj}^{(l+1)} \quad g'\left(z_j^{(l)}\right)}$$

$\delta_j^{(l)}$: δ of j^{th} Neuron in Layer l

$\delta_k^{(l+1)}$: δ of k^{th} Neuron in Layer $l+1$

$g'\left(z_j^{(l)}\right)$: derivative of j^{th} Neuron in Layer l w.r.t. its linear combination input

$w_{kj}^{(l+1)}$: Weight from j^{th} Neuron in Layer l to k^{th} Neuron in Layer $l+1$

Gradient descent with Backpropagation

1. Initialize Network with Random Weights and Biases
2. For each Training Image:
   a. Compute Activations for the Entire Network
   b. Compute δ for Neurons in the Output Layer using Network Activation and Desired Activation
   $$\delta_j^{(L)} = 2\left(y_j - a_j\right)a_j(1 - a_j)$$
   c. Compute δ for all Neurons in the previous Layers
   $$\delta_j^{(l)} = \sum_k \delta_k^{(l+1)} w_{kj}^{(l+1)} a_j^{(l)}(1 - a_j^{(l)})$$
   d. Compute Gradient of Cost w.r.t each Weight and Bias for the Training Image using δ
   $$\frac{\partial E_x}{\partial w_{jk}^{(l)}} = \delta_j^{(l)} a_k^{(l-1)} \qquad \frac{\partial E_x}{\partial b_j^{(l)}} = \delta_j^{(l)}$$

Gradient descent with Backpropagation

3. Average the Gradient w.r.t. each Weight and Bias over the Entire Training Set

$$\frac{\partial E}{\partial w_{jk}^{(l)}} = \frac{1}{n}\sum \frac{\partial E_{\boldsymbol{x}}}{\partial w_{jk}^{(l)}} \qquad \frac{\partial E}{\partial b_{j}^{(l)}} = \frac{1}{n}\sum \frac{\partial E_{\boldsymbol{x}}}{\partial b_{j}^{(l)}}$$

4. Update the Weights and Biases using Gradient Descent

$$w_{jk}^{(l)} \leftarrow w_{jk}^{(l)} - \eta \frac{\partial E}{\partial w_{jk}^{(l)}} \qquad b_{j}^{(l)} \leftarrow b_{j}^{(l)} - \eta \frac{\partial E}{\partial b_{j}^{(l)}}$$

5. Repeat Steps 2-4 till Cost reduces below an acceptable level