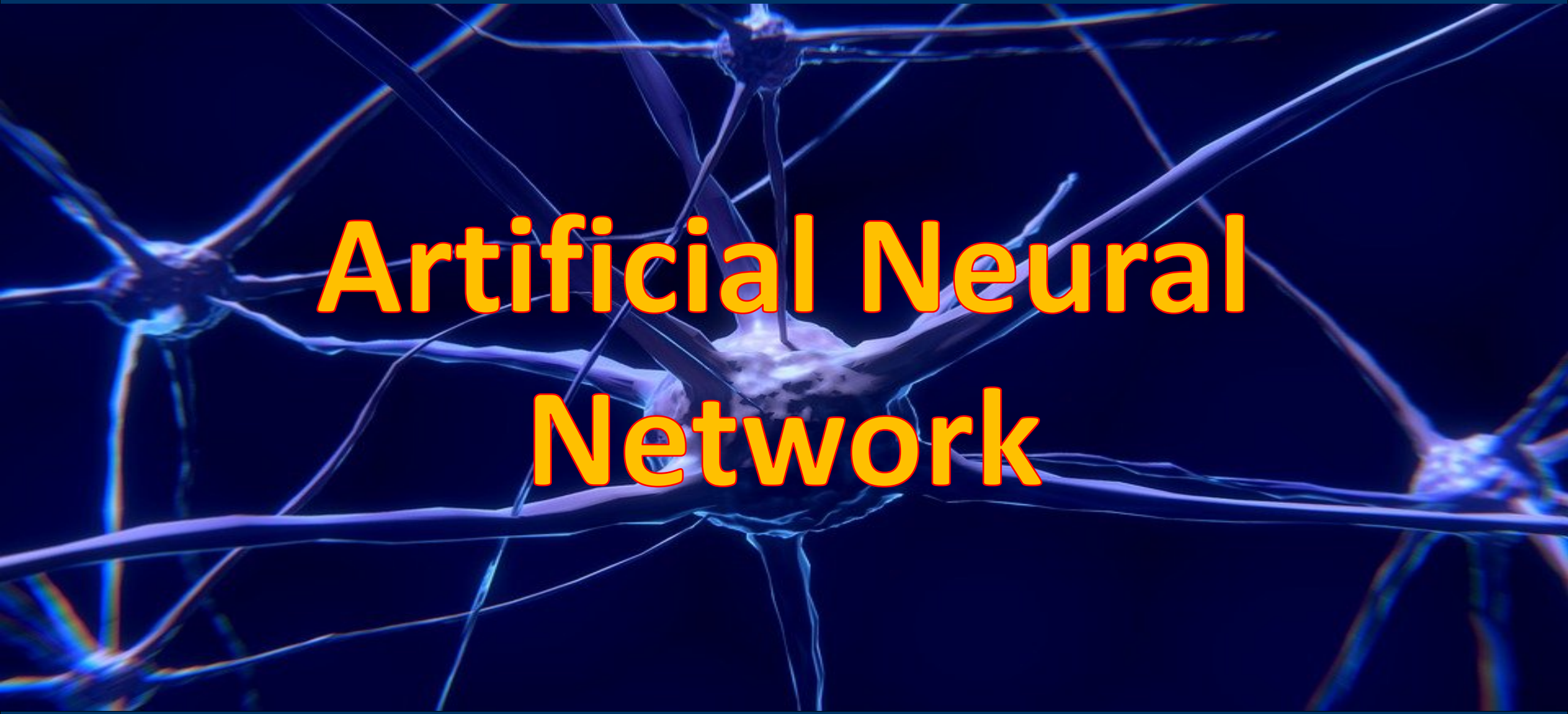
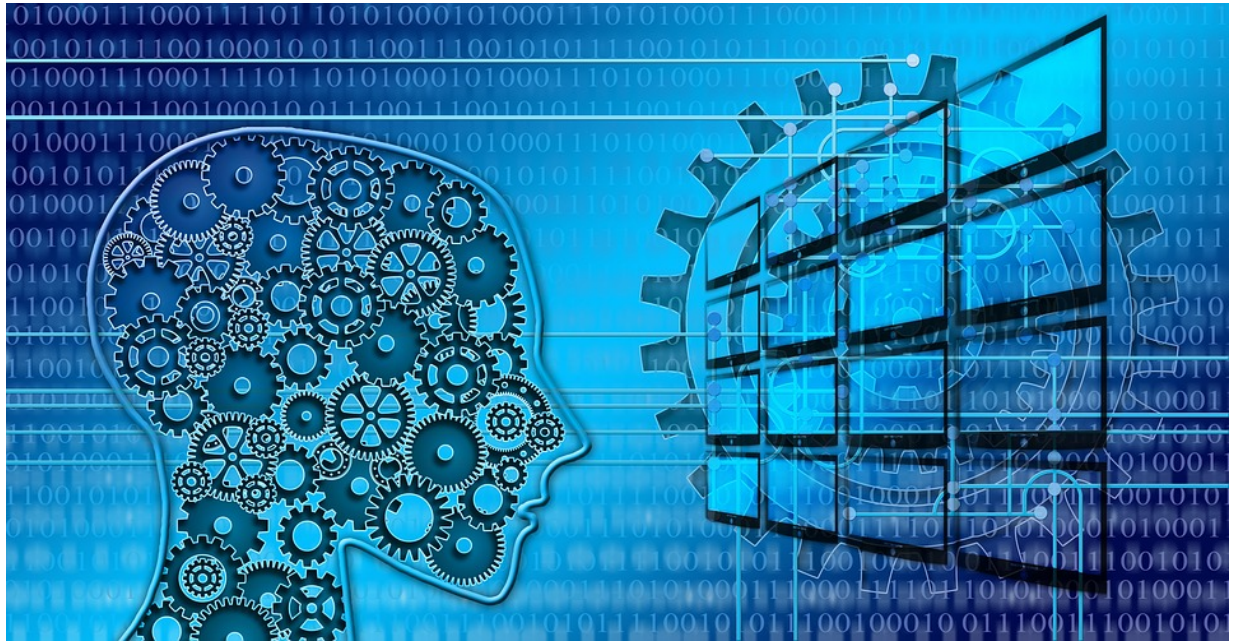
Advanced Machine Learning and Deep Learning

Mohammad Pourhomayoun
Assistant Professor
Computer Science Department
California State University, Los Angeles


Artificial Neural
Network

Training an ANN Model



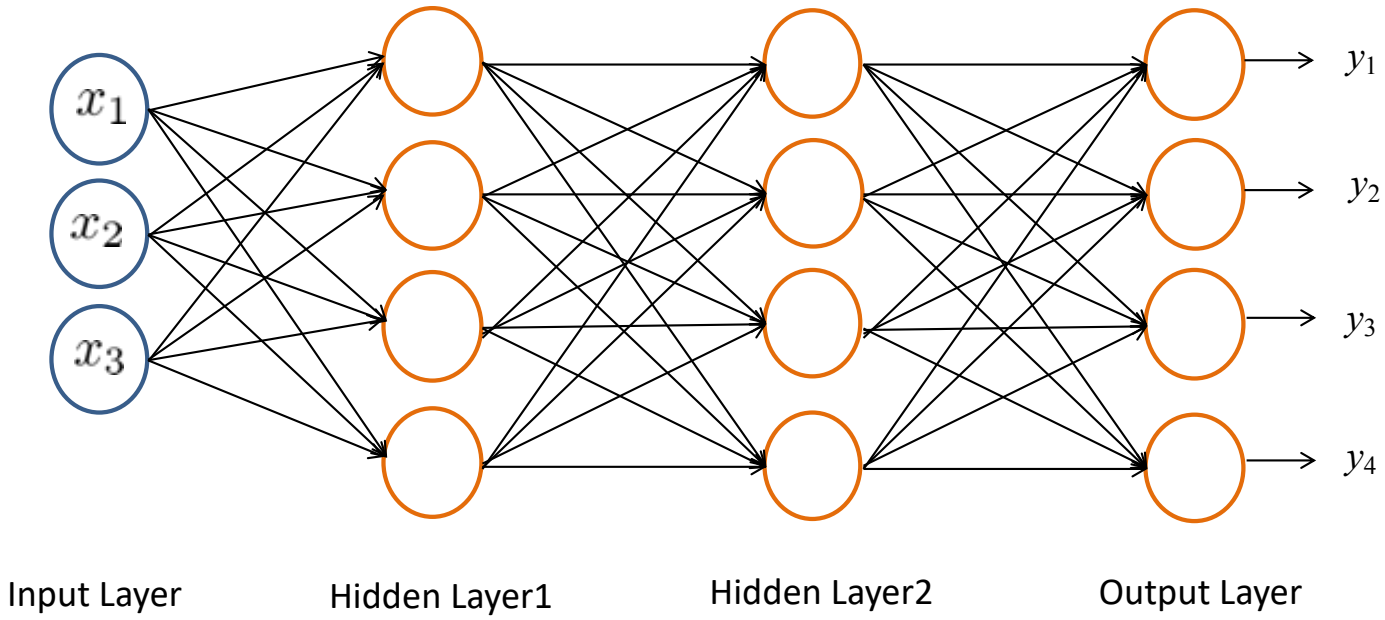
Training for ANN

- **In Training Stage, an ANN tries to find the best wiring structure among neurons. Particularly, it tries to find the best connection WEIGHTS based on the training dataset.**

- To do that, we first need to define a ***cost function (error function)*** that represents the ERROR! Then, try to **minimize the cost function** on the Training Dataset, Solve the minimization problem in terms of **Weights** to find the best weights.

Neural Networks

- The first layer is called **Input Layer**. The input layer only passes and distribute the inputs and perform **no computation**.
- The last layer is called **Output Layer**.
- The middle layers are called **Hidden Layers**.



The Cost Function (Error Function)

The cost function for Neural network:

$$J(W) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log(h_W(x^{(i)}))_k + (1- y_k^{(i)}) \log(1 - (h_W(x^{(i)}))_k)\right]$$

$$+\frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}} (w_{ij}^{(l)})^2$$

$$\min_W J(W)$$

We can use Gradient Descent Algorithm to minimize the Cost Function in terms of "weights", by calculating the partial derivatives: $\frac{\partial}{\partial w_{ij}^{(l)}} J(W)$

BackPropagation:

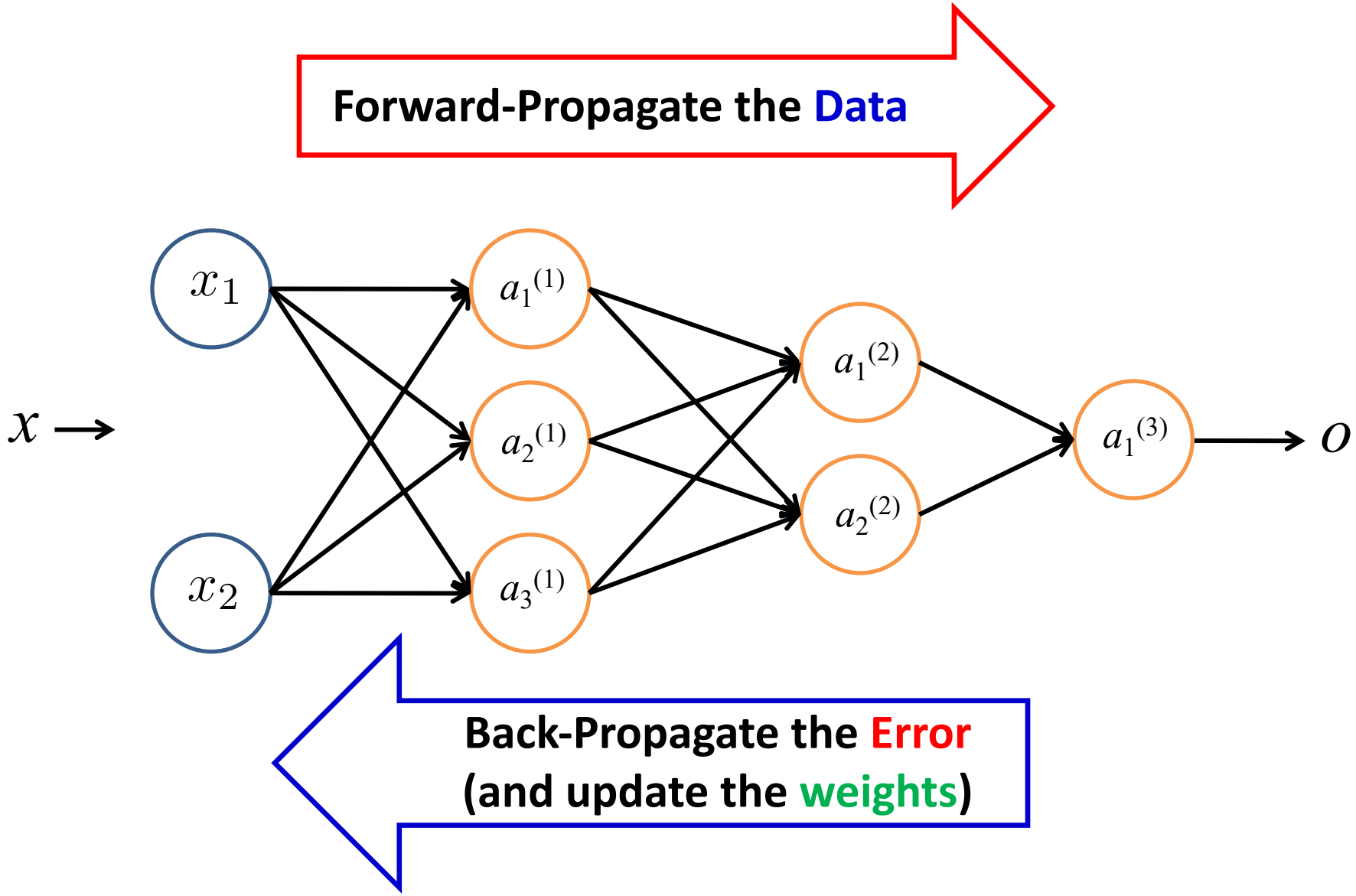
A Systematic Method to Train ANNs

Backpropagation

- **The backward propagation** or **backpropagation**, is a popular method for training artificial neural networks, and is usually used in conjunction with a minimization method such as gradient descent.
- The **backpropagation** is an iterative algorithm that includes two main phases:

I. A training sample *"x"* is presented to the network input layer. The input sample is **propagated forward** through the network, layer by layer, until it reaches the output layer, and an output vector *"o"* is generated at the output layer.

II. The **generated output** *"o"* is compared to the **actual (desired) output** *"y"* in the training set (i.e. the desired output corresponding to the input x), and an error is calculated based on the defined cost function. The error is then **propagated backwards** through the network from the output layer to the input layer. In this process, an associated gradient error is calculated for each unit, and used to update the network weights.


Forward-Propagate the Data
x
x_1
x_2
$a_1^{(1)}$
$a_2^{(1)}$
$a_3^{(1)}$
$a_1^{(2)}$
$a_2^{(2)}$
$a_1^{(3)}$
o
Back-Propagate the Error
(and update the weights)

Backpropagation Algorithm

1. **Randomly initialize** the network weights.
2. Then, apply one training sample as an input to the network, and **propagate forward** until we find the output at output layer.
3. Calculate **error** at the output layer based on a cost function between the **generated output** "o" and the **actual output** "y" from the training set.
   For the sake of simplicity in backpropagation, we just assume that the cost is $J = \frac{1}{2}(y - o)^2$ for now. Then, calculate the **Gradient Error at output layer** as:

$$\delta = \frac{\partial J}{\partial z} = (y - o)\, o\, (1 - o)$$

The term "$o\,(1 - o)$" is necessary in the equation because of the derivative of the Sigmoid Function.

[Ref]: "An Introduction to Practical Neural Networks and Genetic Algorithms For Engineers and Scientists" by Christopher MacLeod.

Backpropagation Algorithm

4. Update the **weights** **at output layer**:

$$w_{ij}^{(l)}(\text{new}) = w_{ij}^{(l)}(\text{old}) + \alpha\ \delta\ a_i^{(l-1)}$$

(where α is the ***learning rate***, higher α means faster learning process)

5. Calculate the **Gradient Error** for the previous **hidden layer neurons**. To do that, we **Back-Propagate** the gradient errors staring from the output layer through the weights to get the hidden layer errors. In fact, Backpropagation calculates the following equation:

$$\delta_i^{(l-1)} = [\ \sum w_{ji}^{(l)} \delta_j^{(l)}\ ]\ a_i^{(l-1)}\ (1 - a_i^{(l-1)})$$

Note: Don't forget the term "$a_i^{(l-1)}\ (1 - a_i^{(l-1)})$" when you back propagate!

Backpropagation Algorithm

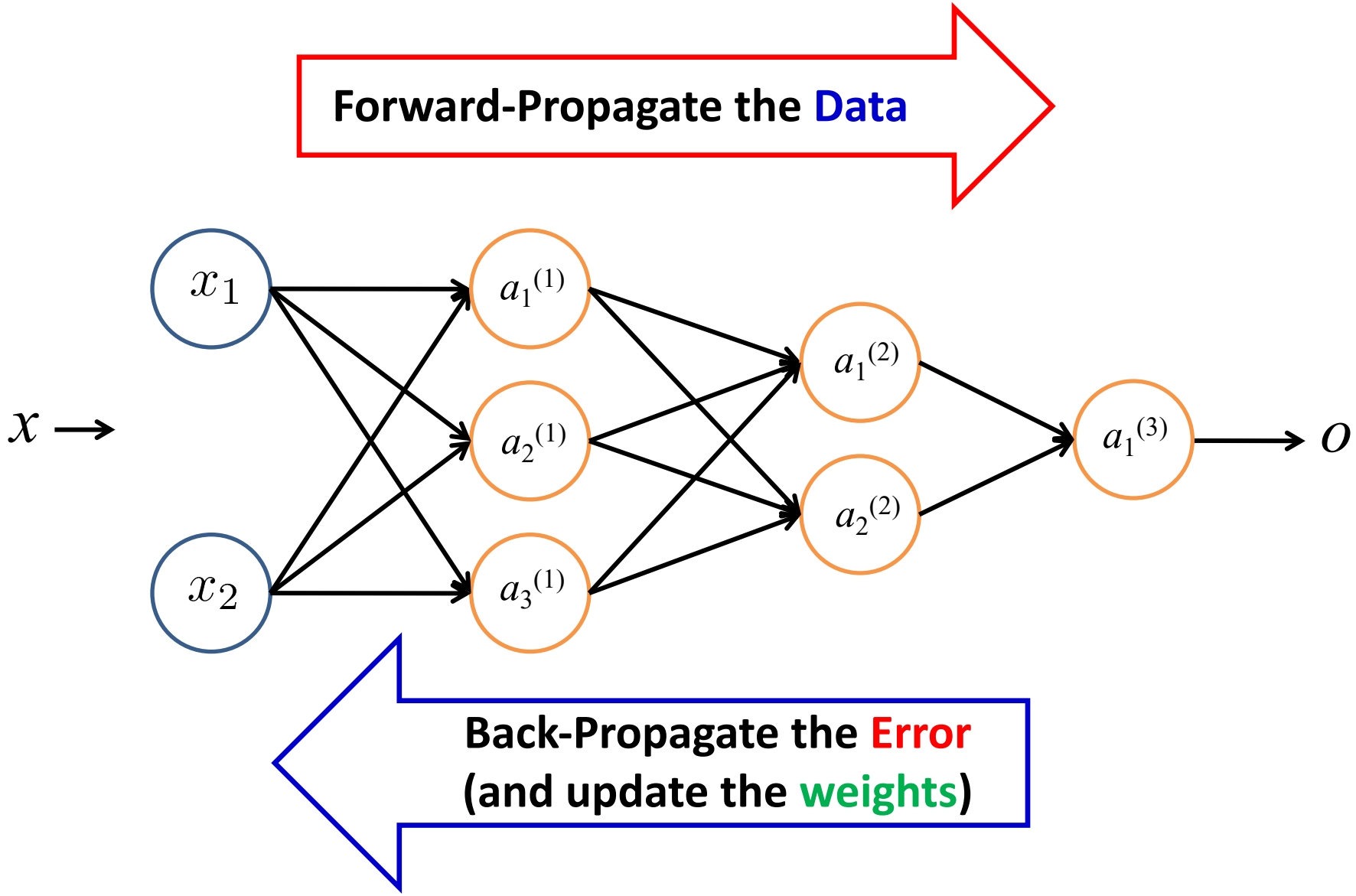
6. Similarly, Having obtained the Gradient Errors for the hidden layer neurons, now proceed to **update the hidden layer weights**:

$$w_{ij}^{(l)}(\text{new}) = w_{ij}^{(l)}(\text{old}) + \alpha\ \delta_i^{(l)}\ a_i^{(l-1)}$$

Note: when you reach the input layer, notice that $a_i^{(0)} = x_i$

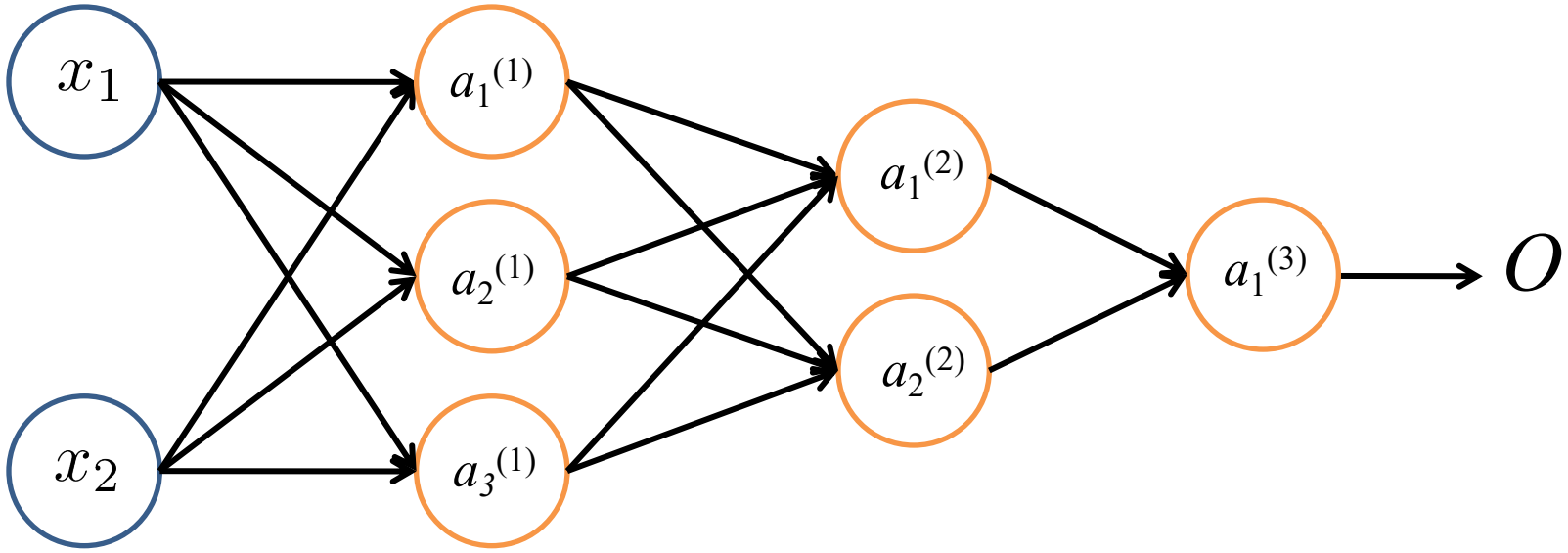
7. Jump to Step 2, and repeat using next training data sample.

Note: All of the "***weight updates***" will be in effect at the end of iteration, and will be used for next data sample.


Forward-Propagate the Data
x
x_1
x_2
$a_1^{(1)}$
$a_2^{(1)}$
$a_3^{(1)}$
$a_1^{(2)}$
$a_2^{(2)}$
$a_1^{(3)}$
o
Back-Propagate the Error
(and update the weights)

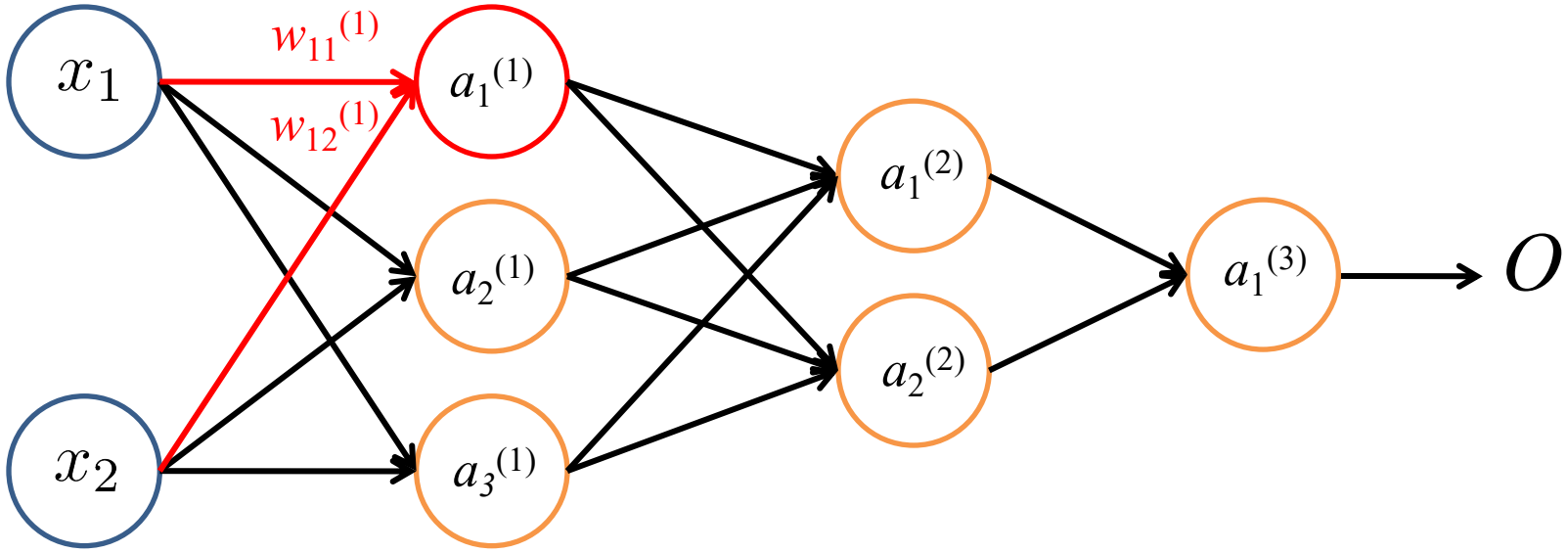
Example for Backpropagation

Step1: Moving Forward in the network to find the output based on a sample input and random initial weights.



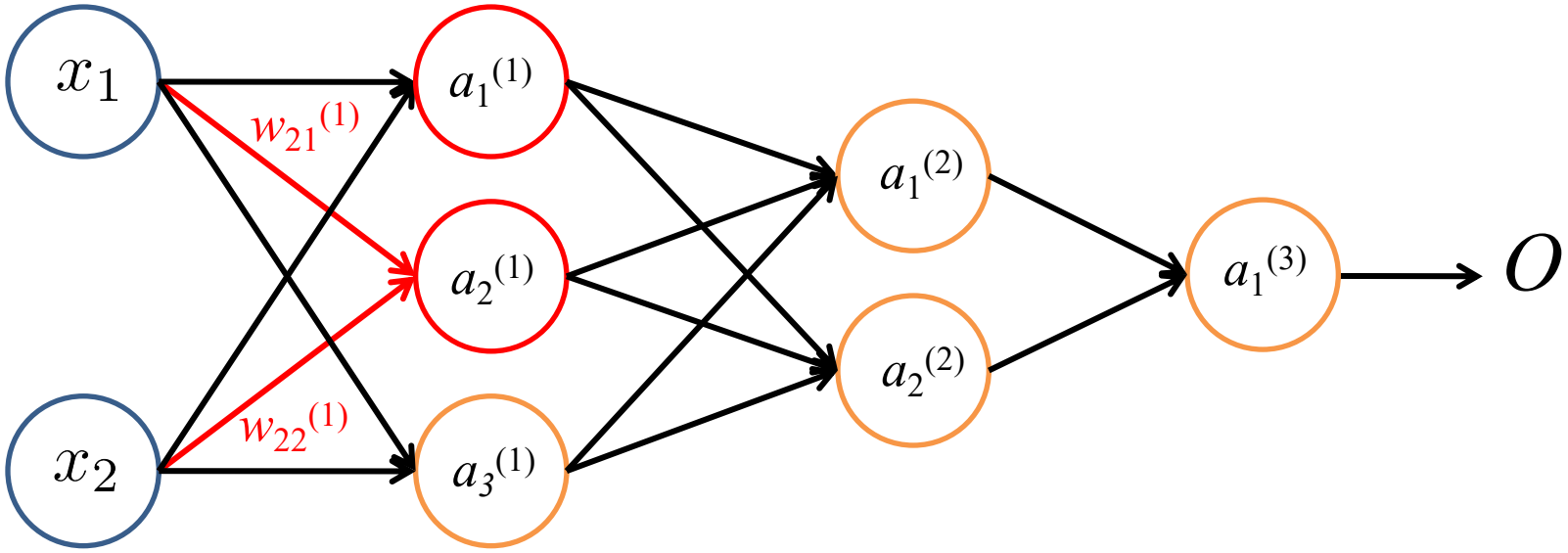
Example for Backpropagation

Notice that in this example, for the sake of simplicity, we assume that we don't have any Bias. However, in general we have bias term as well. In that case, We can update the bias weights exactly similar to other weights.



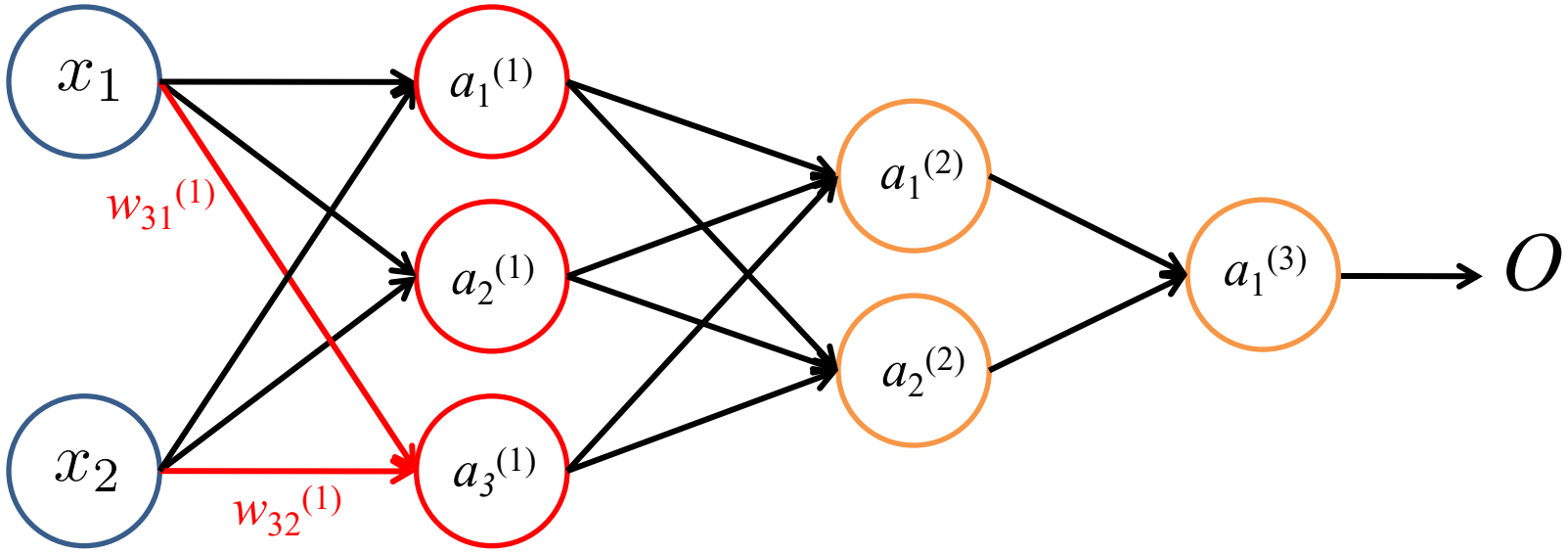
$$a_1^{(1)} = g\left(w_{11}^{(1)} x_1 + w_{12}^{(1)} x_2\right)$$

Example for Backpropagation



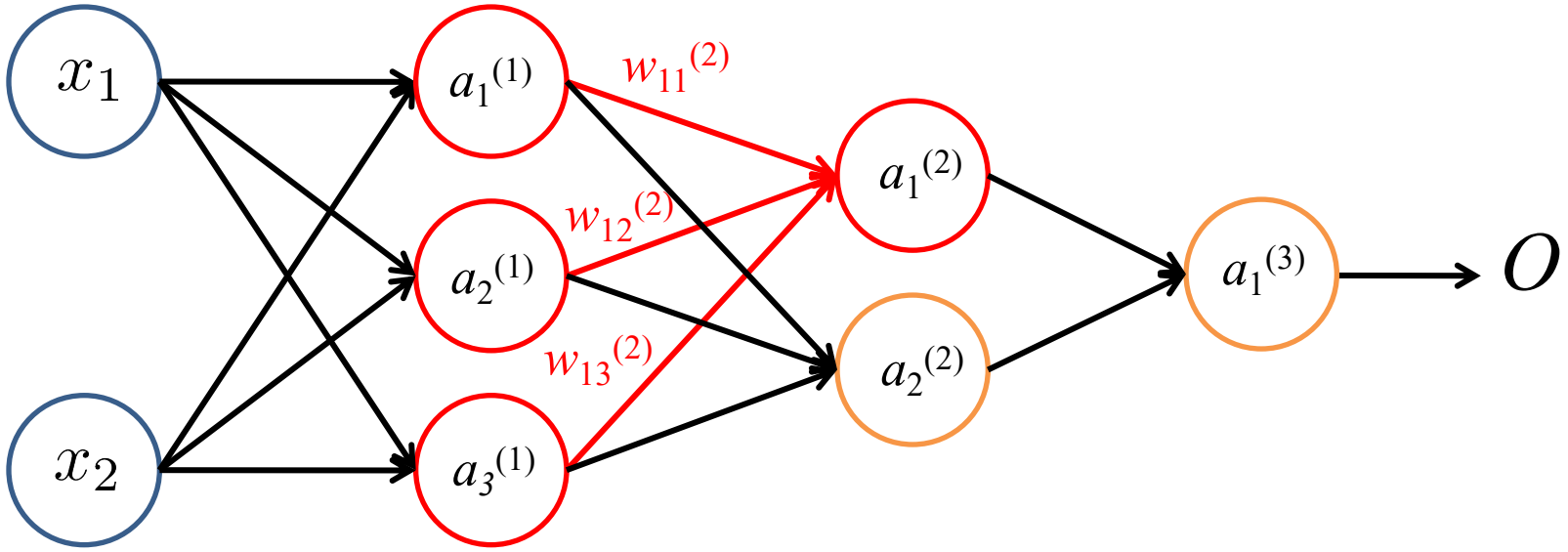
$$a_2^{(1)} = g\left(w_{21}^{(1)} x_1 + w_{22}^{(1)} x_2\right)$$

Example for Backpropagation



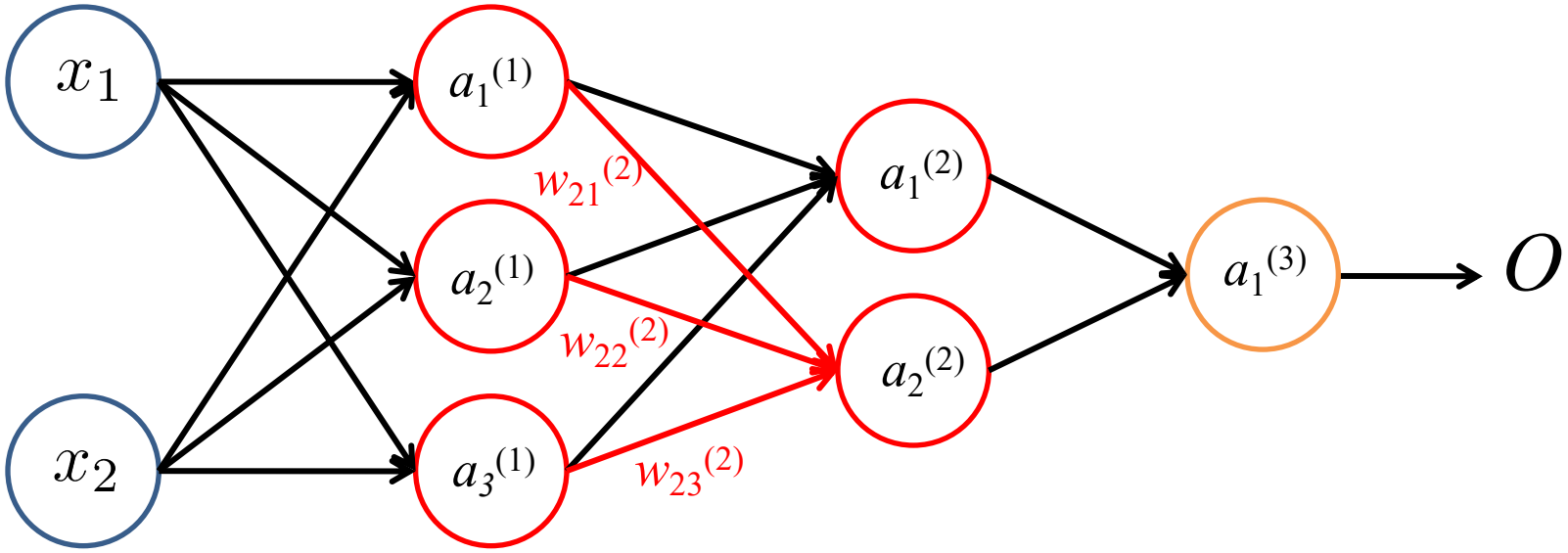
$$a_3^{(1)} = g\left(w_{31}^{(1)} x_1 + w_{32}^{(1)} x_2\right)$$

Example for Backpropagation



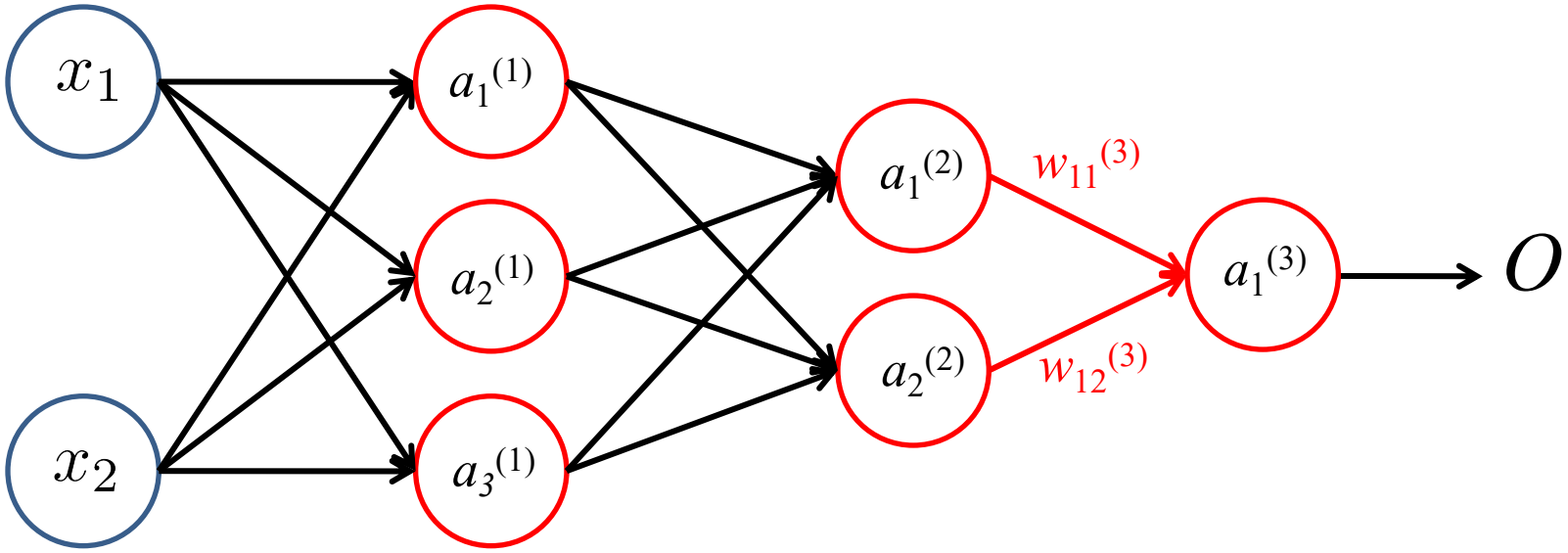
$$a_1^{(2)} = g\left(w_{11}^{(2)} a_1^{(1)} + w_{12}^{(2)} a_2^{(1)} + w_{13}^{(2)} a_3^{(1)}\right)$$

Example for Backpropagation



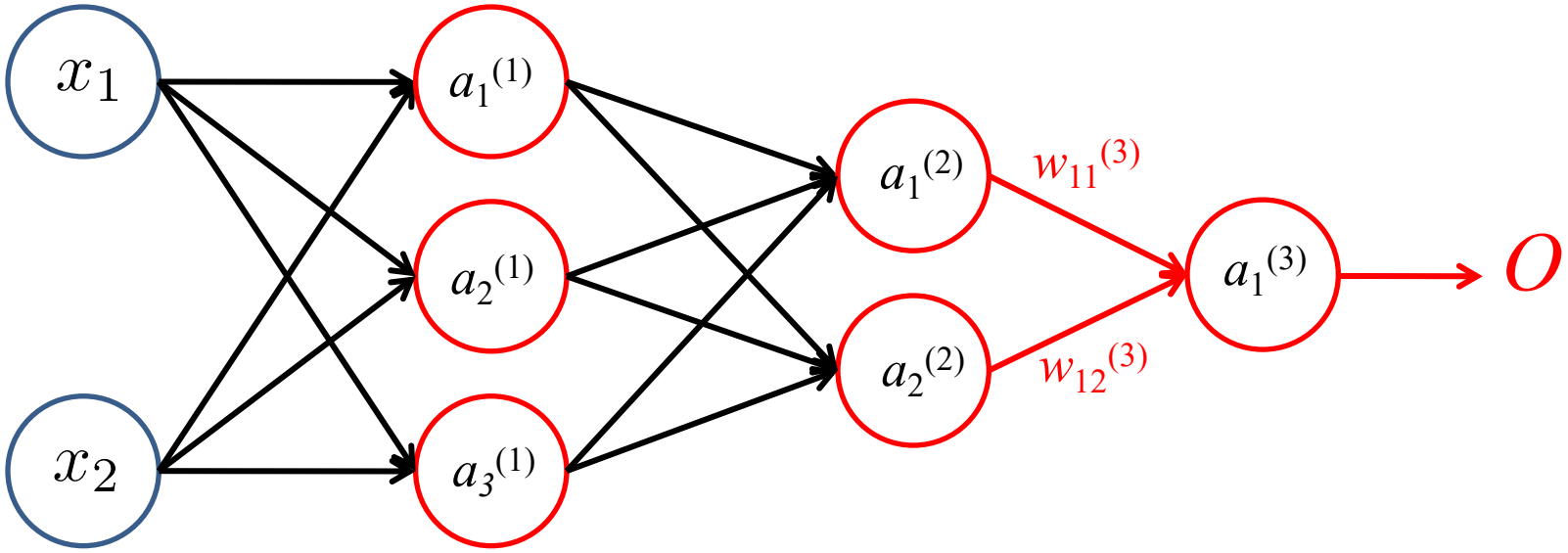
$$a_2^{(2)} = g\left(w_{21}^{(2)} a_1^{(1)} + w_{22}^{(2)} a_2^{(1)} + w_{23}^{(2)} a_3^{(1)}\right)$$

Example for Backpropagation



$$a_1^{(3)} = g\left(w_{11}^{(3)} a_1^{(2)} + w_{12}^{(3)} a_2^{(2)}\right)$$

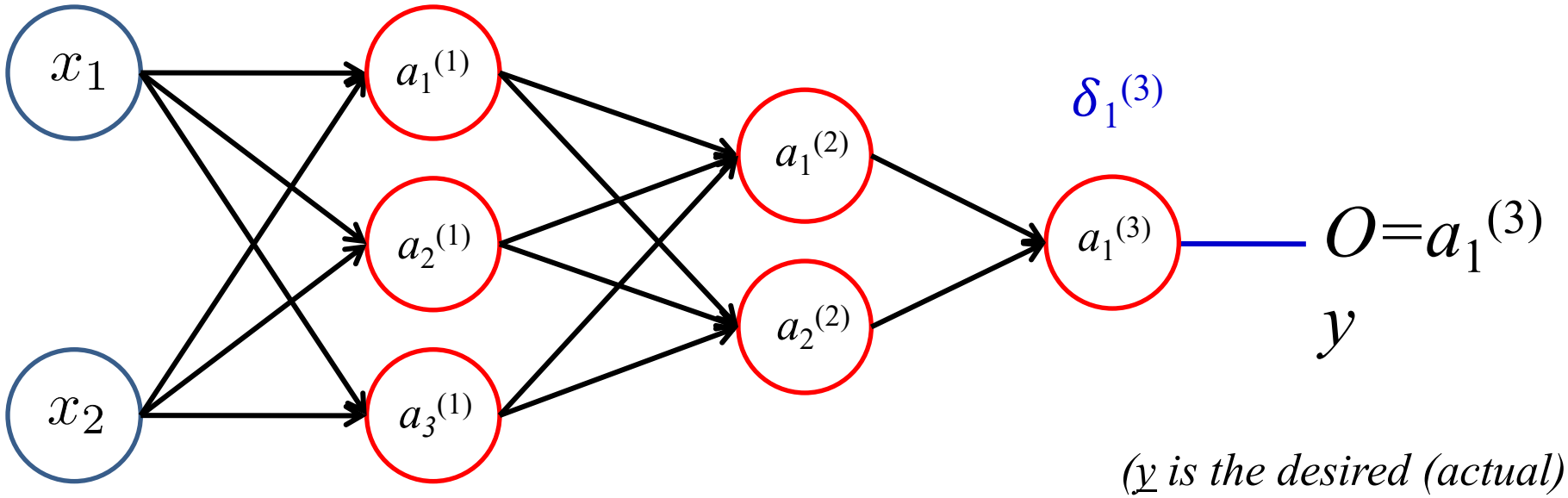
Example for Backpropagation



$$O = a_1^{(3)} = g\left(w_{11}^{(3)} a_1^{(2)} + w_{12}^{(3)} a_2^{(2)}\right)$$

Example for Backpropagation

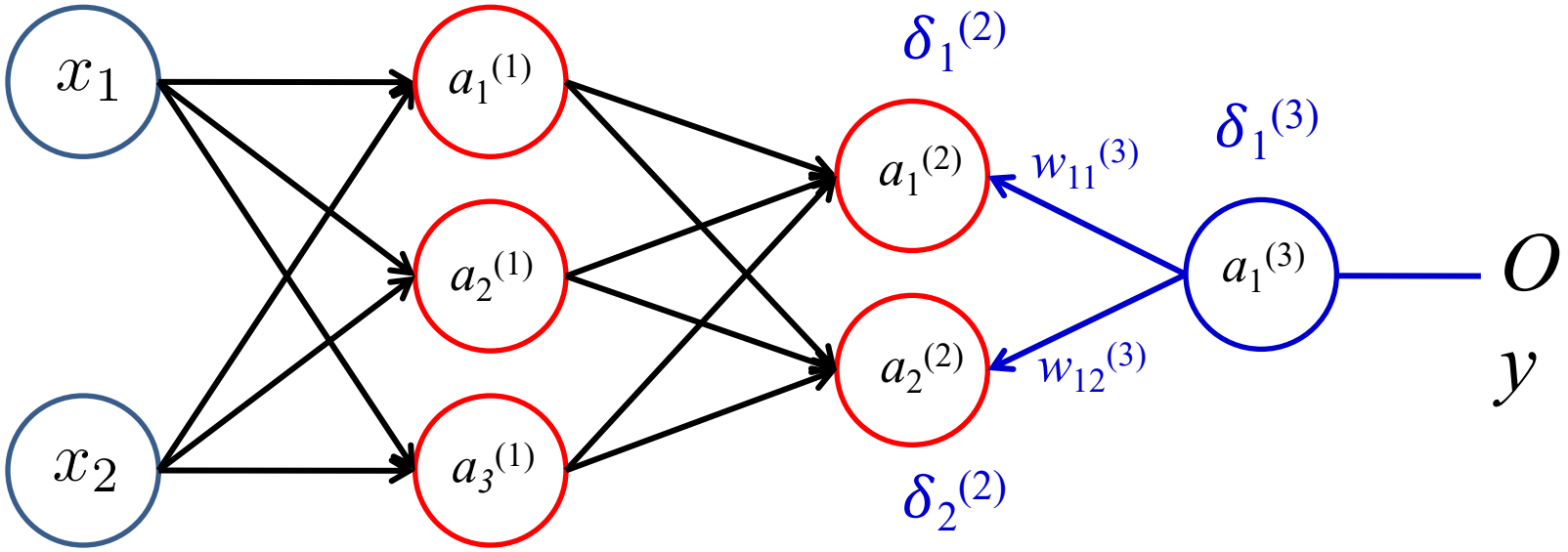
Step2: Find the output Error Gradient δ at output later, and then Move Backward in the network to propagate back the Error, find the Error Gradient of each node in hidden layers, and eventually update the weights.



$$\delta_1^{(3)} = (y - a_1^{(3)})\, a_1^{(3)}\, (1 - a_1^{(3)})$$

($\underline{y}$ is the desired (actual) output, $o=a_1^{(3)}$ is the predicted output by the NN)

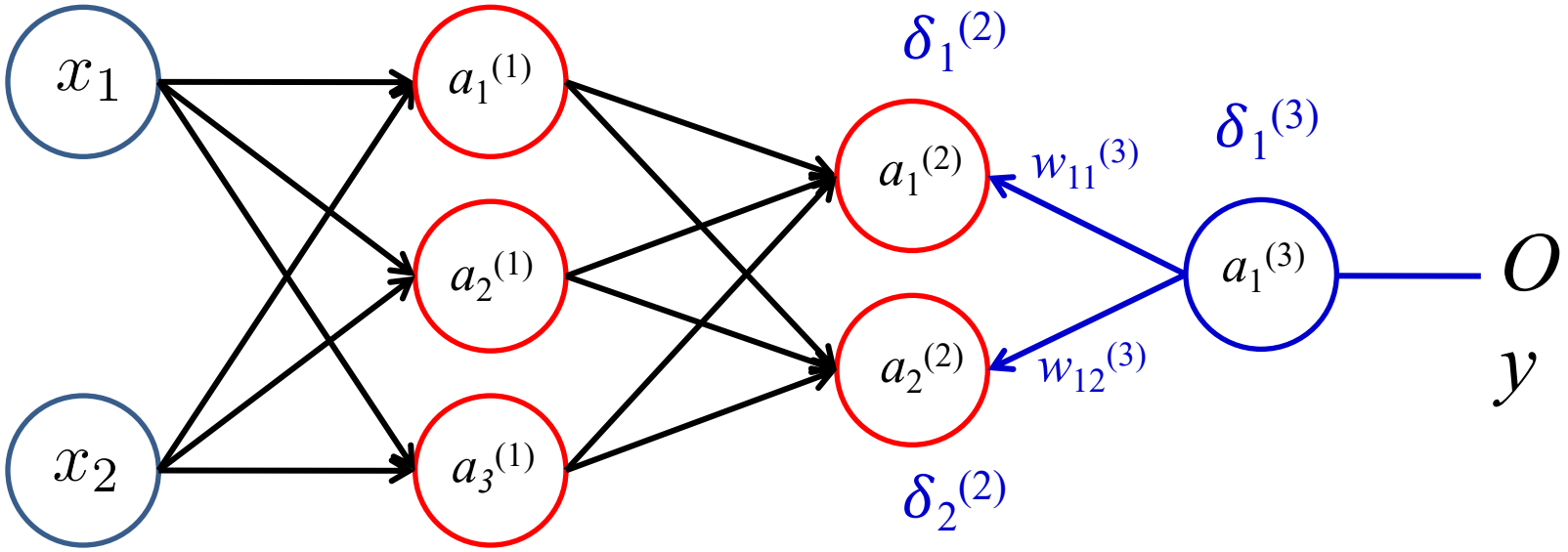
Example for Backpropagation



$$\delta_1^{(2)} = [\, w_{11}^{(3)}\, \delta_1^{(3)} \,]\, a_1^{(2)}\,(1 - a_1^{(2)})$$

$$\delta_2^{(2)} = [\, w_{12}^{(3)}\, \delta_1^{(3)} \,]\, a_2^{(2)}\,(1 - a_2^{(2)})$$

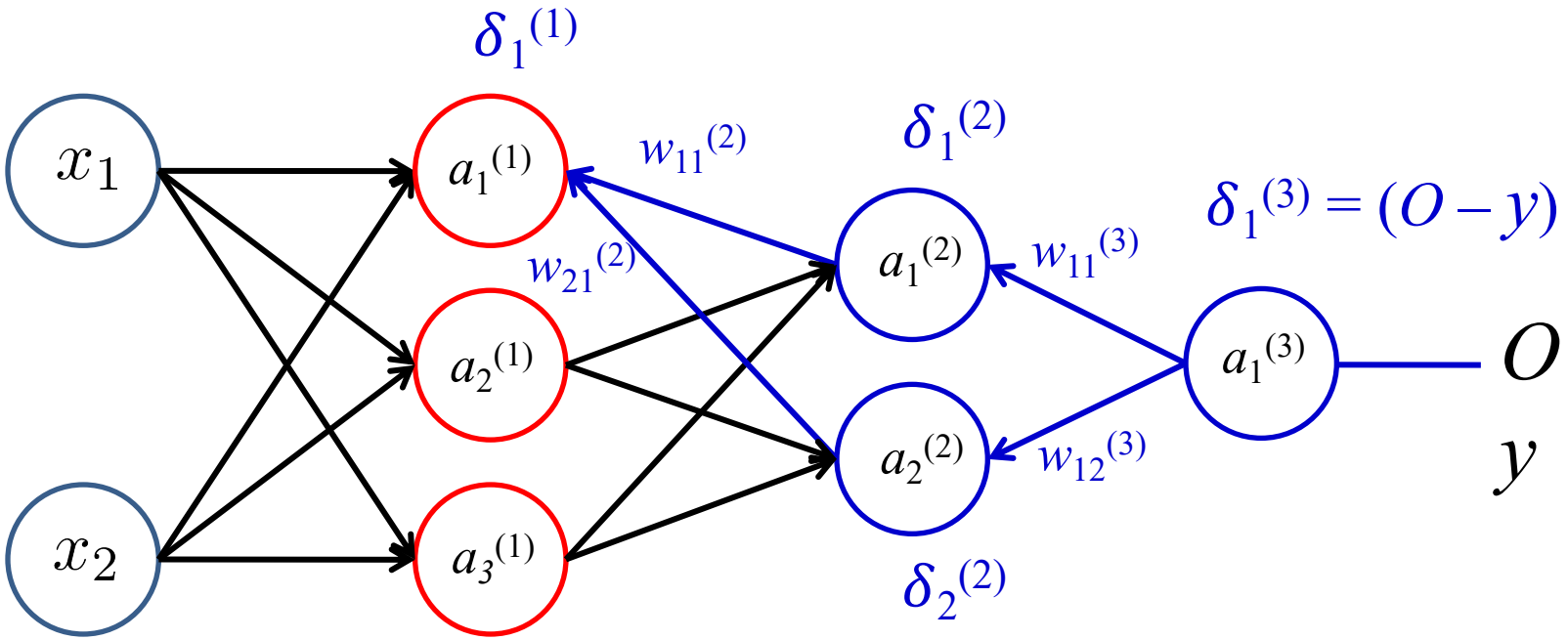
Example for Backpropagation



$$w_{11}^{(3)}(\text{new}) = w_{11}^{(3)} + \alpha\ \delta_1^{(3)}\ a_1^{(2)}$$

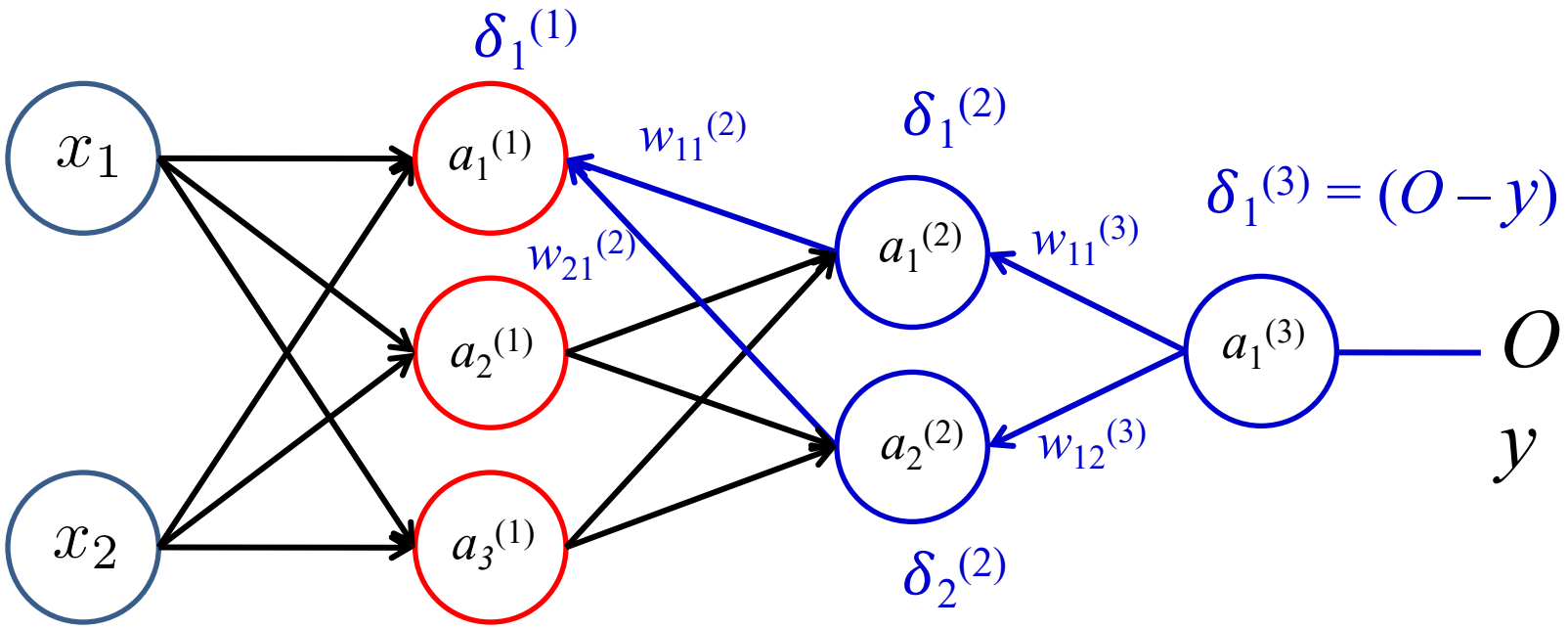
$$w_{12}^{(3)}(\text{new}) = w_{12}^{(3)} + \alpha\ \delta_1^{(3)}\ a_2^{(2)}$$

Example for Backpropagation



$$\delta_1^{(1)} = [\, w_{11}^{(2)}\, \delta_1^{(2)} + w_{21}^{(2)}\, \delta_2^{(2)} \,]\, a_1^{(1)}\, (1 - a_1^{(1)})$$

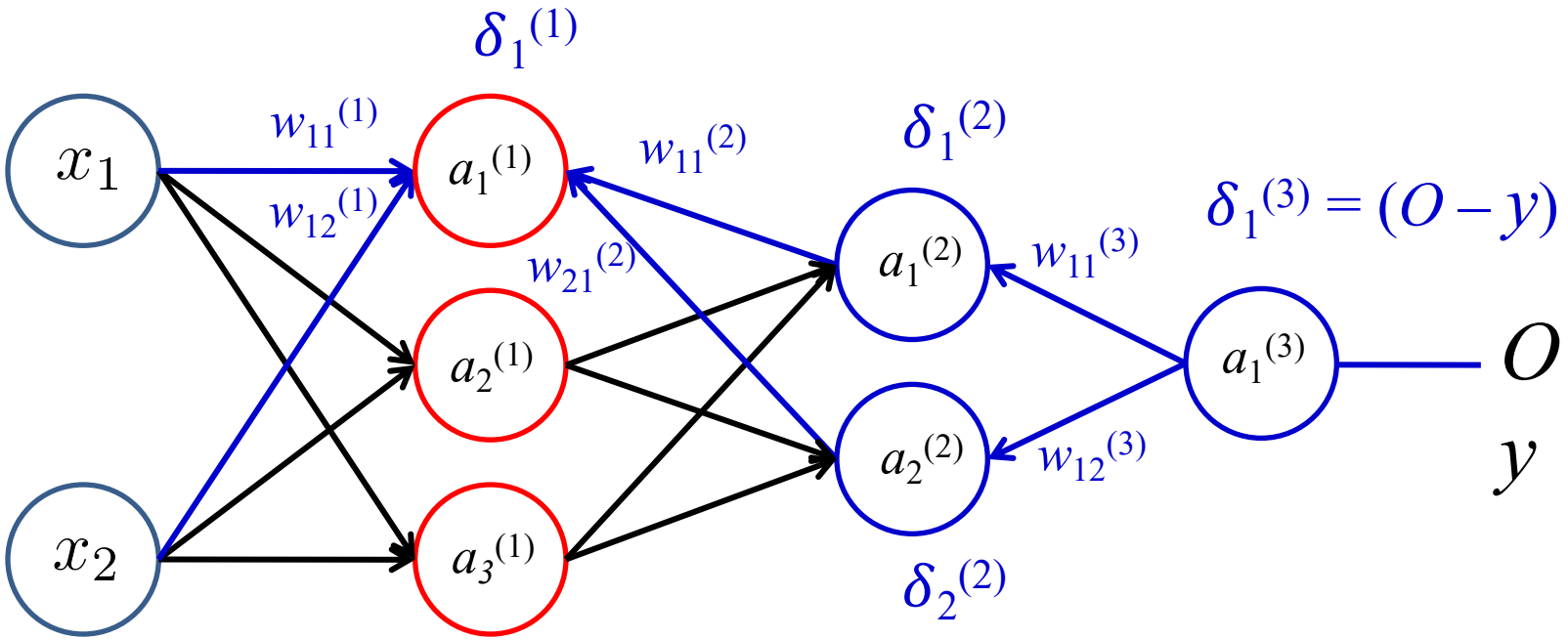
Example for Backpropagation



$$w_{11}^{(2)}(\text{new}) = w_{11}^{(2)} + \alpha\ \delta_1^{(2)}\ a_1^{(1)}$$

$$w_{21}^{(2)}(\text{new}) = w_{21}^{(2)} + \alpha\ \delta_2^{(2)}\ a_1^{(1)}$$

Example for Backpropagation

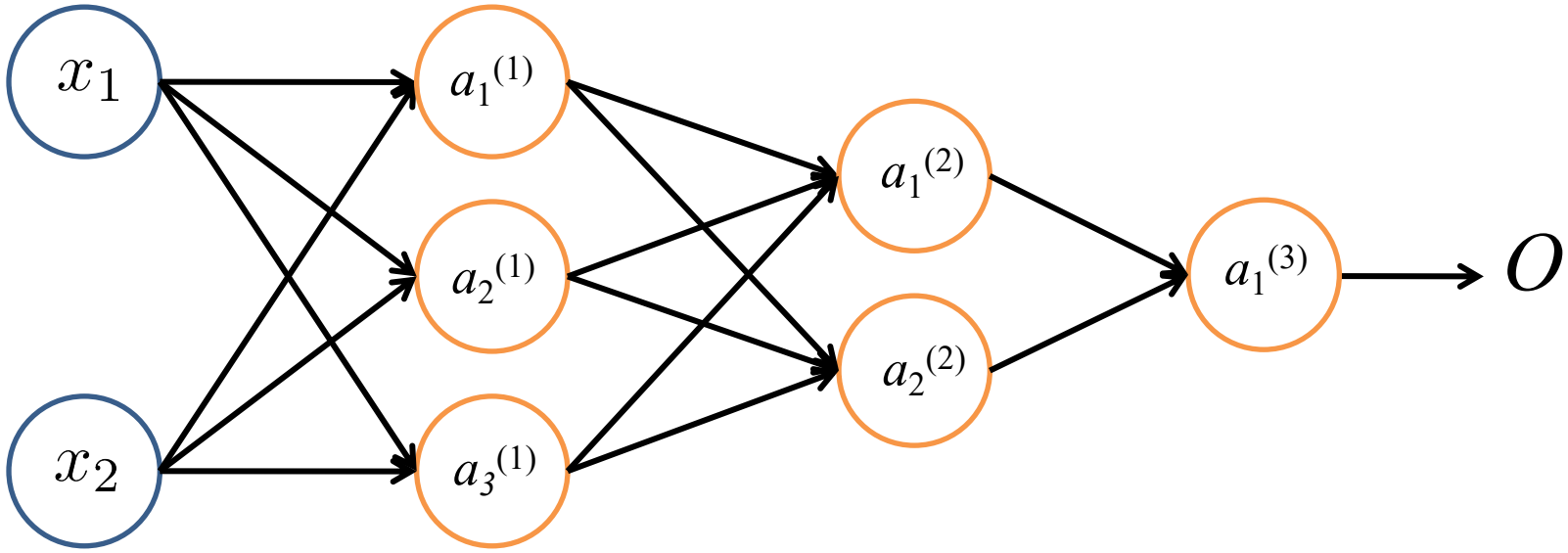


$$w_{11}^{(1)}(\text{new}) = w_{11}^{(1)} + \alpha\, \delta_1^{(1)}\, x_1$$

$$w_{12}^{(1)}(\text{new}) = w_{12}^{(1)} + \alpha\, \delta_1^{(1)}\, x_2$$

Example for Backpropagation

- Repeat the process for next training sample.
- Repeat the Iteration until we achieve the desired accuracy or reach the iteration limit!



Facts!

- **Important Questions:**

  Backpropagation makes sense intuitively! But, we still need to minimize the cost function. How does it minimize the Cost Function? Why does Backpropagation work in theory?

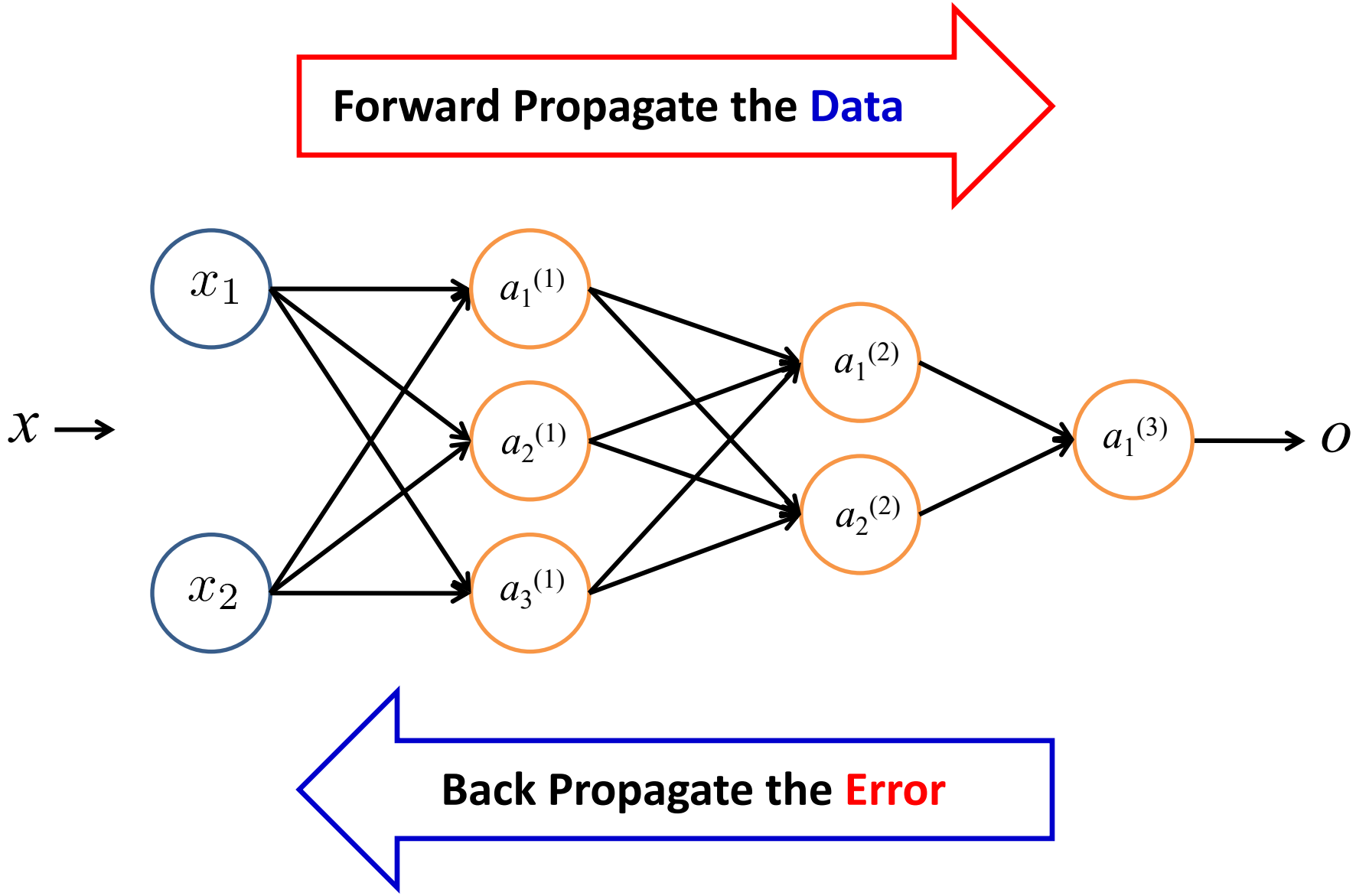
- **Answer:**

  Backpropagation is actually the implementation of Chain Rule when we take derivative to minimize the Cost Function!!!

$$[\, f(g(x)) \,]' = f'(g(x))\, g'(x)$$

$$[\, f(g(h(x))) \,]' = f'(g(h(x)))\, g'(h(x))\, h'(x)$$

BackPropagation
(Continue)

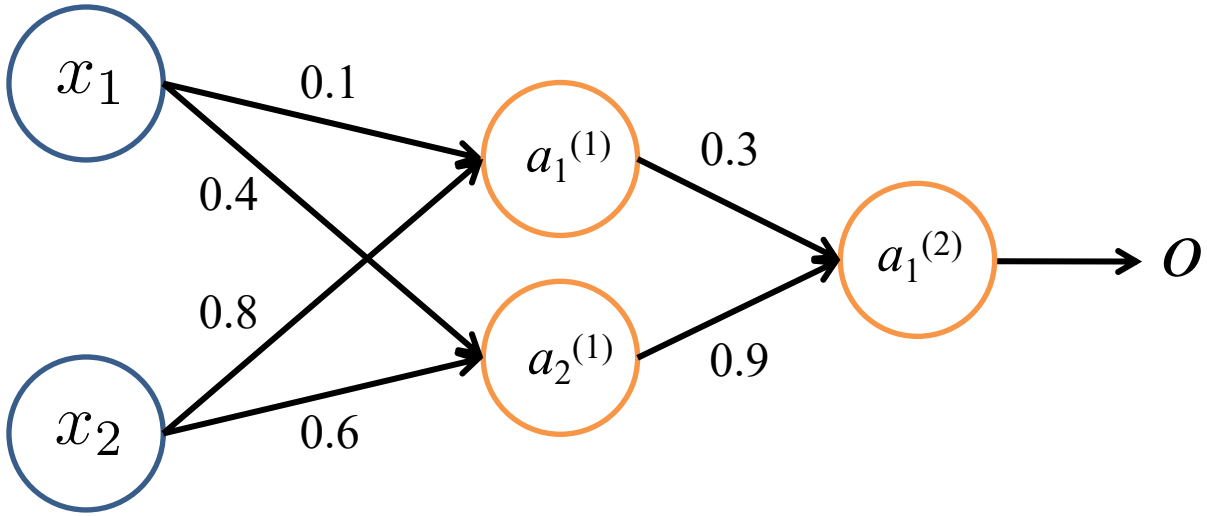

Forward Propagate the Data
x
x_1
x_2
$a_1^{(1)}$
$a_2^{(1)}$
$a_3^{(1)}$
$a_1^{(2)}$
$a_2^{(2)}$
$a_1^{(3)}$
o
Back Propagate the Error

Summary: Backpropagation in one Slide!

1. **Randomly initialize** the network weights.
2. Apply one training sample as input, and **propagate forward** until you find the output "o".
3. Calculate the **Gradient Error at output layer** as: $\delta = (y - o)\, o\, (1 - o)$
4. Find **Gradient Error** $\delta_i^{(l)}$ for **hidden layer neurons** using **BackPropagation.** Don't forget the term "$a_i^{(l)}\,(1 - a_i^{(l)})$" when you back propagate!
5. Update the **weights at output layer**:
   (α is learning rate)
   $w_{ij}^{(l)}(\text{new}) = w_{ij}^{(l)}(\text{old}) + \alpha\ \delta\ a_i^{(l\text{-}1)}$
6. Update the **weights at hidden layers**:
   Note that $a_i^{(0)} = x_i$.
   $w_{ij}^{(l)}(\text{new}) = w_{ij}^{(l)}(\text{old}) + \alpha\ \delta_i^{(l)}\ a_i^{(l\text{-}1)}$
7. Jump to Step 2, and repeat using next data sample.

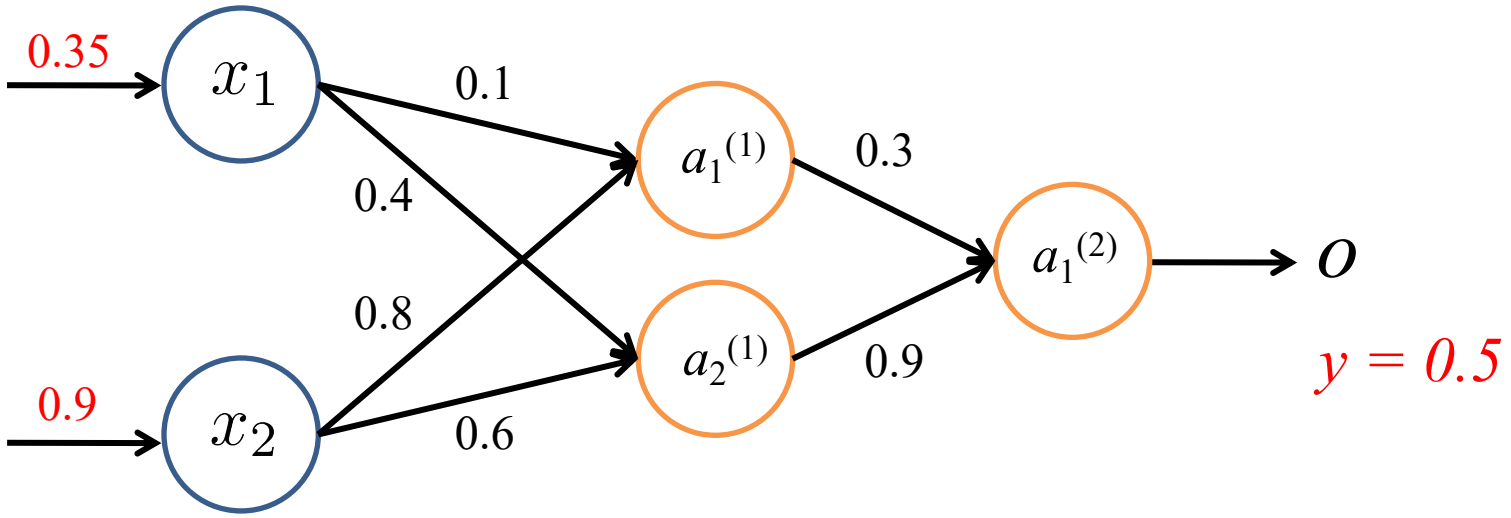
Backpropagation: A Numerical Example

In the following neural network, we have initialized the weights randomly. Use a training sample $(\boldsymbol{x},\boldsymbol{y}) = ((0.35,0.9), 0.5)$ to update the weights. Use α =1 as the learning rate. For the sake of simplicity, assume that there is no bias term involved in the network.



Ref: Example: "An Introduction to Practical Neural Networks and Genetic Algorithms For Engineers and Scientists", Christopher MacLeod.

A Numerical Example

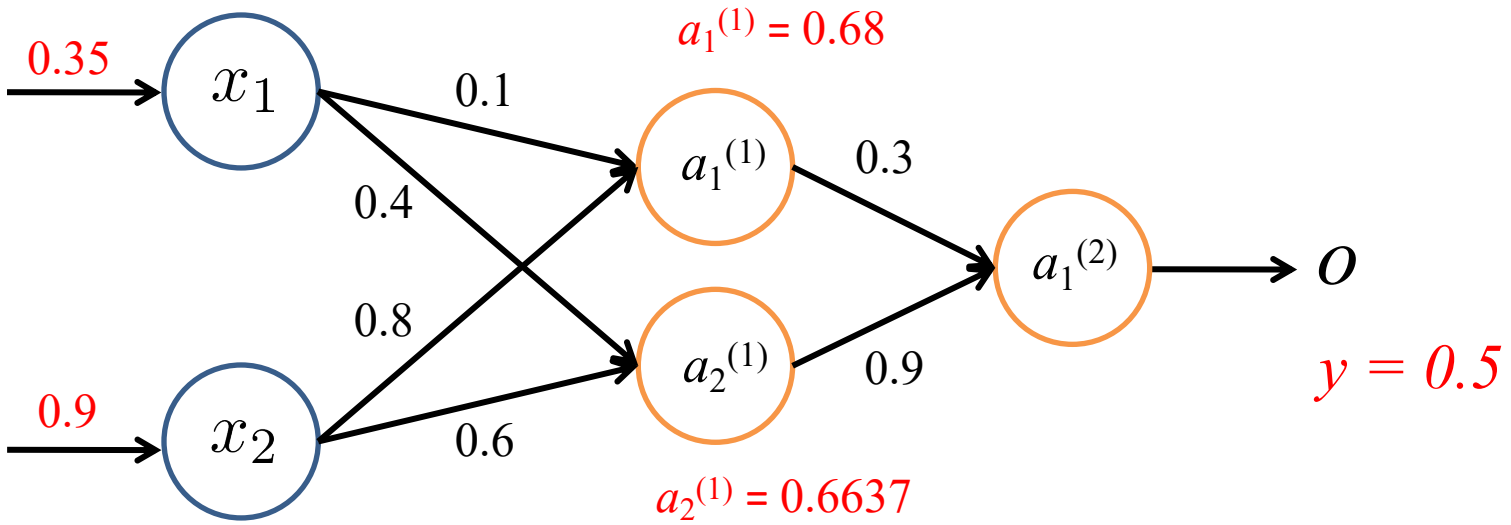


$$a_1^{(1)} = g\,(w_{11}^{(1)} x_1 + w_{12}^{(1)} x_2) = g(0.1*0.35+0.8*0.9) = 0.68$$

$$a_2^{(1)} = g\,(w_{21}^{(1)} x_1 + w_{22}^{(1)} x_2) = g(0.6*0.9+0.4*0.35) = 0.6637$$

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

A Numerical Example

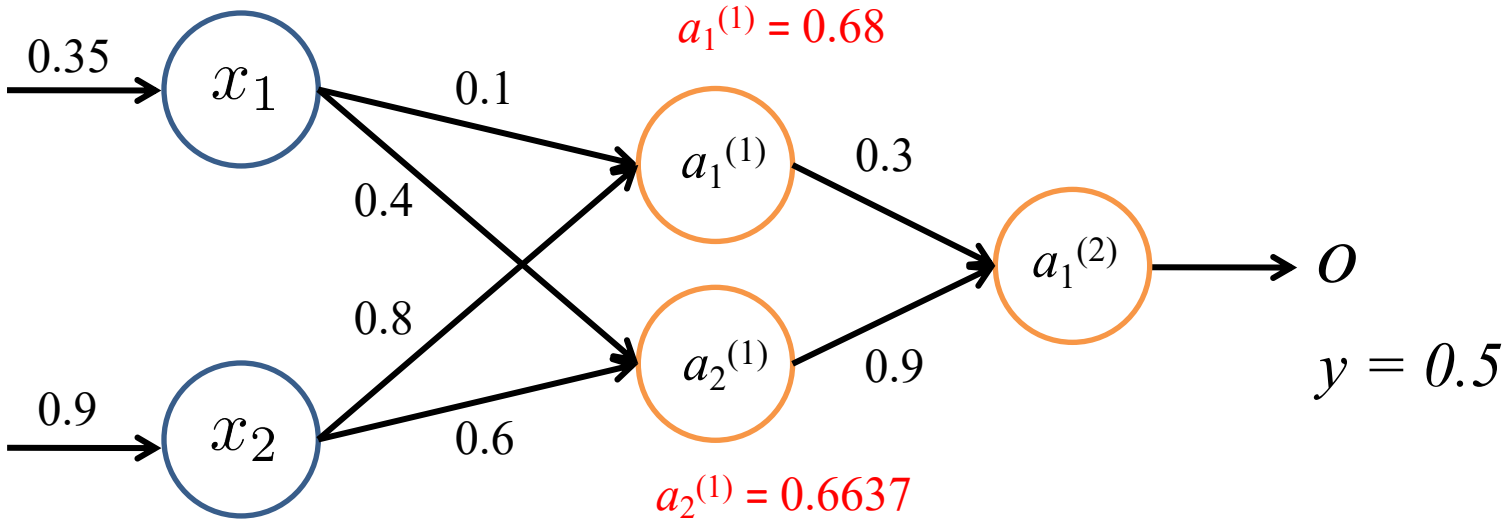


$$a_1^{(1)} = g\,(w_{11}^{(1)} x_1 + w_{12}^{(1)} x_2) = \text{g}(0.1*0.35+0.8*0.9) = 0.68$$

$$a_2^{(1)} = g\,(w_{21}^{(1)} x_1 + w_{22}^{(1)} x_2) = \text{g}(0.6*0.9+0.4*0.35) = 0.6637$$

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

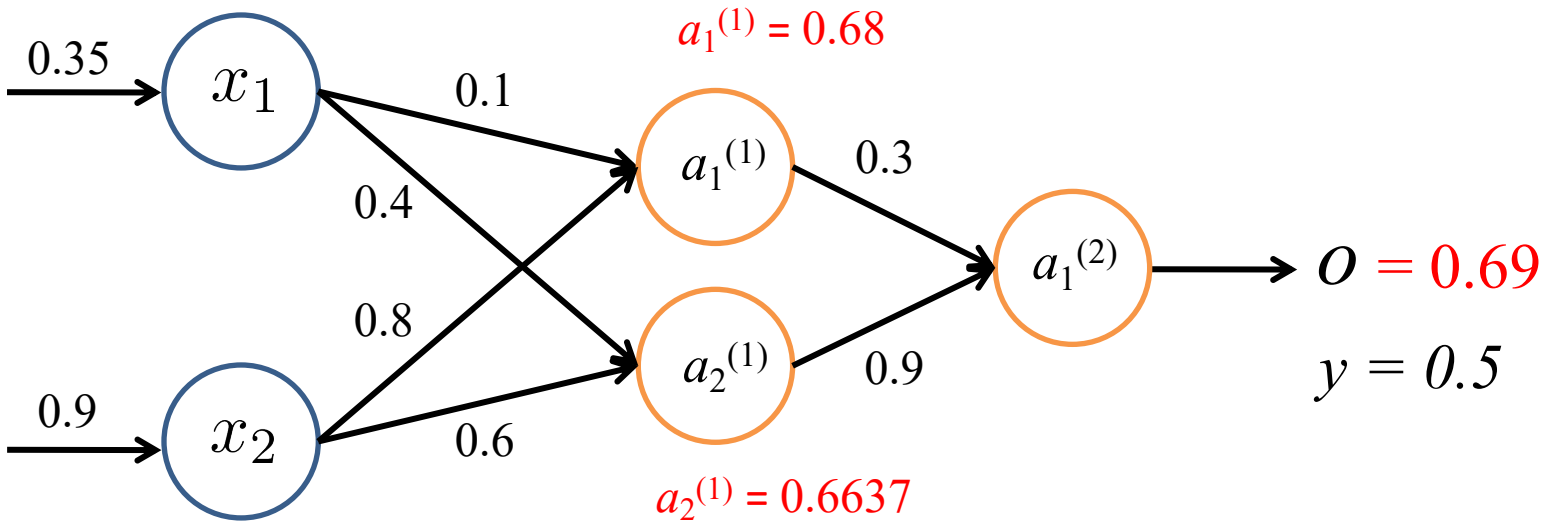
A Numerical Example



$$O = a_1^{(2)} = g\left(w_{11}^{(2)} a_1^{(1)} + w_{12}^{(2)} a_2^{(1)}\right)$$

$$= g\left(0.3*0.68 + 0.9*0.6637\right) = 0.69$$

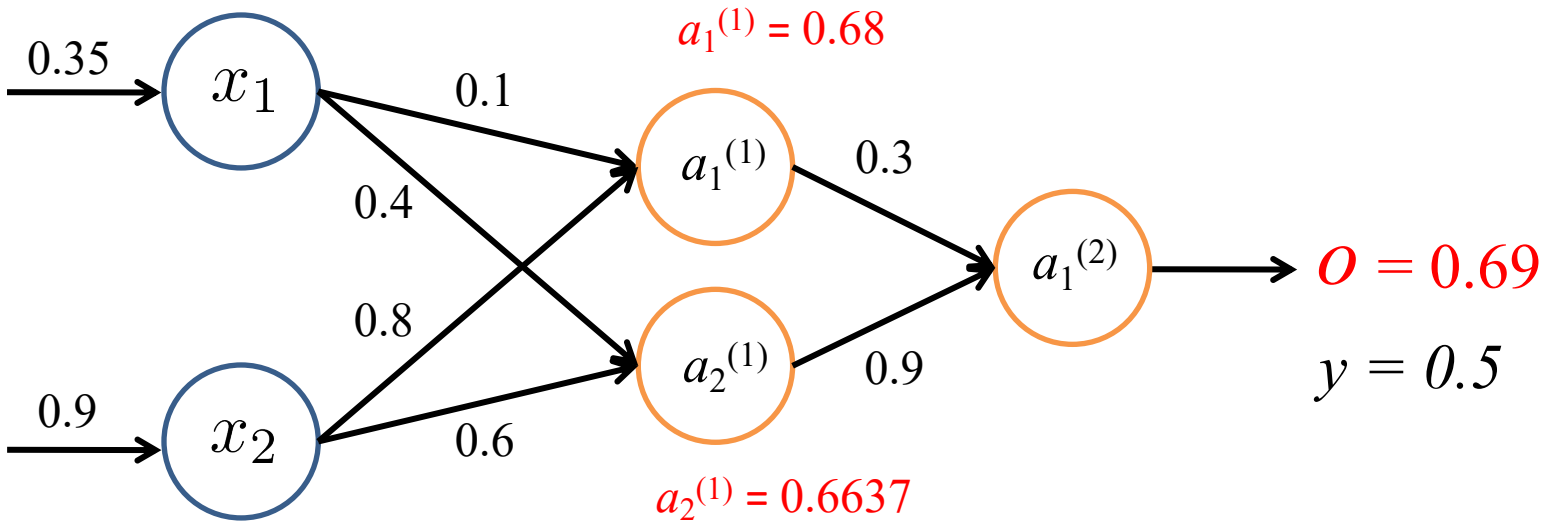
A Numerical Example



$$O = a_1^{(2)} = g\left(w_{11}^{(2)} a_1^{(1)} + w_{12}^{(2)} a_2^{(1)}\right)$$

$$= g\left(0.3*0.68 + 0.9*0.6637\right) = 0.69$$

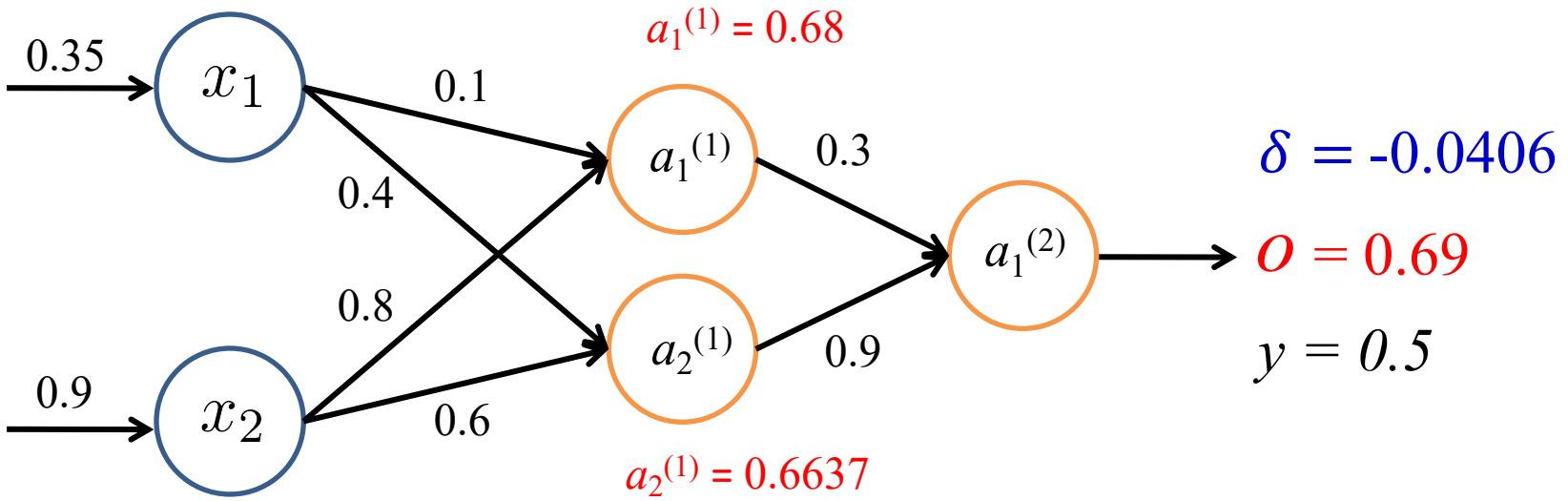
A Numerical Example



$$\delta = (y - o)\, o\, (1 - o)$$

$$\delta = (0.5 - 0.69) * 0.69 * (1 - 0.69) = -0.0406$$

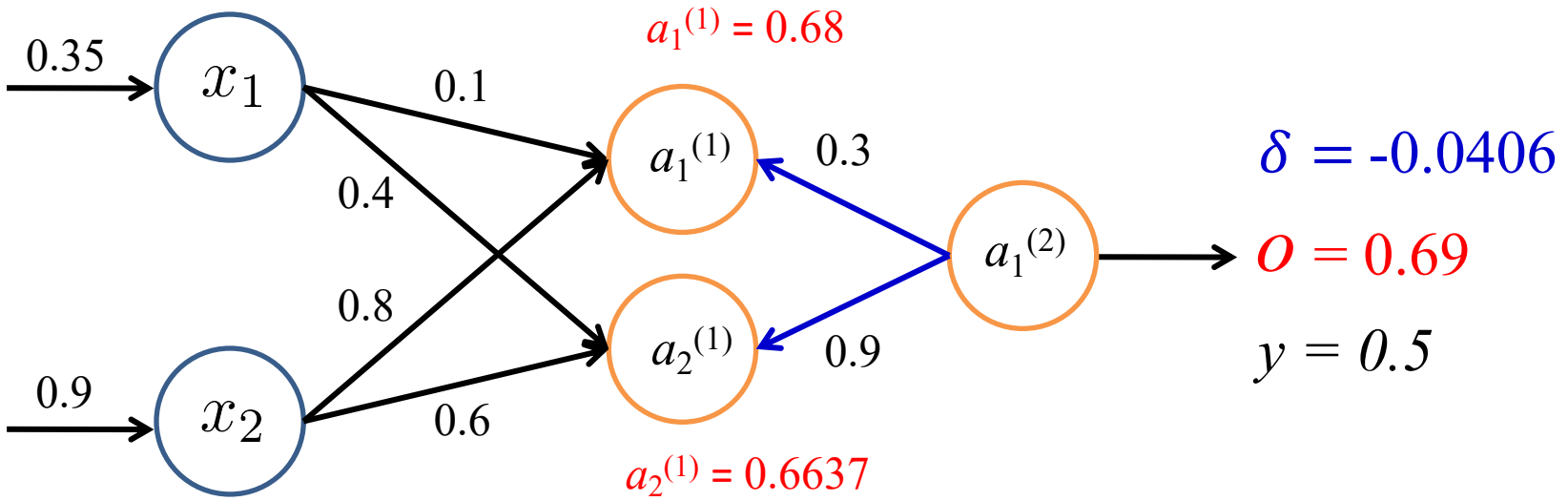
A Numerical Example



$$\delta = (y - o)\, o\, (1 - o)$$

$$\delta = (0.5 - 0.69) * 0.69 * (1 - 0.69) = -0.0406$$

A Numerical Example

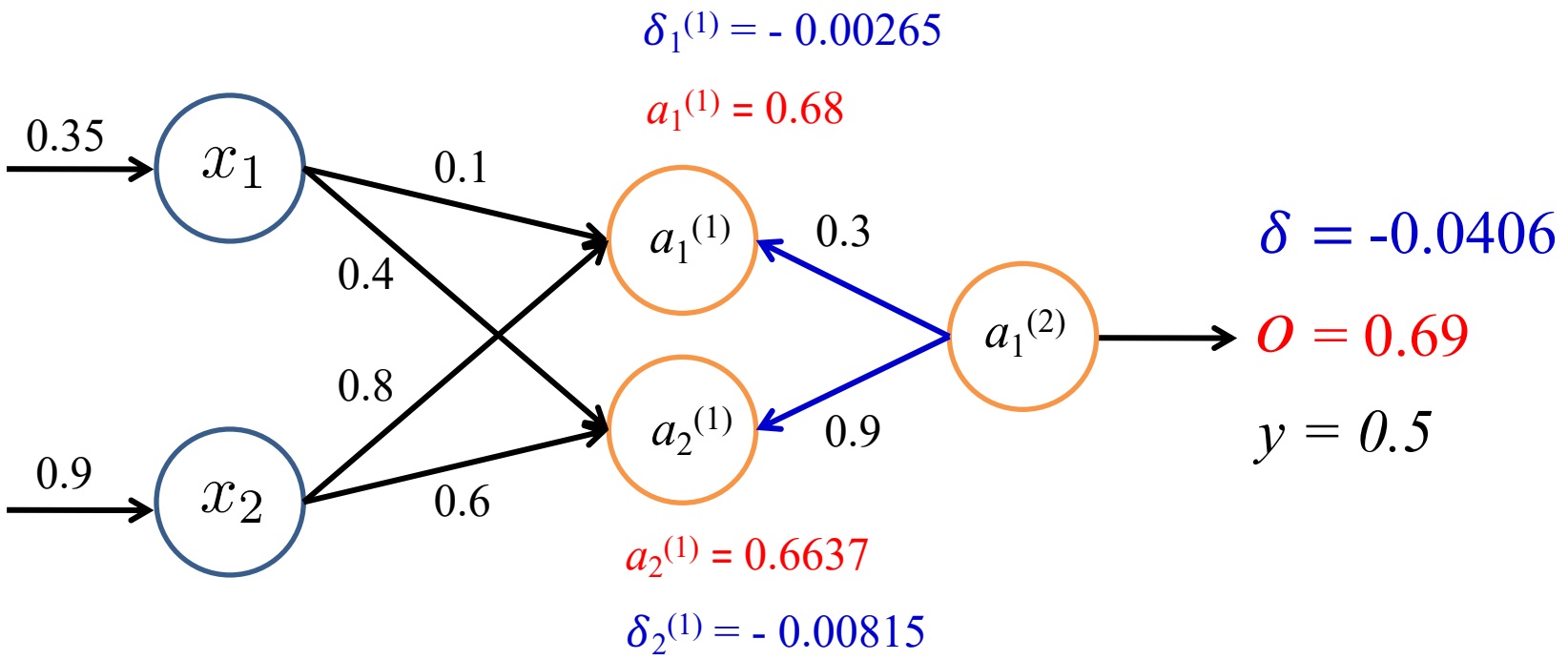


$$\delta_1^{(1)} = [\, w_{11}^{(2)}\, \delta \,]\, a_1^{(1)}\, (1 - a_1^{(1)}) =$$

$$= [0.3*(-0.0406)]*0.68*(1-0.68) = -0.00265$$

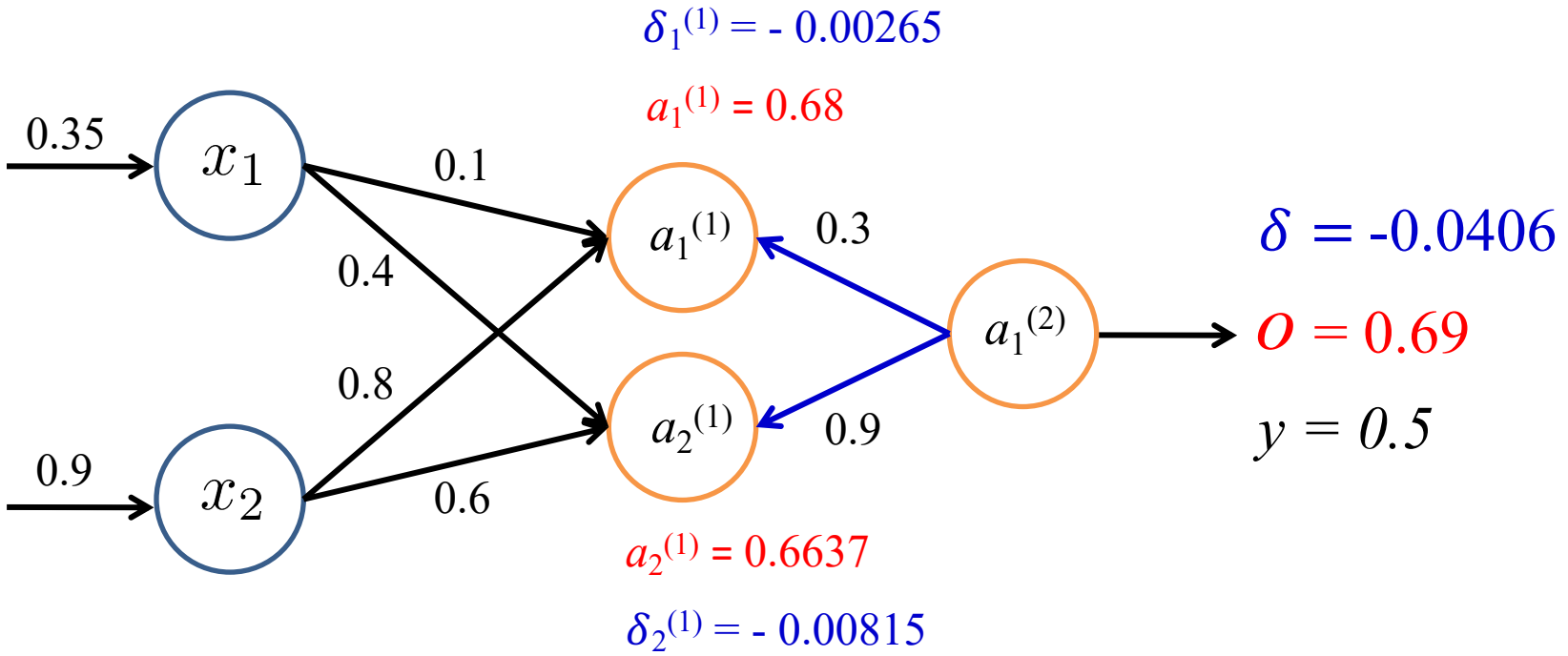
$$\delta_2^{(1)} = [\, w_{12}^{(2)}\, \delta \,]\, a_2^{(1)}\, (1 - a_2^{(1)}) =$$

$$= [0.9*(-0.0406)]*0.6637*(1-0.6637) = -0.00815$$



$$\delta_1^{(1)} = [\, w_{11}^{(2)}\, \delta \,]\, a_1^{(1)}\, (1 - a_1^{(1)}) =$$
$$= [0.3*(-0.0406)]*0.68*(1-0.68) = -0.00265$$

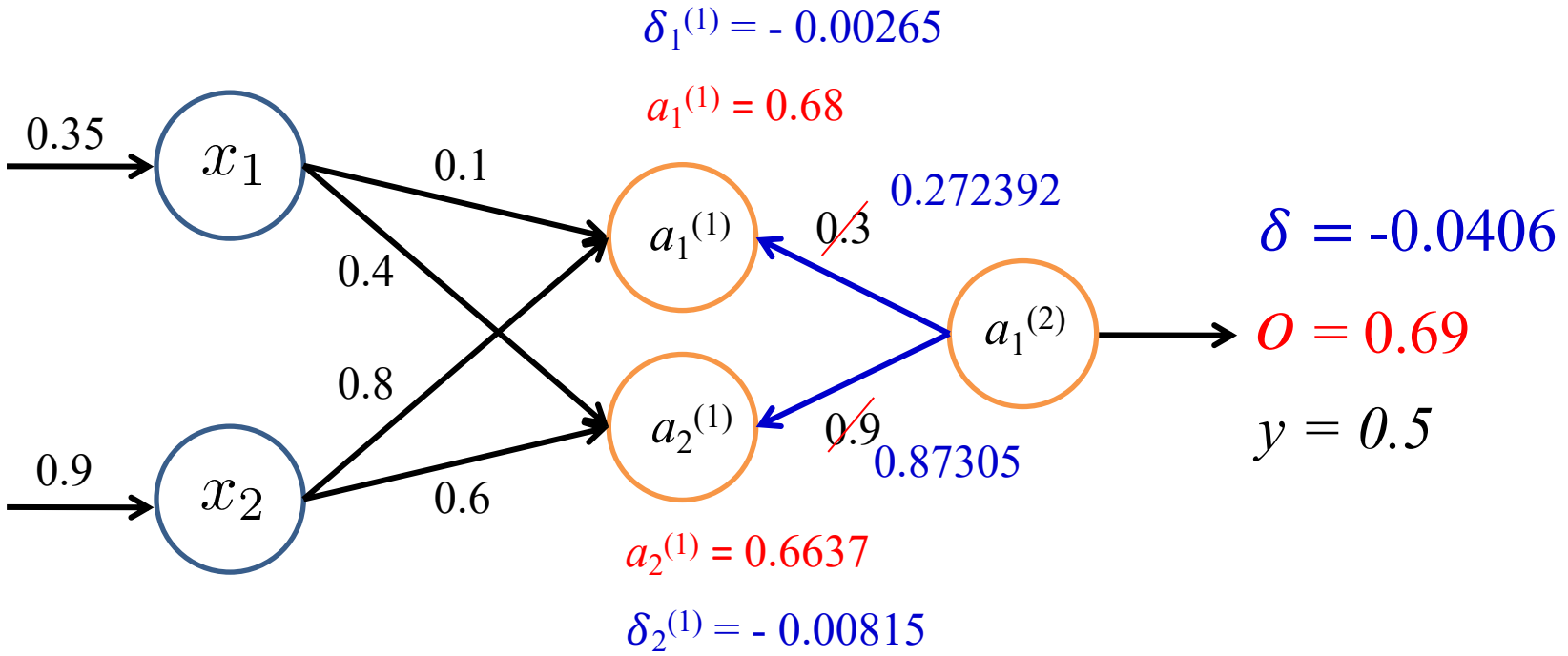
$$\delta_2^{(1)} = [\, w_{12}^{(2)}\, \delta \,]\, a_2^{(1)}\, (1 - a_2^{(1)}) =$$
$$= [0.9*(-0.0406)]*0.6637*(1-0.6637) = -0.00815$$



$$w_{ij}^{(l)}(\text{new}) = w_{ij}^{(l)}(\text{old}) + \alpha\ \delta\ a_i^{(l-1)}$$

$$w_{11}^{(2)}(\text{new}) = w_{11}^{(2)} + \alpha\ \delta\ a_1^{(1)} = 0.3 + 1*(-0.0406)*0.68 = 0.272392$$

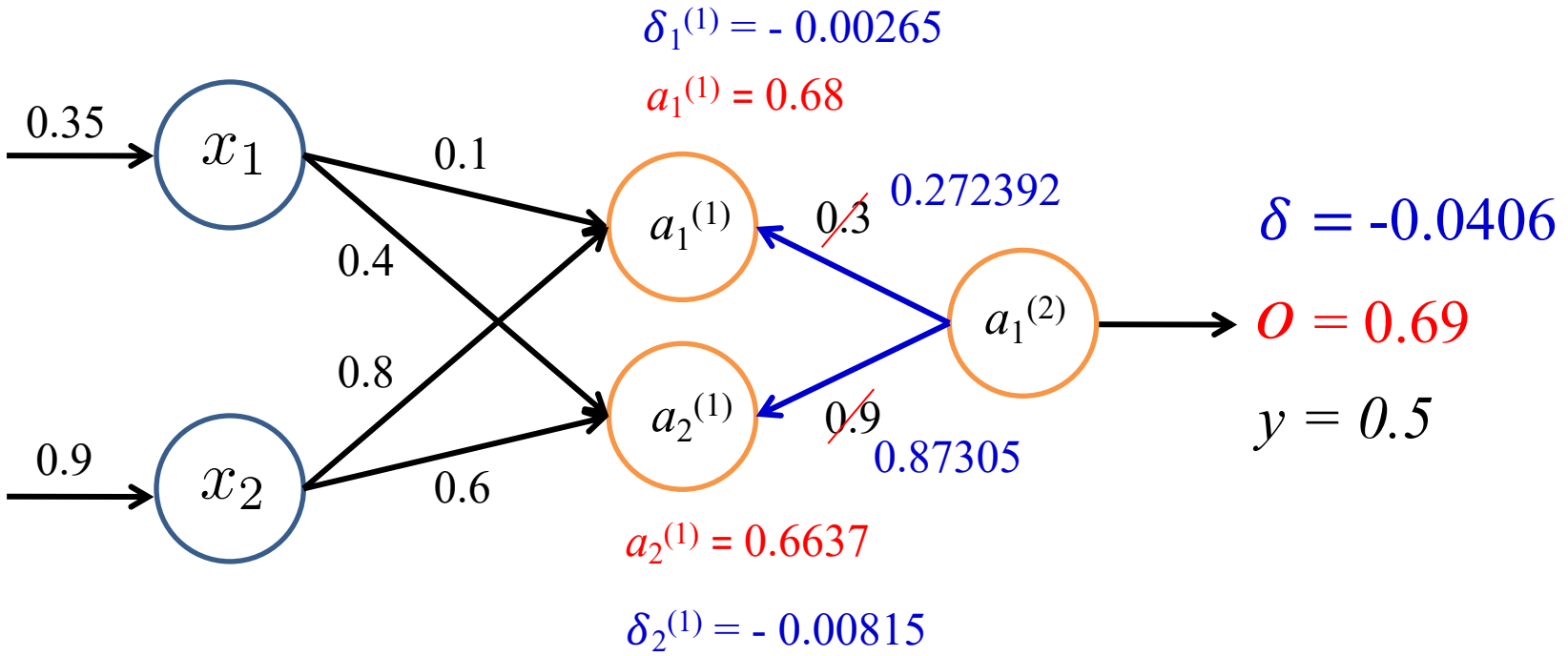
$$w_{12}^{(2)}(\text{new}) = w_{12}^{(2)} + \alpha\ \delta\ a_2^{(1)} = 0.9 + 1*(-0.0406)*0.6637 = 0.87305$$



$$w_{ij}^{(l)}(\text{new}) = w_{ij}^{(l)}(\text{old}) + \alpha\ \delta\ a_i^{(l-1)}$$

$$w_{11}^{(2)}(\text{new}) = w_{11}^{(2)} + \alpha\ \delta\ a_1^{(1)} = 0.3 + 1*(-0.0406)*0.68 = 0.272392$$

$$w_{12}^{(2)}(\text{new}) = w_{12}^{(2)} + \alpha\ \delta\ a_2^{(1)} = 0.9 + 1*(-0.0406)*0.6637 = 0.87305$$



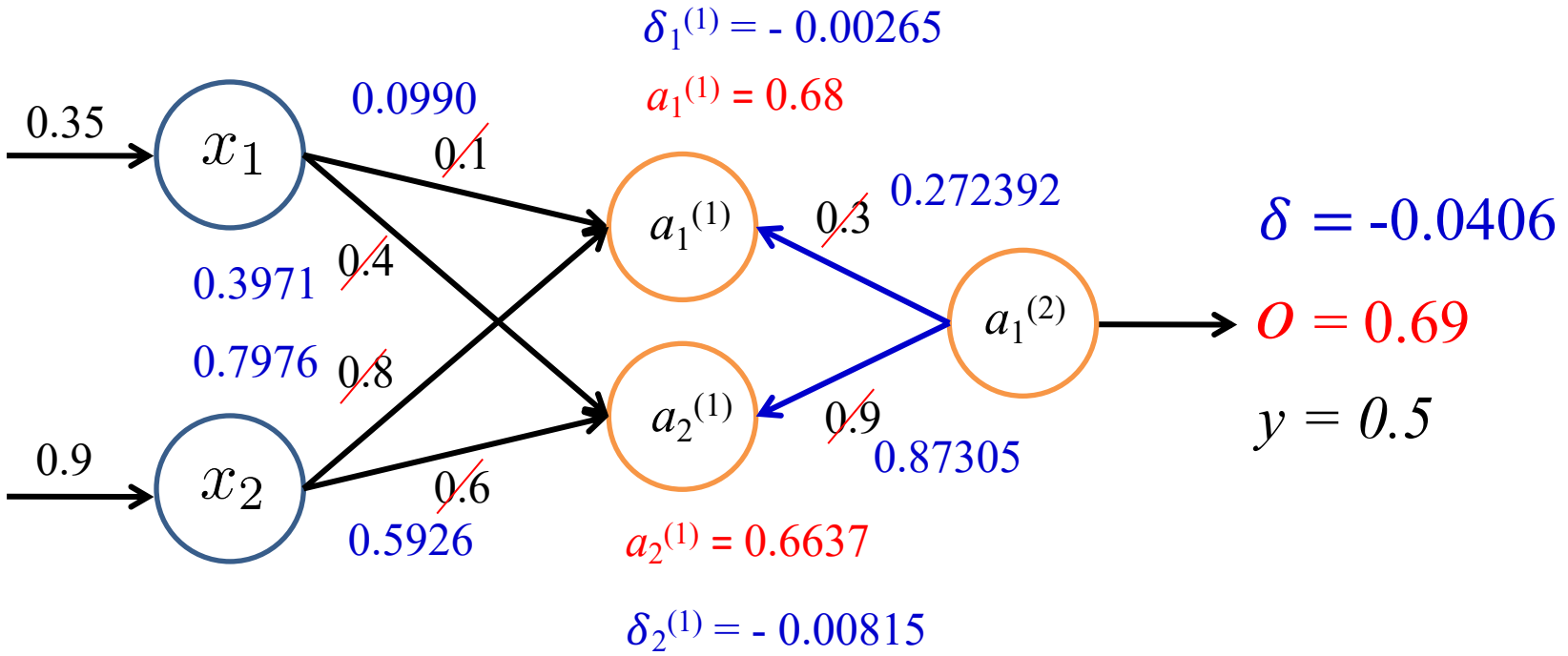
$$w_{ij}^{(l)}(\text{new}) = w_{ij}^{(l)}(\text{old}) + \alpha\ \delta_i^{(l)}\ a_i^{(l-1)}$$

$$w_{11}^{(1)}(\text{new}) = w_{11}^{(1)} + \alpha\ \delta_1^{(1)}\ x_1 = 0.1 + (-0.00265 * 0.35) = 0.0990$$

$$w_{12}^{(1)}(\text{new}) = w_{12}^{(1)} + \alpha\ \delta_1^{(1)}\ x_2 = 0.8 + (-0.00265 * 0.9) = 0.7976$$

$$w_{21}^{(1)}(\text{new}) = w_{21}^{(1)} + \alpha\ \delta_2^{(1)}\ x_1 = 0.4 + (-0.00815 * 0.35) = 0.3971$$

$$w_{22}^{(1)}(\text{new}) = w_{22}^{(1)} + \alpha\ \delta_2^{(1)}\ x_2 = 0.6 + (-0.00815 * 0.9) = 0.5926$$



$$w_{ij}^{(l)}(\text{new}) = w_{ij}^{(l)}(\text{old}) + \alpha\ \delta_i^{(l)}\ a_i^{(l-1)}$$

$$w_{11}^{(1)}(\text{new}) = w_{11}^{(1)} + \alpha\ \delta_1^{(1)}\ x_1 = 0.1 + (-0.00265 * 0.35) = 0.0990$$

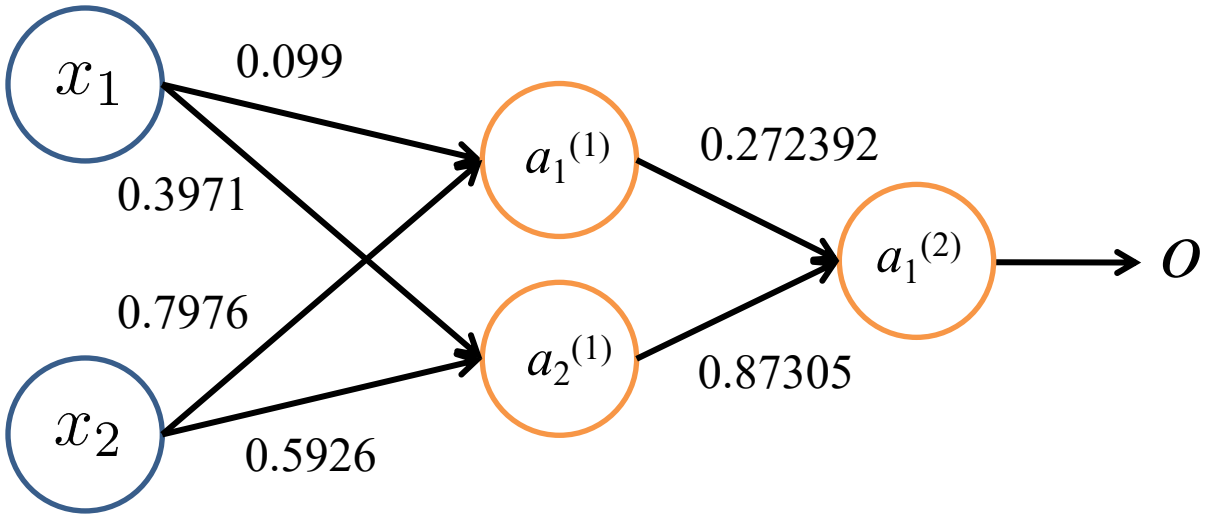
$$w_{12}^{(1)}(\text{new}) = w_{12}^{(1)} + \alpha\ \delta_1^{(1)}\ x_2 = 0.8 + (-0.00265 * 0.9) = 0.7976$$

$$w_{21}^{(1)}(\text{new}) = w_{21}^{(1)} + \alpha\ \delta_2^{(1)}\ x_1 = 0.4 + (-0.00815 * 0.35) = 0.3971$$

$$w_{22}^{(1)}(\text{new}) = w_{22}^{(1)} + \alpha\ \delta_2^{(1)}\ x_2 = 0.6 + (-0.00815 * 0.9) = 0.5926$$

A Numerical Example

The updated Network:



Thank You!

Questions?