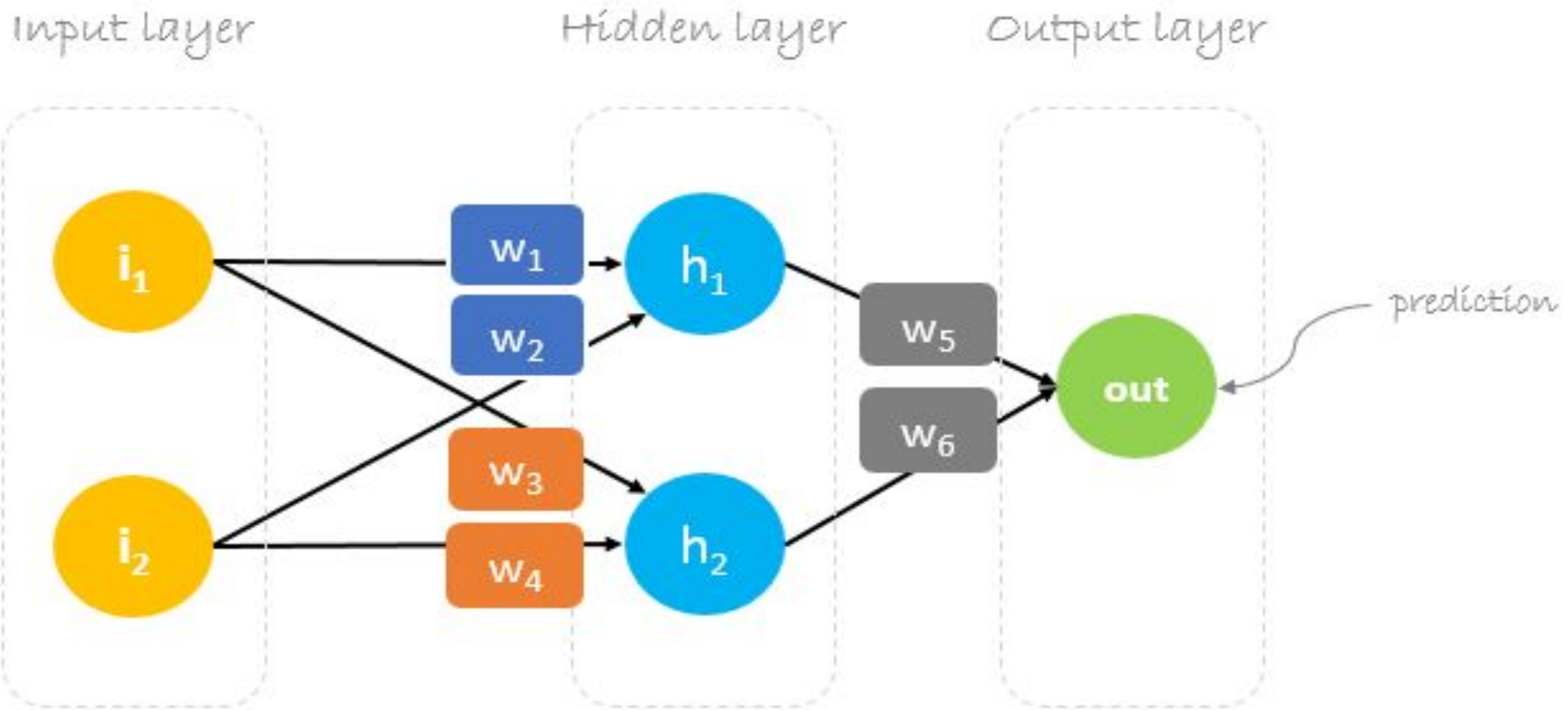
Lecture

- **Backpropagation Step by Step**

Backpropagation Step by Step



Initial Weights

- Initial weights are following:

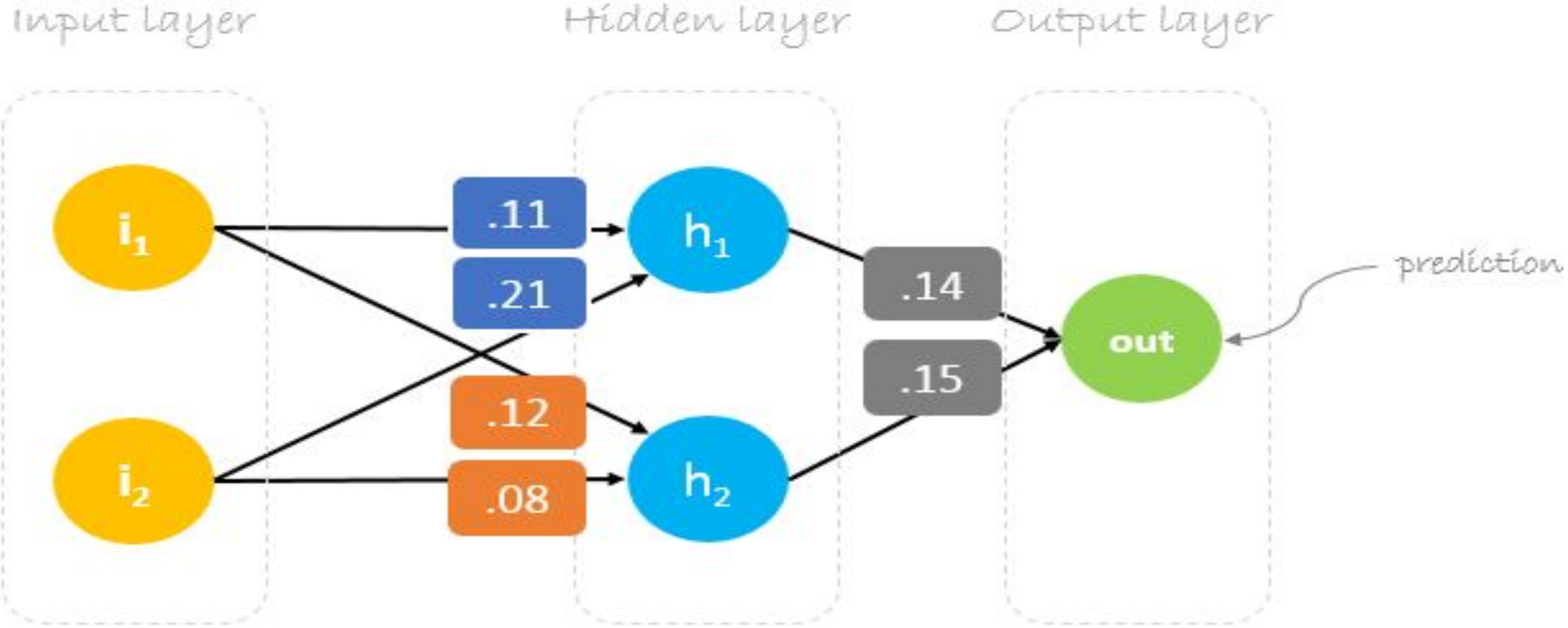
$$w_1 = 0.11,\ w_2 = 0.21,\ w_3 = 0.12,\ w_4 = 0.08,\ w_5 = 0.14 \text{ and } w_6 = 0.15$$

LAYOUT?

Initial Weights

- Initial weights are following:

$w_1 = 0.11$, $w_2 = 0.21$, $w_3 = 0.12$, $w_4 = 0.08$, $w_5 = 0.14$ and $w_6 = 0.15$



Dataset

- Dataset has one sample with two inputs and one output.

- Single sample is as following inputs=[2, 3] and output=[1].

LAYOUT?

Dataset

- Dataset has one sample with two inputs and one output.

- Single sample is as following inputs=[2, 3] and output=[1].

Forward Pass

- Use given weights and inputs to predict the output. Inputs are multiplied by weights; the results are then passed forward to next layer.

LAYOUT

?

Forward Pass

What are the values in hidden layer?

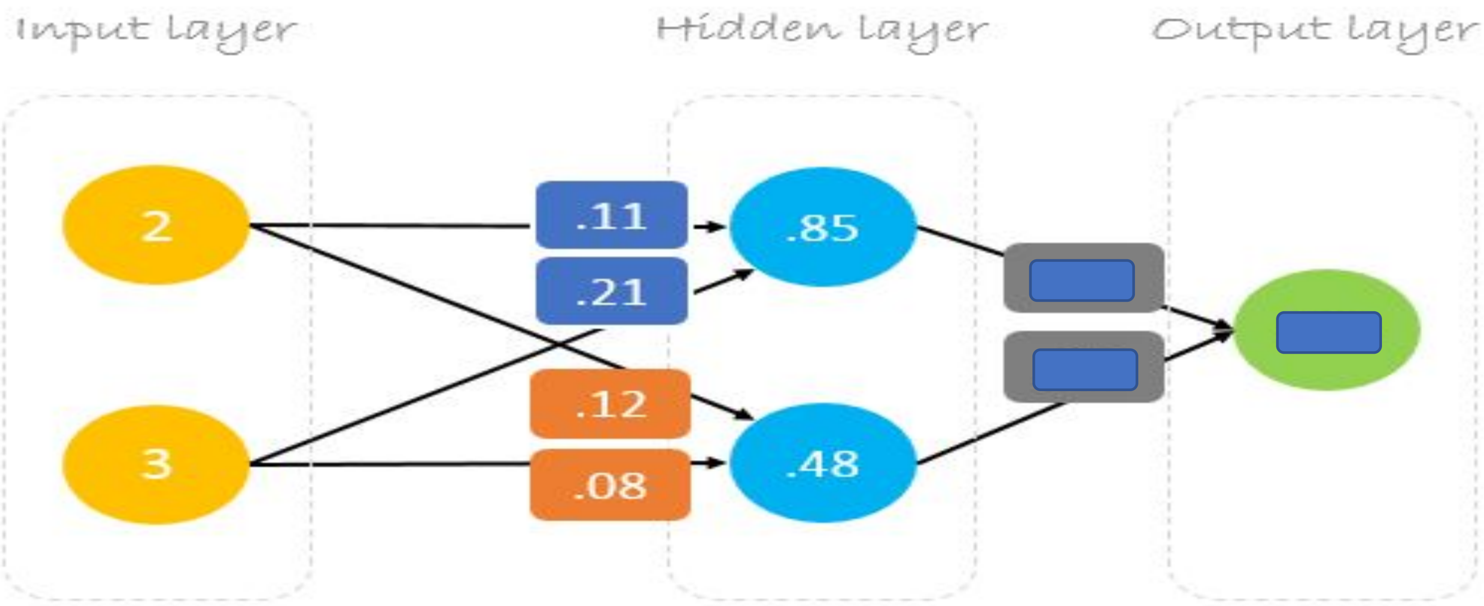
Compute them.

Forward Pass

What are the values in hidden layer?

(0.85, 0.48)

Forward Pass



$$\begin{bmatrix} 2 & 3 \end{bmatrix} \cdot \begin{bmatrix} 0.11 & 0.12 \\ 0.21 & 0.08 \end{bmatrix} = \begin{bmatrix} 0.85 & 0.48 \end{bmatrix}$$

Forward Pass

What is the value in output layer?

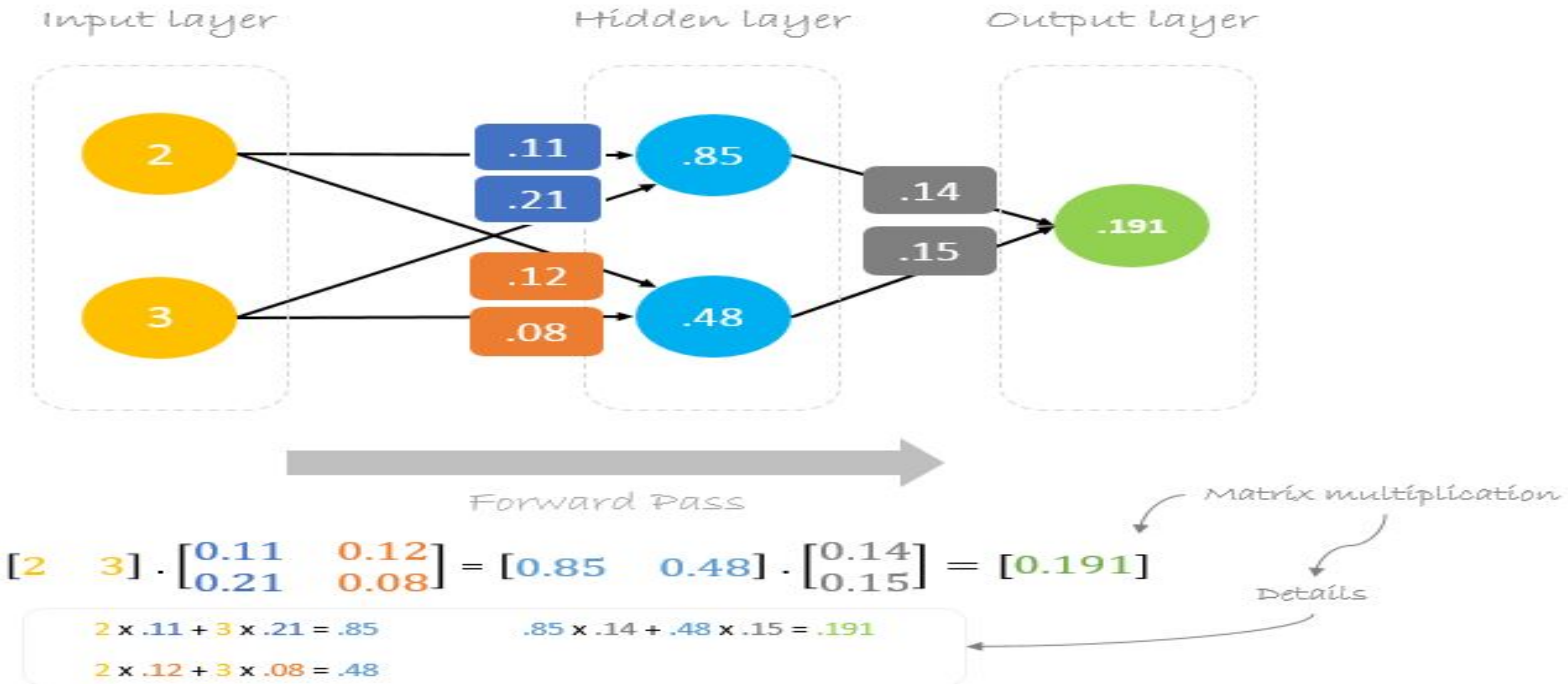
Compute the value.

Forward Pass

What is the values in output layer?

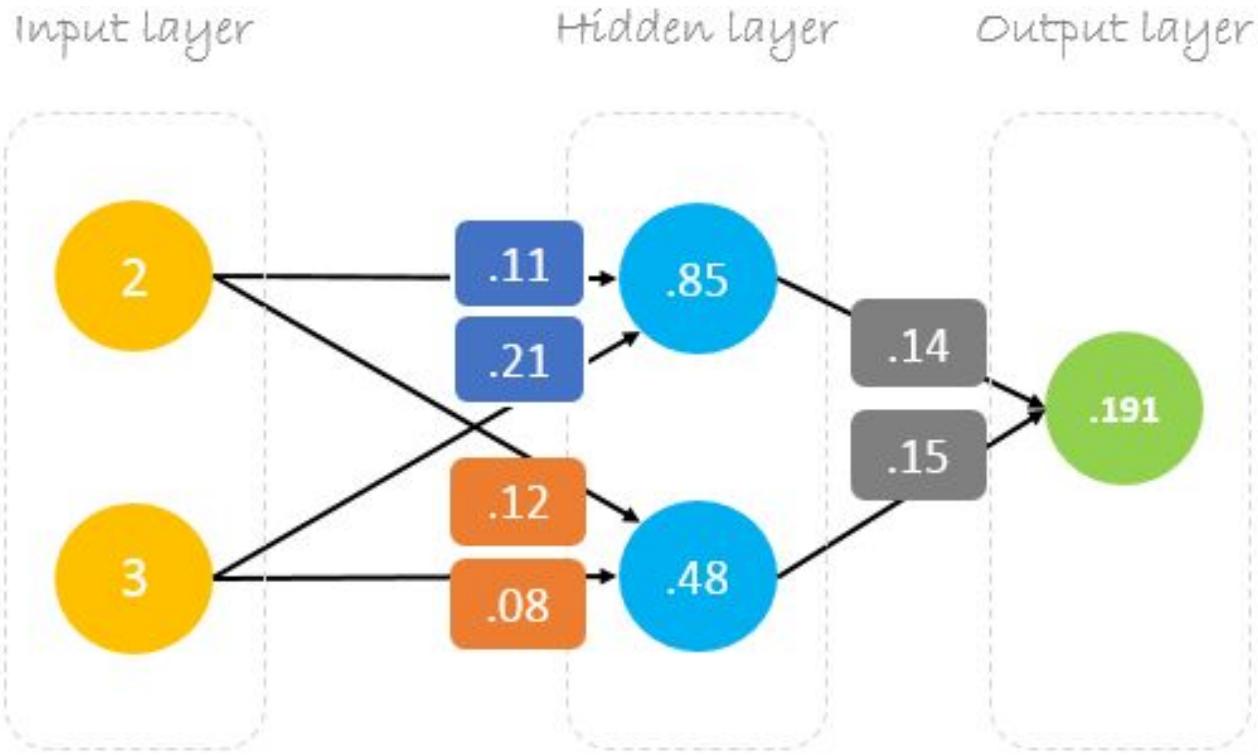
(0.191)

Forward Pass



Forward Pass

- Use given weights and inputs to predict the output. Inputs are multiplied by weights; the results are then passed forward to next layer.



Calculate Error

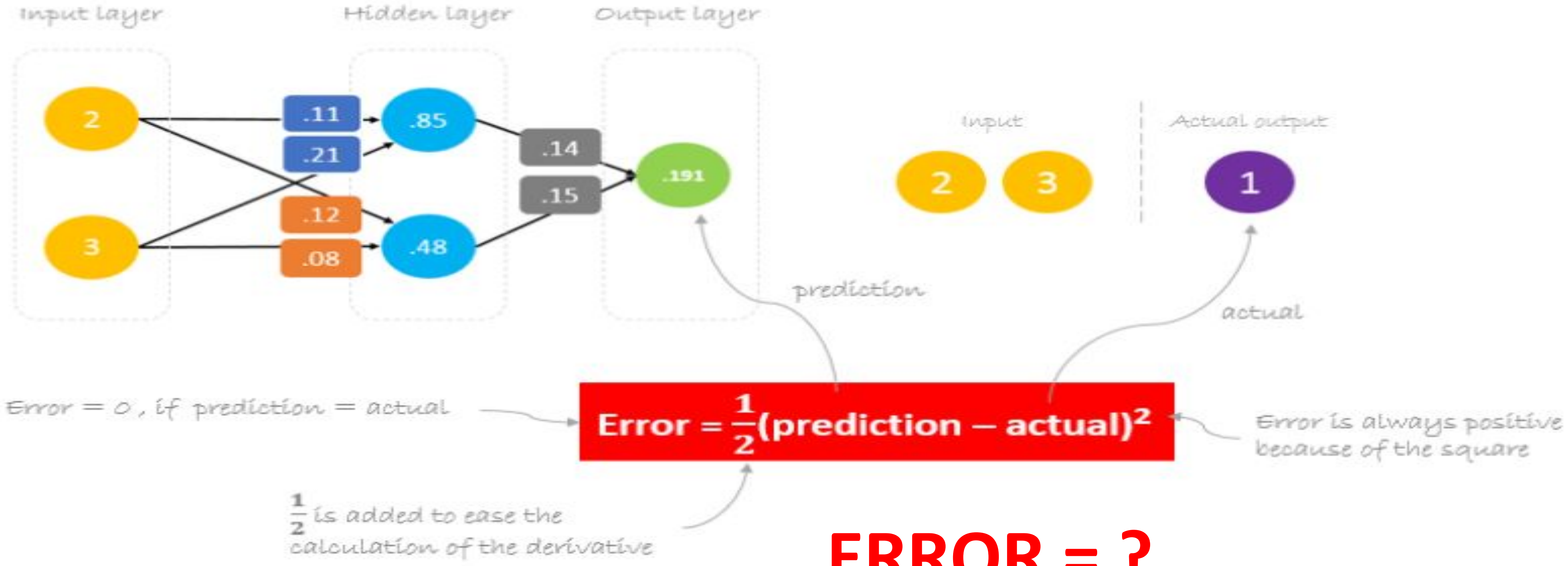
- Network output, or **prediction (0.191)**, is not even close to **actual output (1)**. We can calculate the difference or the error.

LAYOUT

?

Calculate Error

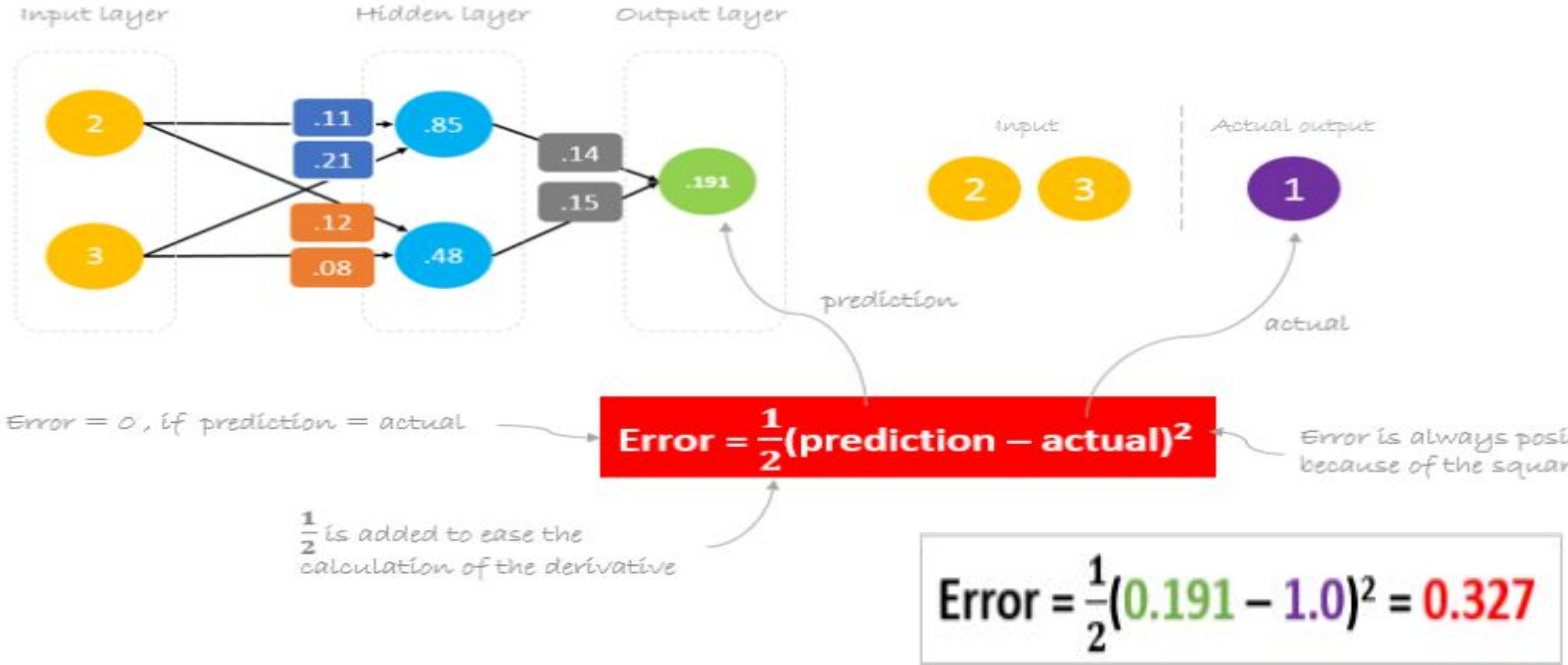
- Network output, or **prediction (0.191)**, is not even close to **actual output (1)**. We can calculate the difference or the error as following.



ERROR = ?

Calculate Error

- Network output, or **prediction (0.191)**, is not even close to **actual output (1)**. We can calculate the difference or the error as following.



Reducing Error

- Our main goal of the training is to reduce the **error** or the difference between **prediction** and **actual output**.

- Since **actual output** is constant, “not changing”, the only way to reduce the error is to change **prediction** value.

- The question now is, how to change **prediction** value?

Reducing Error

- Our main goal of the training is to reduce the **error** or the difference between **prediction** and **actual output**.

- Since **actual output** is constant, "not changing", the only way to reduce the error is to change **prediction** value. The question now is, how to change **prediction** value?

- By decomposing **prediction** into its basic elements we can find that **weights** are the variable elements affecting **prediction** value.

- In other words, in order to change **prediction** value, we need to **change weights values**.

Reducing Error

prediction = out

prediction = (?) w_5 + (?) w_6

prediction = (?) w_5 + (?) w_6

to change ***prediction*** value,
we need to change ***weights***

Reducing Error

$$\text{prediction} = out$$

$$\text{prediction} = (\boxed{?})\ w_5 + (\boxed{?})\ w_6$$

$$\text{prediction} = (\boxed{\quad ? \quad})\ w_5 + (\boxed{\quad ? \quad})\ w_6$$

to change ***prediction*** value,
we need to change ***weights***

Reducing Error

$$\text{prediction} = \text{out}$$

$$\text{prediction} = (h_1)\, w_5 + (\boxed{?})\, w_6$$

$$\text{prediction} = (\boxed{?})\, w_5 + (\boxed{?})\, w_6$$

to change ***prediction*** value,
we need to change ***weights***

Reducing Error

$$\text{prediction} = \text{out}$$

$$\text{prediction} = (h_1)\, w_5 + (h_2)\, w_6$$

$$\text{prediction} = (\ ?\ )\, w_5 + (\ ?\ )\, w_6$$

to change ***prediction*** value,
we need to change ***weights***

Reducing Error

prediction = out

prediction = $(h_1)\ w_5 + (h_2)\ w_6$

prediction = (?) w_5 + (?) w_6

to change ***prediction*** value,
we need to change ***weights***

Reducing Error

$$\textbf{prediction} = \text{out}$$

$$\textbf{prediction} = (h_1)\, w_5 + (h_2)\, w_6$$

$$\textbf{prediction} = (i_1\, \boldsymbol{w_1} + i_2\, \boldsymbol{w_2})\, \boldsymbol{w_5} + (\ ?\ )\, \boldsymbol{w_6}$$

to change ***prediction*** value,
we need to change ***weights***

Reducing Error

$$\textbf{prediction} = \text{out}$$

$$\textbf{prediction} = (h_1)\, w_5 + (h_2)\, w_6$$

$$\textbf{prediction} = (i_1\, \boldsymbol{w_1} + i_2\, \boldsymbol{w_2})\, \boldsymbol{w_5} + (i_1\, \boldsymbol{w_3} + i_2\, \boldsymbol{w_4})\, \boldsymbol{w_6}$$

to change ***prediction*** value,
we need to change ***weights***

Reducing Error

$$\mathbf{prediction} = \text{out}$$

$$\mathbf{prediction} = (h_1)\, w_5 + (h_2)\, w_6$$

$h_1 = i_1 w_1 + i_2 w_2$

$h_2 = i_1 w_3 + i_2 w_4$

$$\mathbf{prediction} = (i_1\, \boldsymbol{w_1} + i_2\, \boldsymbol{w_2})\, \boldsymbol{w_5} + (i_1\, \boldsymbol{w_3} + i_2\, \boldsymbol{w_4})\, \boldsymbol{w_6}$$

to change ***prediction*** value,
we need to change ***weights***

Reducing Error

$$\textbf{prediction} = out$$

$$\textbf{prediction} = (h_1)\, w_5 + (h_2)\, w_6$$

$$h_1 = i_1 w_1 + i_2 w_2$$

$$h_2 = i_1 w_3 + i_2 w_4$$

$$\textbf{prediction} = (i_1\, w_1 + i_2\, w_2)\, w_5 + (i_1\, w_3 + i_2\, w_4)\, w_6$$

to change ***prediction*** value,
we need to change ***weights***

The question now is **how to change\update the weights value so that the error is reduced?**

The answer is **Backpropagation!**

Backpropagation

- **Backpropagation**, short for “backward propagation of errors”, is a mechanism used to update the **weights** using gradient descent.

- It calculates the gradient of the error function with respect to the neural network’s weights.

- The calculation proceeds backwards through the network.

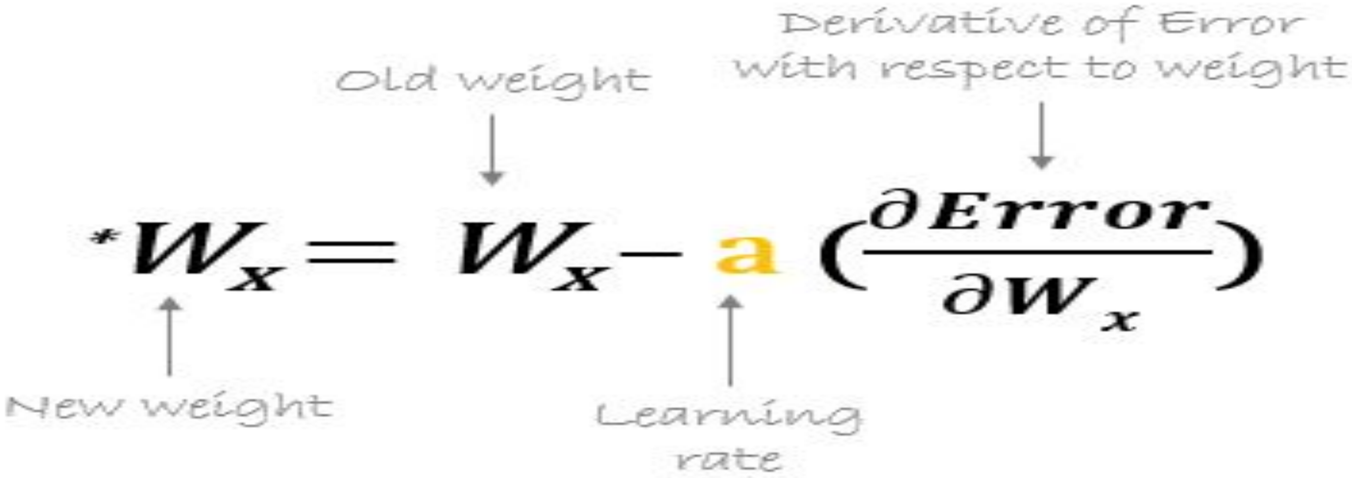
- **Gradient descent** is an iterative optimization algorithm for finding the minimum of a function; in our case we want to minimize the error function.

Gradient Descent

- **Gradient descent** is an iterative optimization algorithm for finding the minimum of a function; in our case we want to minimize the error function.

- To find a local minimum of a function using gradient descent, one takes steps proportional to the negative of the gradient of the function at the current point.

Update Weight



- For example, to update w_6, we take the current w_6 and subtract the partial derivative of **error** function with respect to w_6. Optionally, we multiply the derivative of the **error** function by a selected number to make sure that the new updated **weight** is minimizing the error function; this number is called ***learning rate***.

Local Minima

- Training is essentially minimizing the mean square error function.

  ► Key problem is avoiding local minima

  ► Traditional techniques for avoiding local minima:

  ■ Simulated annealing: Perturb the weights in progressively smaller amounts

  ■ Genetic algorithms: Use the weights as chromosomes, Apply natural selection, mating, and mutations to these chromosomes

REFERENCES

- John Paul Mueller and Luca Massaron, *Machine Learning (in Python and R) For Dummies*, John Wiley & Sons, 2016

- Tom M. Mitchell, *Machine Learning*, McGrawHill Publications, Indian Edition, 2017

Tutorial:

- http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture14.pdf
- https://www.bing.com/images/search?view=detailV2&ccid=wQD8qKaG&id=07A6E6CB5195A27FD4E601CAB57551B787D0D817&thid=OIP.wQD8qKaGR_lf902l5hx18wHaD4&mediaurl=https://cdn-images-1.medium.com/max/1600/1*_M4bZyuwaGby6KMiYVYXvg.jpeg&exph=840&expw=1600&q=Multilayer+Neural+Network&simid=608048841338979193&ck=3093FFEA6AD1D032F7E34A5AF059AE83&selectedIndex=0&FORM=IRPRST&ajaxhist=0
- https://hmkcode.com/ai/backpropagation-step-by-step/