Deep Learning Basics

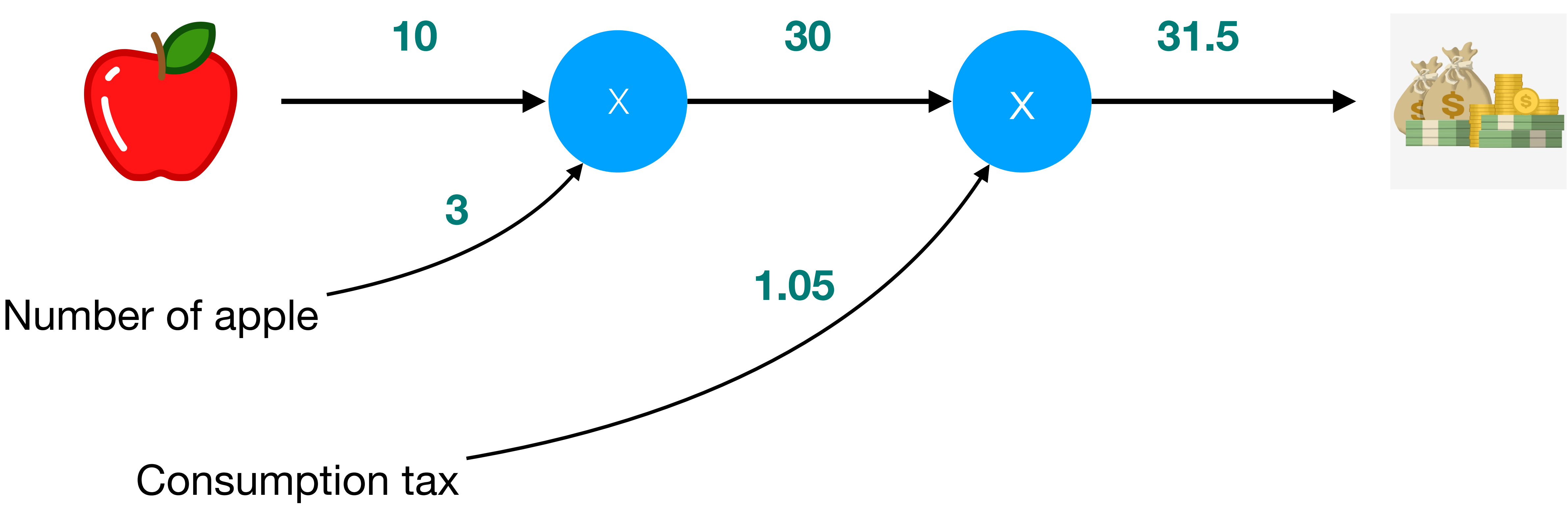
3: Backpropagation

Francis Steen and Xinyu You
Red Hen Lab
August 2019

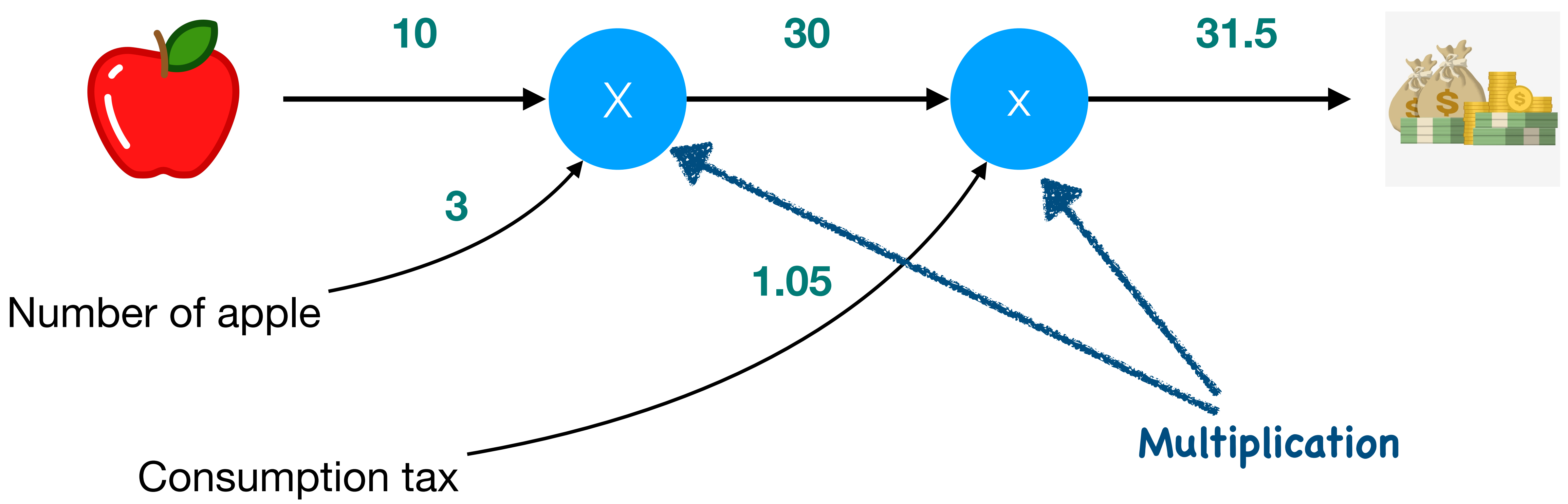
A. Compute Graph

Compute Graph

- Question: You are going to buy 3 apples. An apple's price is 10, plus 5% consumption tax. How much should you pay?

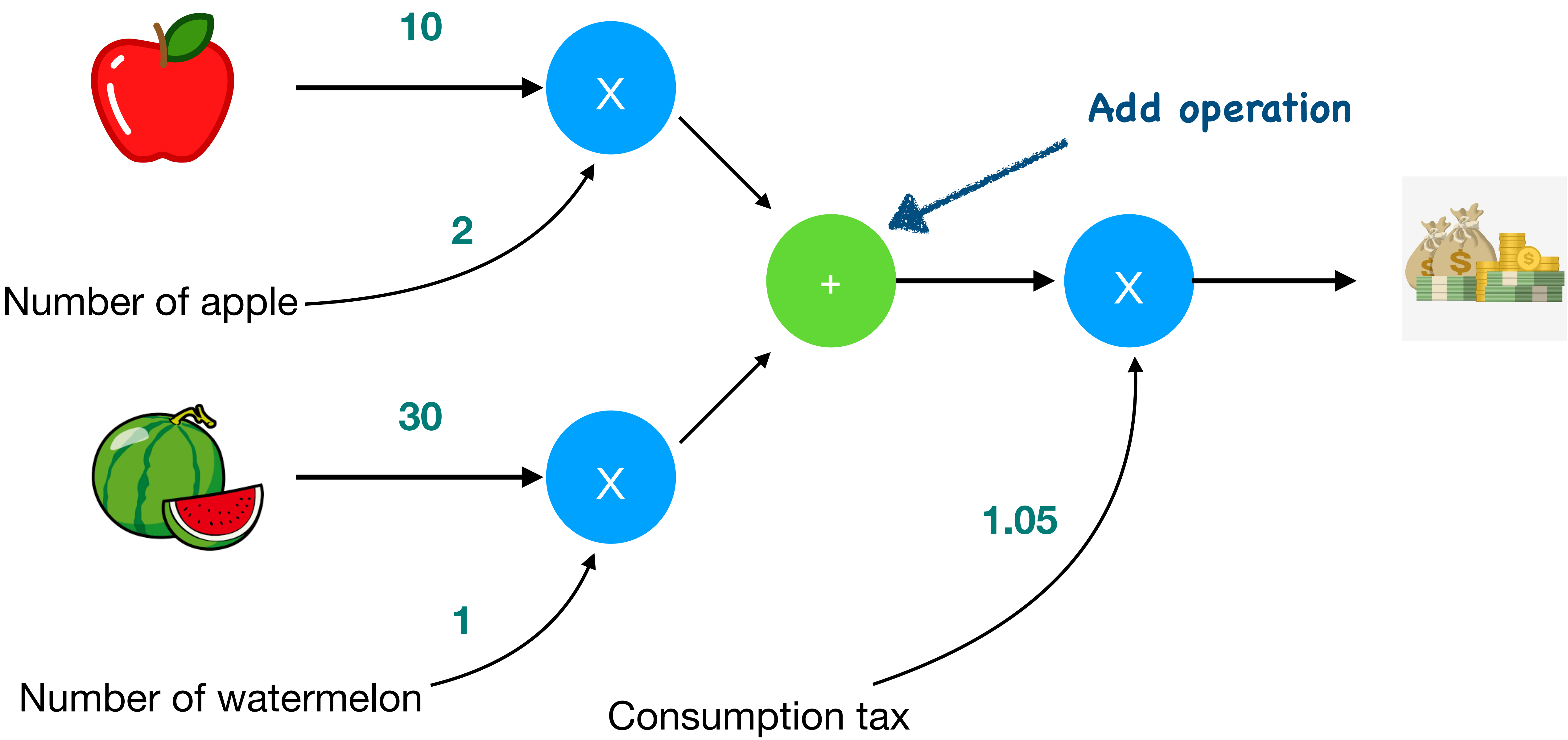


Compute Graph



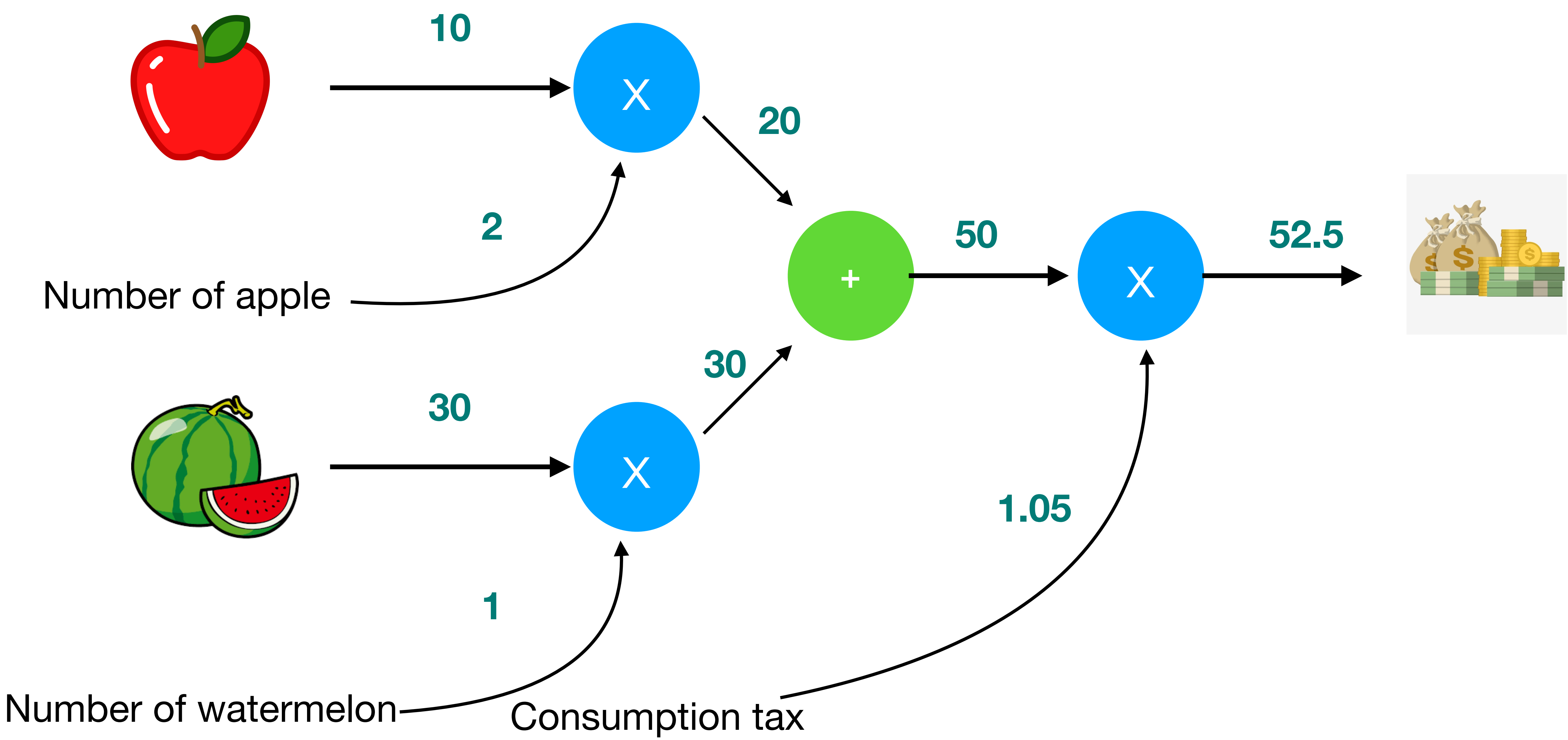
Compute Graph

- Question: You are going to buy 2 apples and 1 watermelon. An apple's price is 10 plus 5% consumption tax. A watermelon's price is 30 plus 5% consumption tax. How much should you pay?

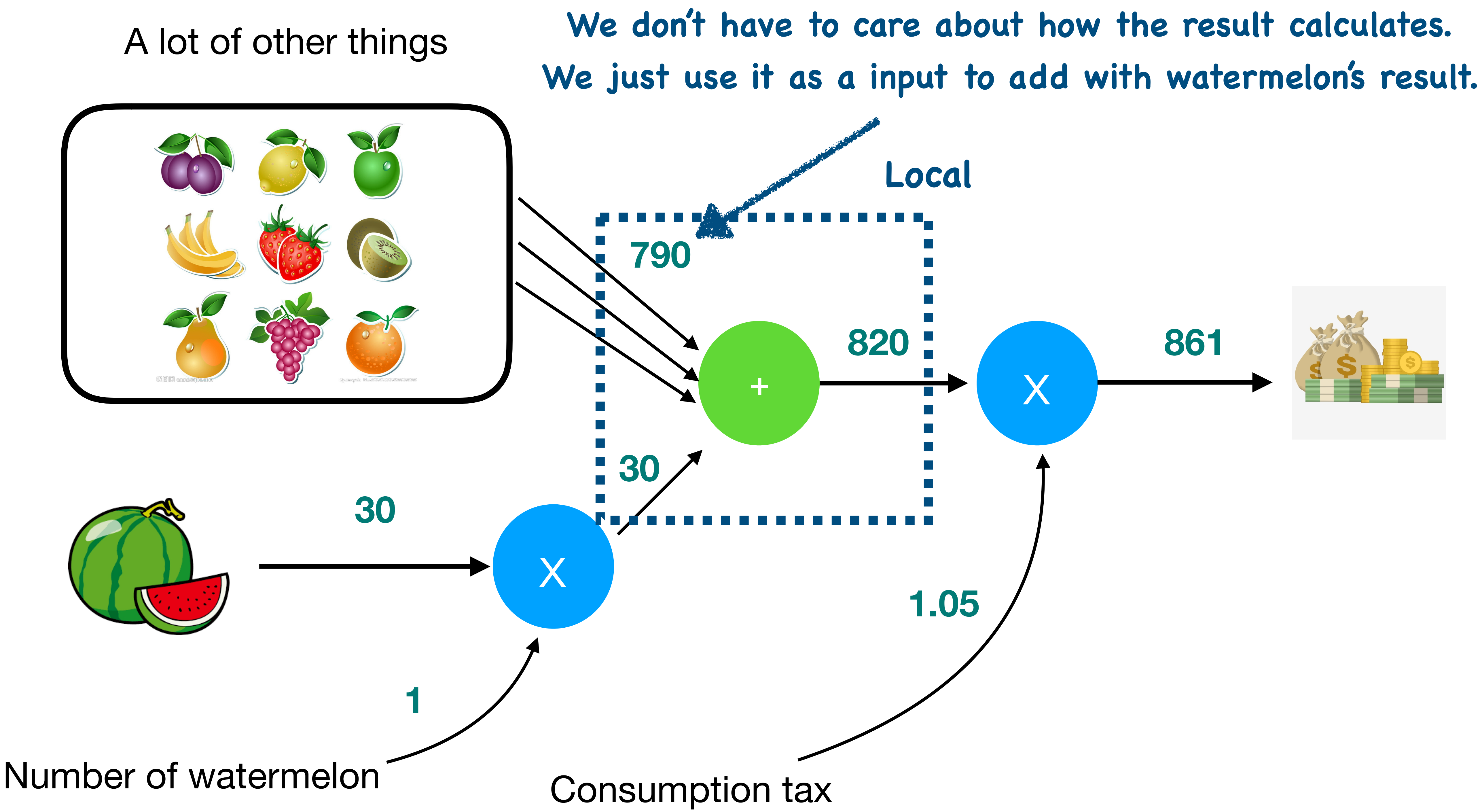


Compute Graph

We compute the graph from left to right in the direction of the arrow.

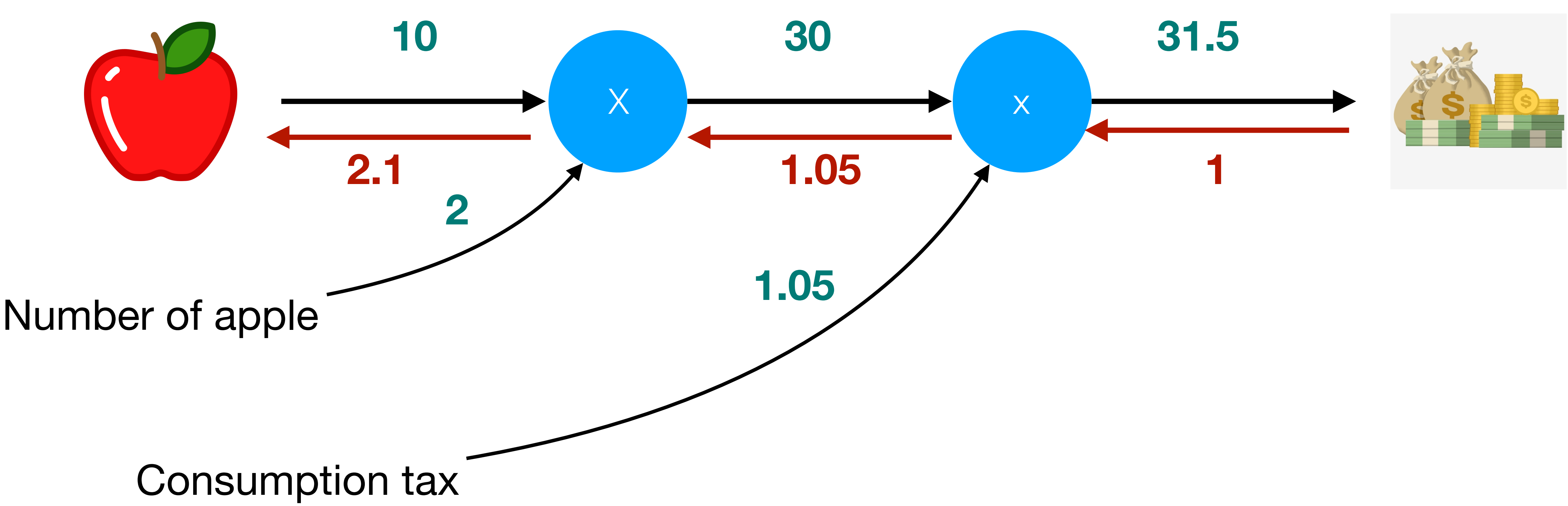


Local Compute



Compute Graph Advantage

- All the intermediate results can be saved using the compute graph.
- We can calculate derivatives efficiently by propagating backwards. (We will cover this later)



B. The Chain Rule

The Chain Rule

Chain Rule

If f and g are both differentiable and $F(x)$ is the composite function defined by $F(x) = f(g(x))$ then F is differentiable and F' is given by the product

$$F'(x) = \underbrace{f'}_{\text{Differentiate outer function}}(g(x))\ \underbrace{g'}_{\text{Differentiate inner function}}(x)$$

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial x} \longrightarrow \frac{\partial f}{\partial x} = \frac{\partial f}{\cancel{\partial g}}\frac{\cancel{\partial g}}{\partial x}$$

Example

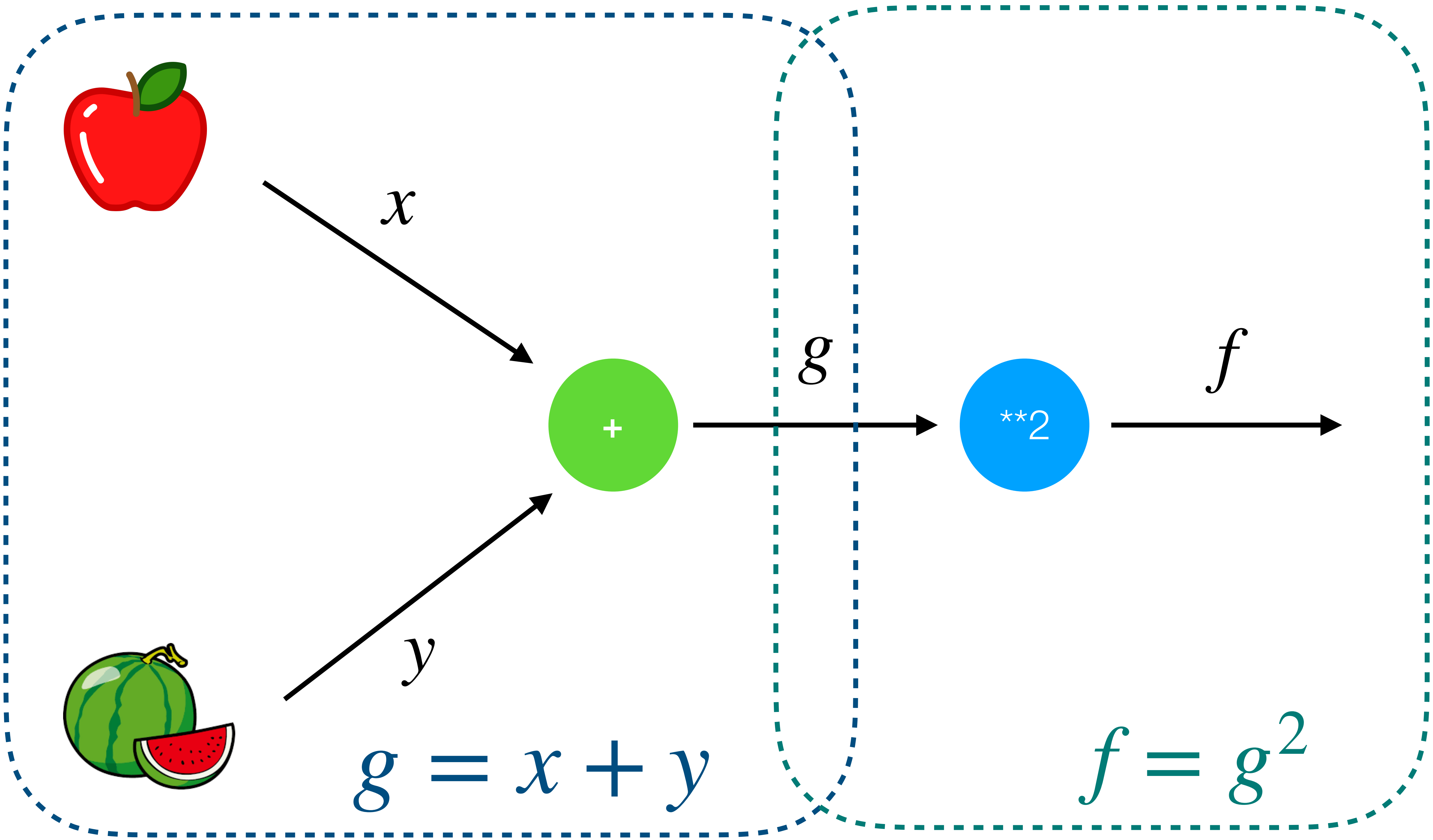
$$f = g^2 \longrightarrow \frac{\partial f}{\partial g} = 2g$$

$$g = x + y \longrightarrow \frac{\partial g}{\partial x} = 1$$

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial x} = 2g * 1 \longrightarrow \frac{\partial f}{\partial x} = 2(x + y)$$

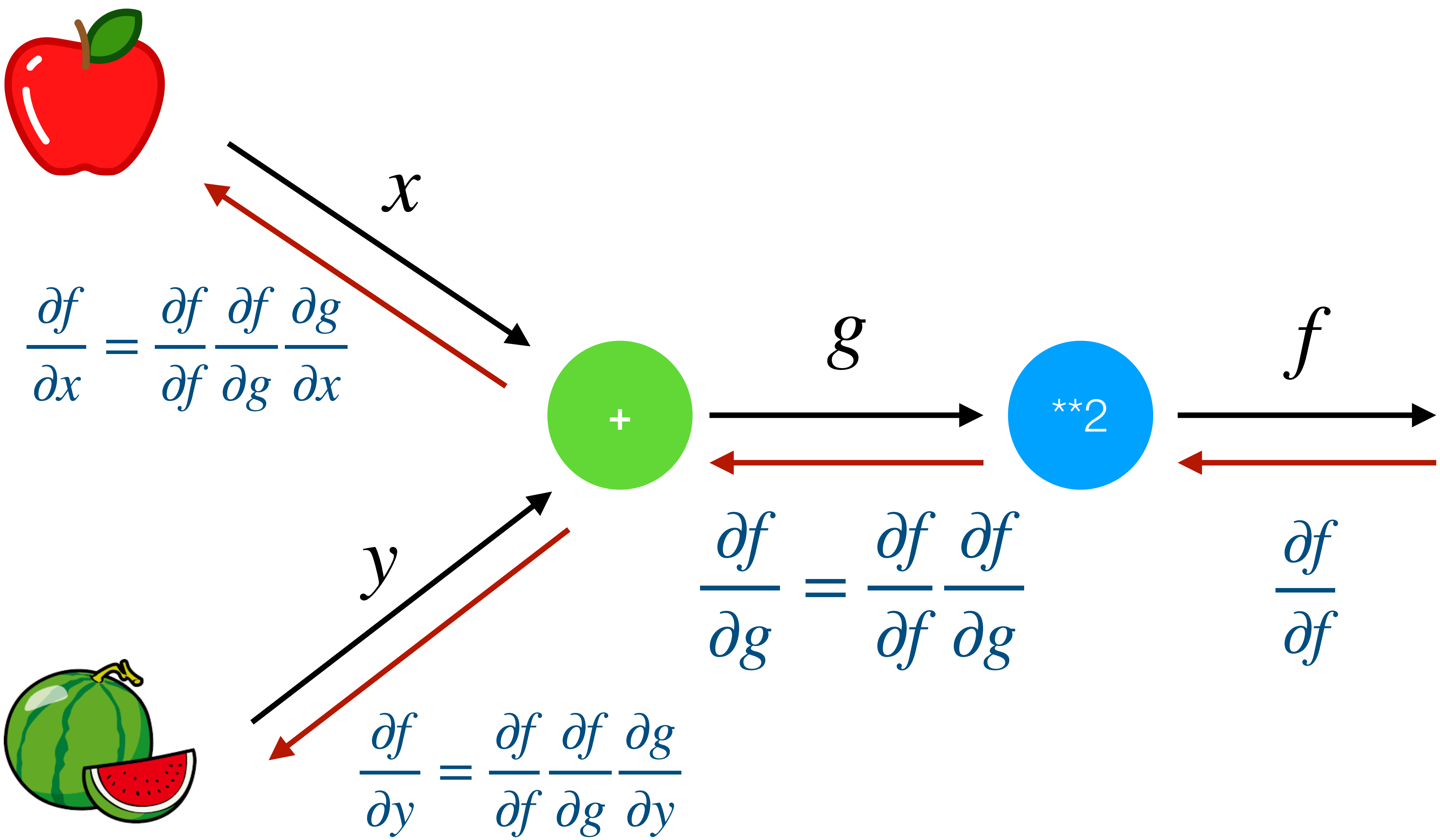
Compute graph & Chain Rule

Use the compute graph to represent the function calculations



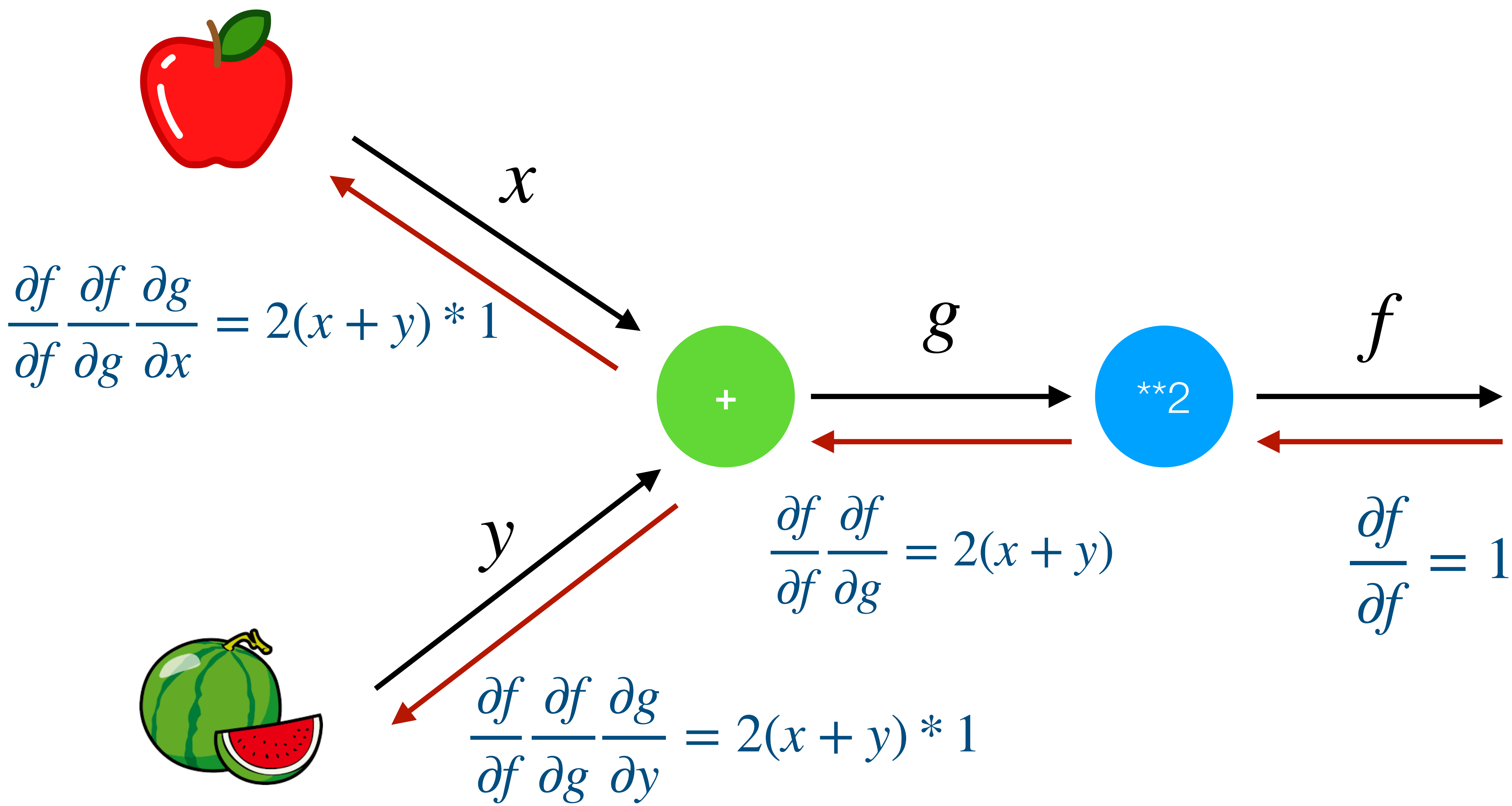
Compute graph &Chain Rule

Use the chain rule for back propagation



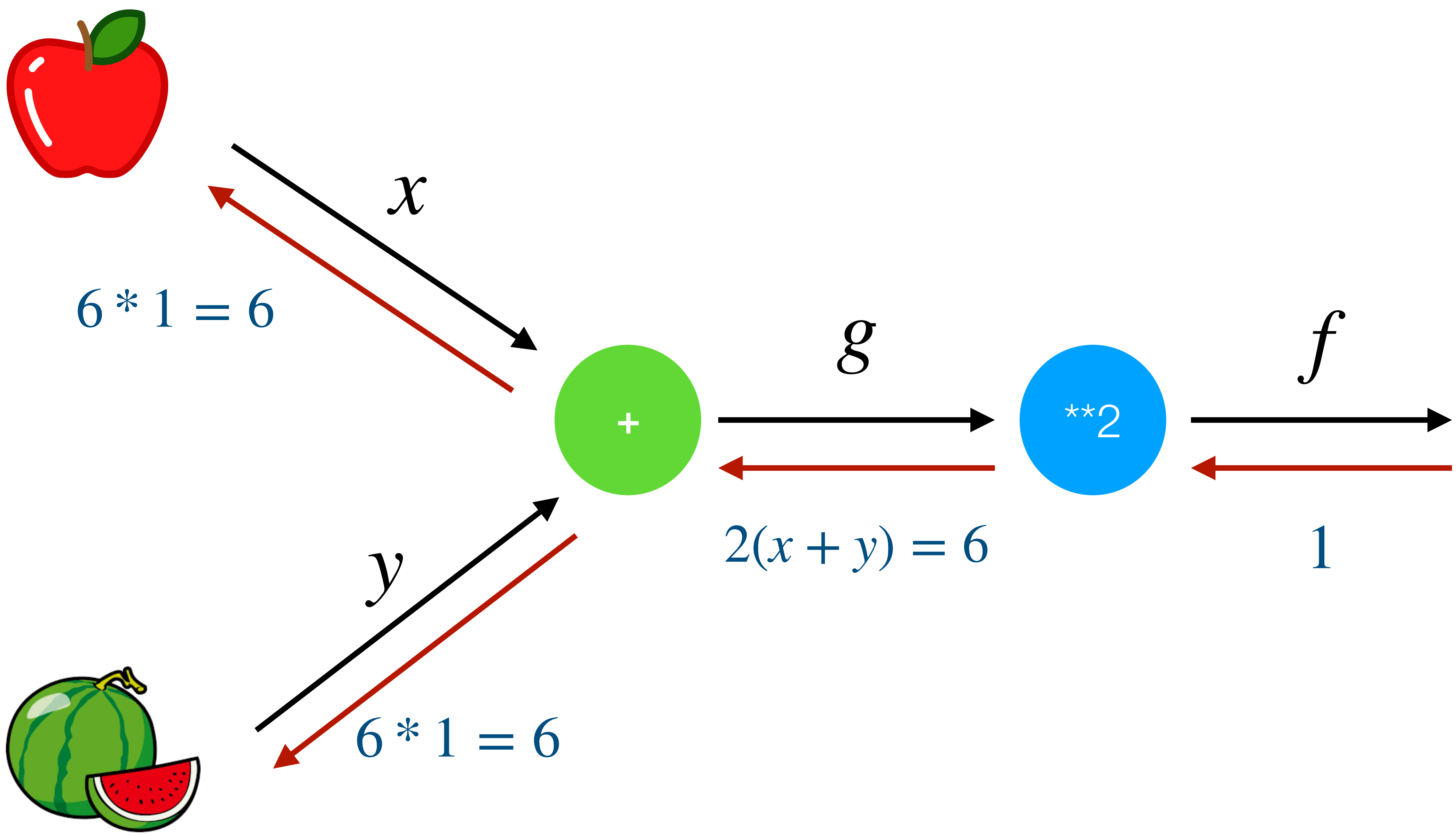
Compute graph &Chain Rule

Use the chain rule for back propagation



Compute graph &Chain Rule

Let x = 1, y = 2, we can calculate each gradient.

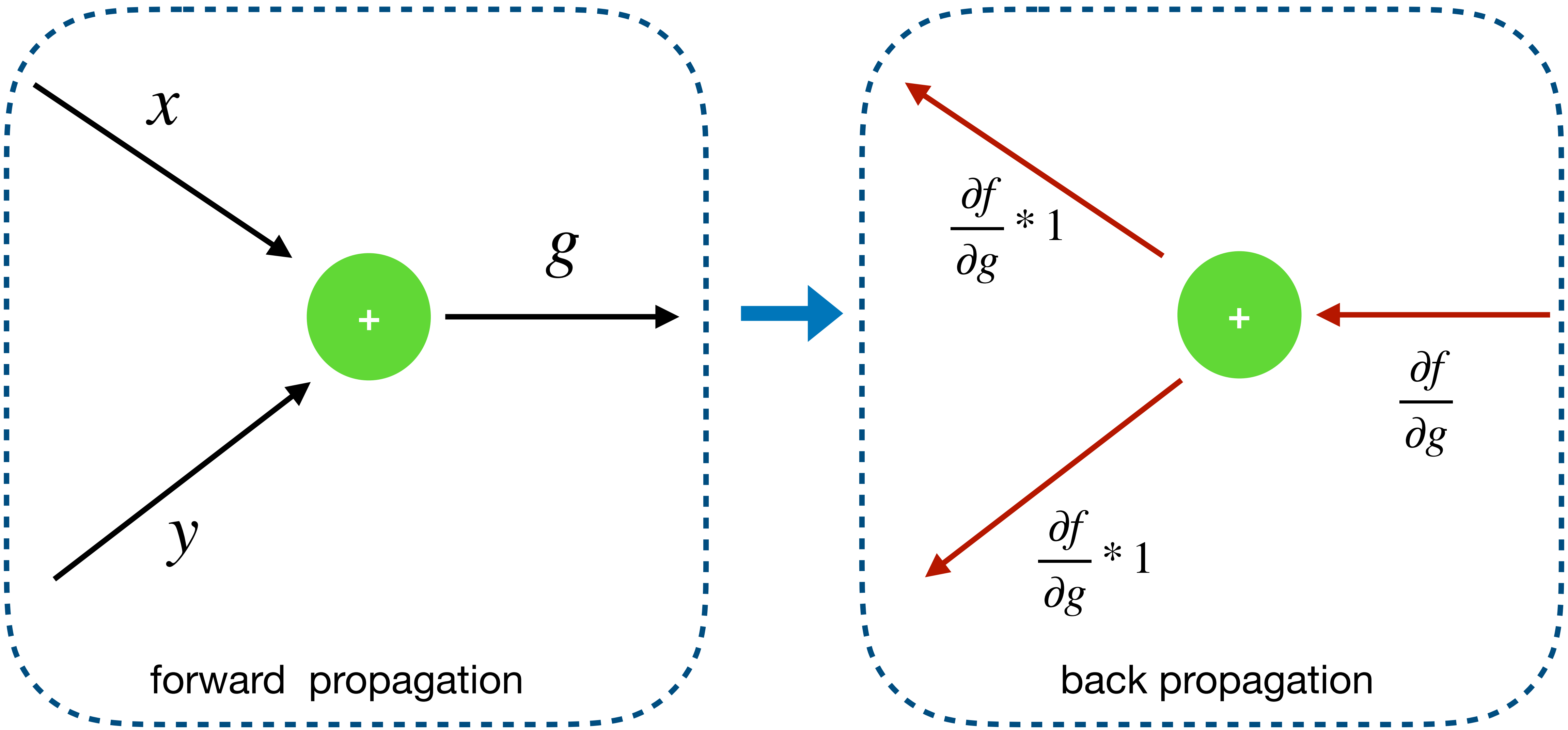


C. Back propagation

Back propagation of addition nodes

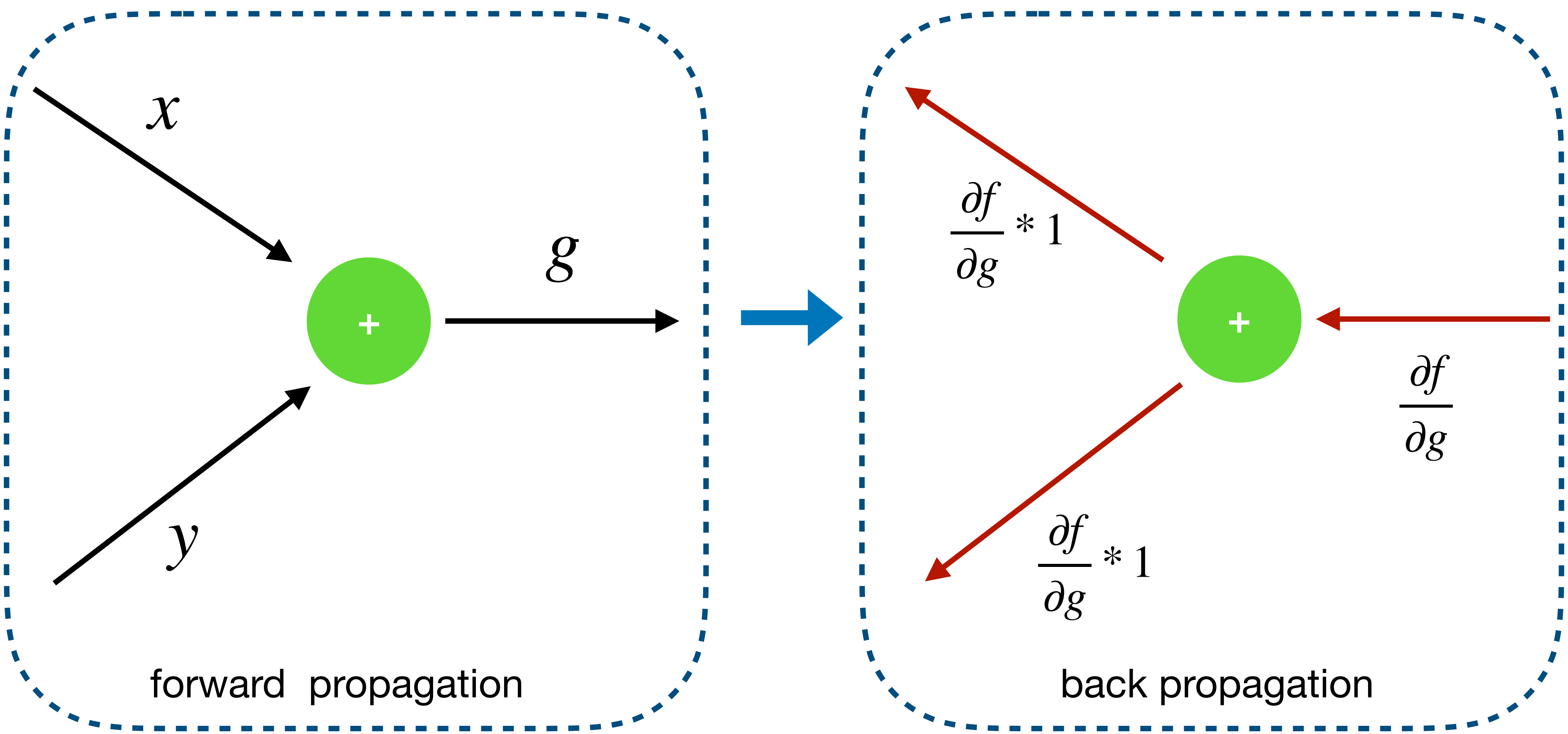
Partial derivatives of x and y

$$g = 2(x + y) \quad \rightarrow \quad \frac{\partial g}{\partial x} = 1, \quad \frac{\partial g}{\partial y} = 1$$



Back propagation of addition nodes

The back propagation of the addition node outputs the value of the upstream directly to the downstream



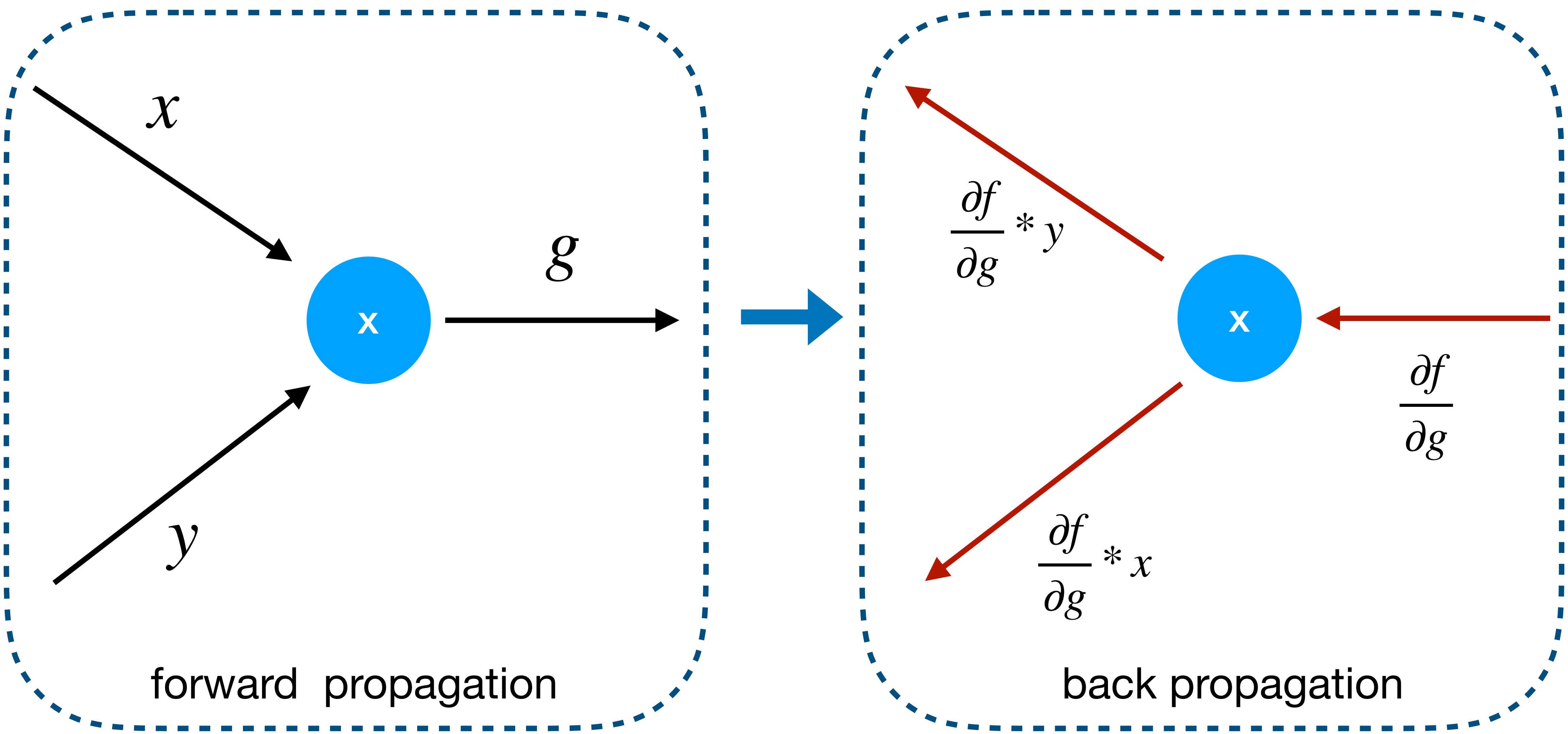
Code

```python
class AddLayer:
    def __init__(self):
        pass

    def forward(self, x, y):
        out = x + y

        return out

    def backward(self, dout):
        dx = dout * 1
        dy = dout * 1

        return dx, dy
```

Back propagation of multiplication nodes

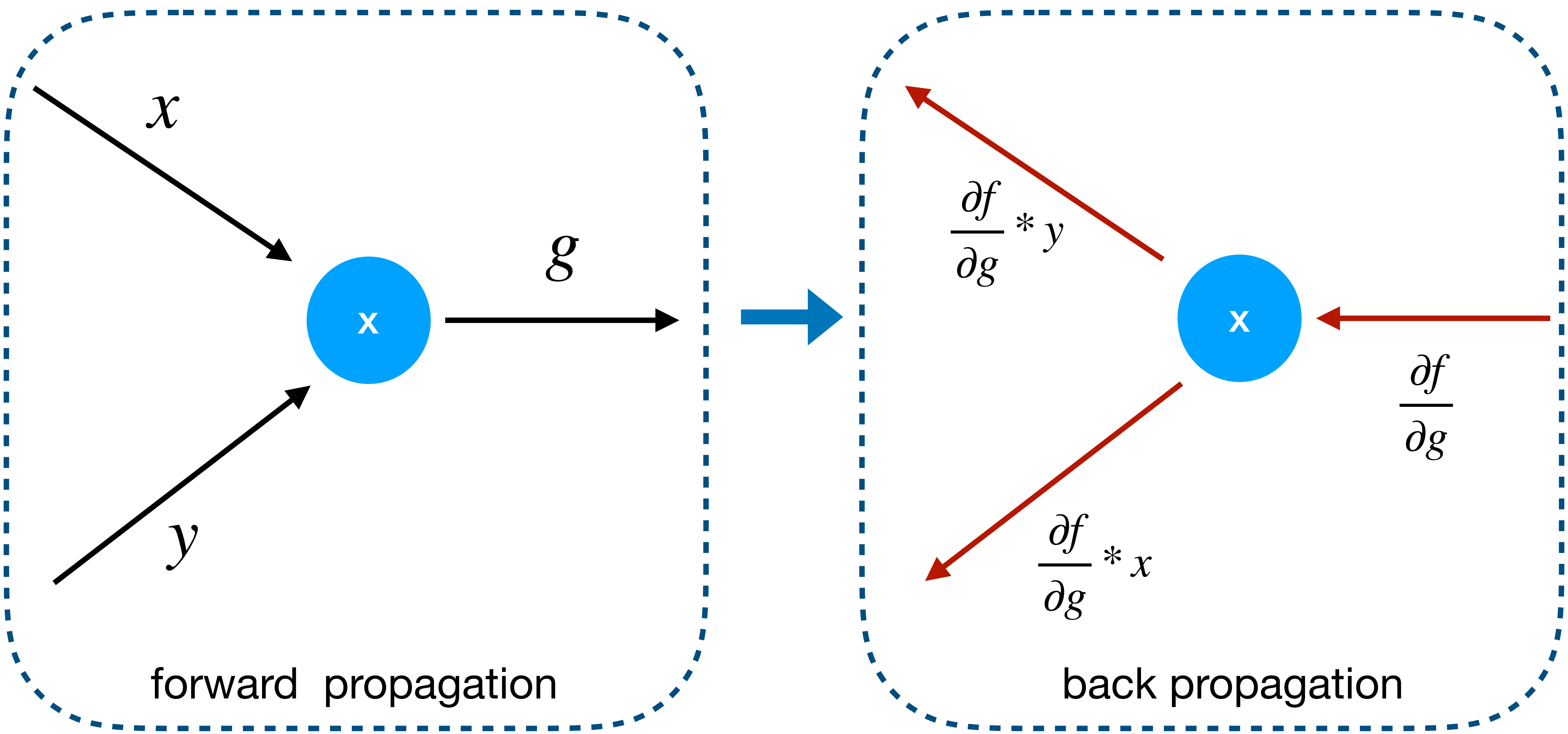
Partial derivatives of x and y

$$g = xy \quad \rightarrow \quad \frac{\partial g}{\partial x} = y, \quad \frac{\partial g}{\partial y} = x$$



Back propagation of multiplication nodes

The back propagation of the multiplication is multiplied by the flip value of the input signal



Code

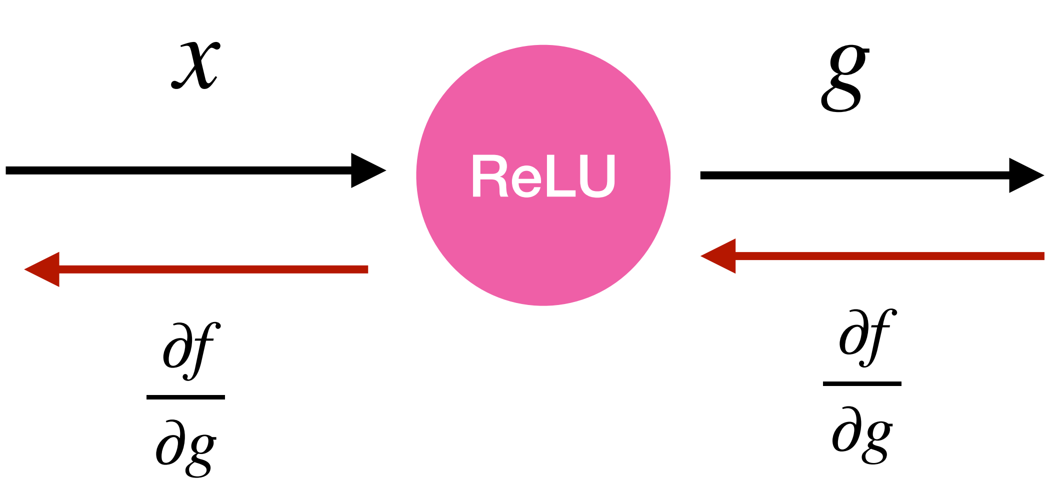
```
class MulLayer:
    def __init__(self):
        self.x = None
        self.y = None

    def forward(self, x, y):
        self.x = x
        self.y = y
        out = x * y

        return out

    def backward(self, dout):
        dx = dout * self.y
        dy = dout * self.x

        return dx, dy
```

Back propagation of ReLU

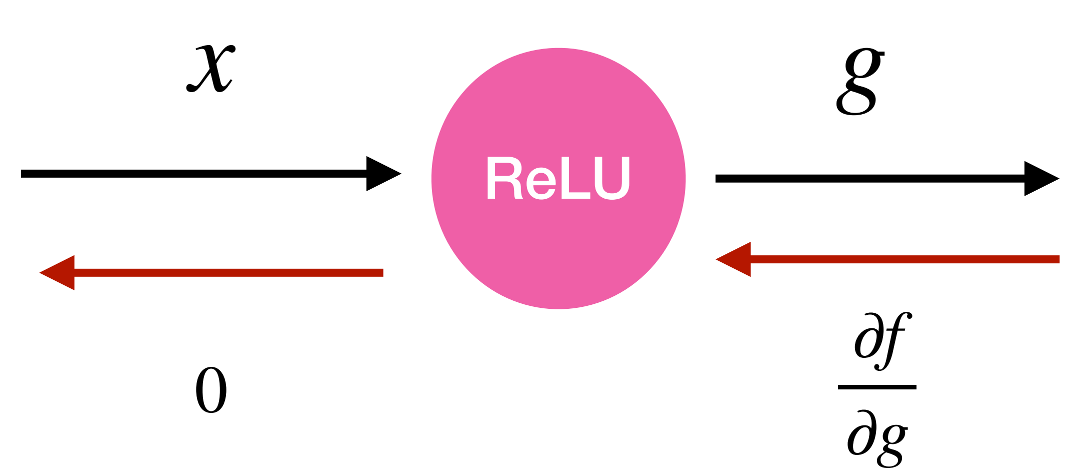
Partial derivatives of x

$$y = \begin{cases} x & (x > 0) \\ 0 & (x \leqslant 0) \end{cases} \quad \rightarrow \quad \frac{\partial y}{\partial x} = \begin{cases} 1 & (x > 0) \\ 0 & (x \leqslant 0) \end{cases}$$

$x > 0$



$x \leqslant 0$



Code

```python
class Relu:
    def __init__(self):
        self.mask = None

    def forward(self, x):
        self.mask = (x <= 0)
        out = x.copy()
        out[self.mask] = 0

        return out

    def backward(self, dout):
        dout[self.mask] = 0
        dx = dout

        return dx
```

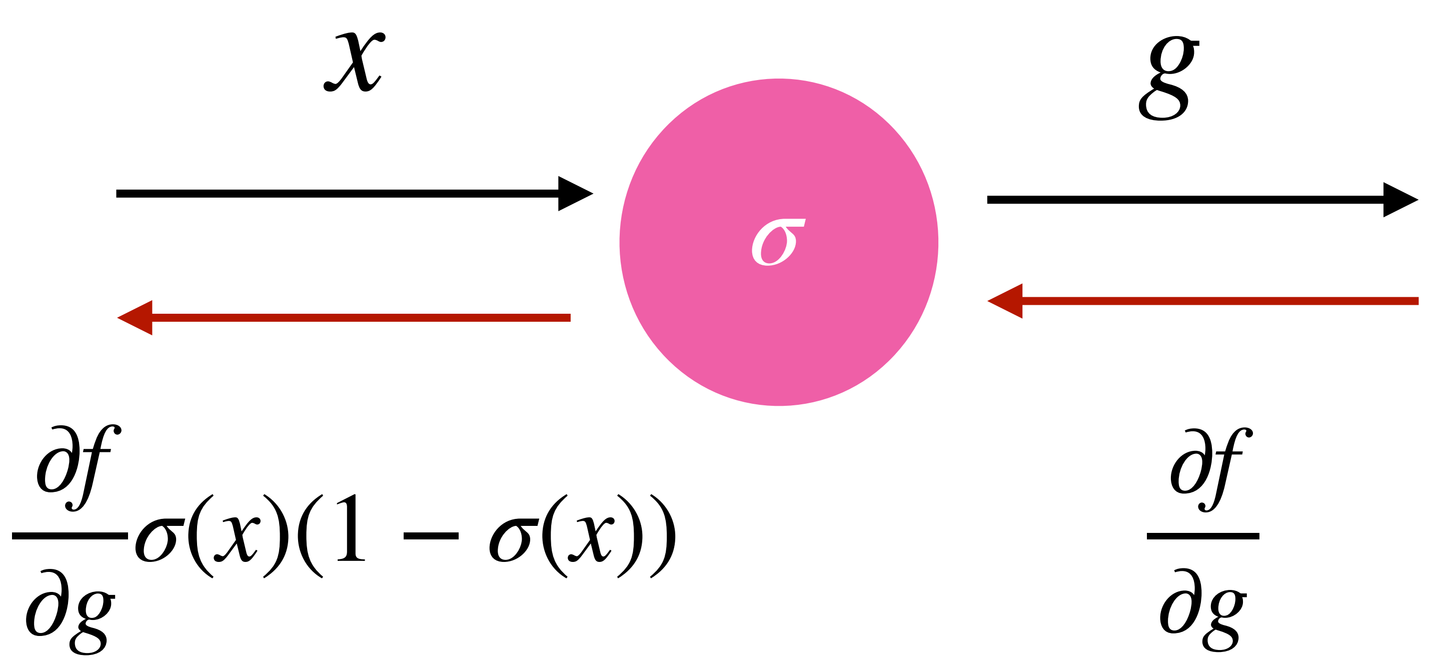
How mask work

```python
>>> import numpy as np
>>> x = np.array([1,2,-3,4,0])
>>> mask = (x<=0)
>>> mask
array([False, False,  True, False,  True])
>>> dout = np.array([1,-3,0,5,-2])
>>> dout[mask] = 0
>>> dout
array([ 1, -3,  0,  5,  0])
```

Back propagation of Sigmoid

Partial derivatives of x

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad \Rightarrow \quad \frac{\partial \sigma(x)}{\partial x} = \sigma(x)(1 - \sigma(x))$$



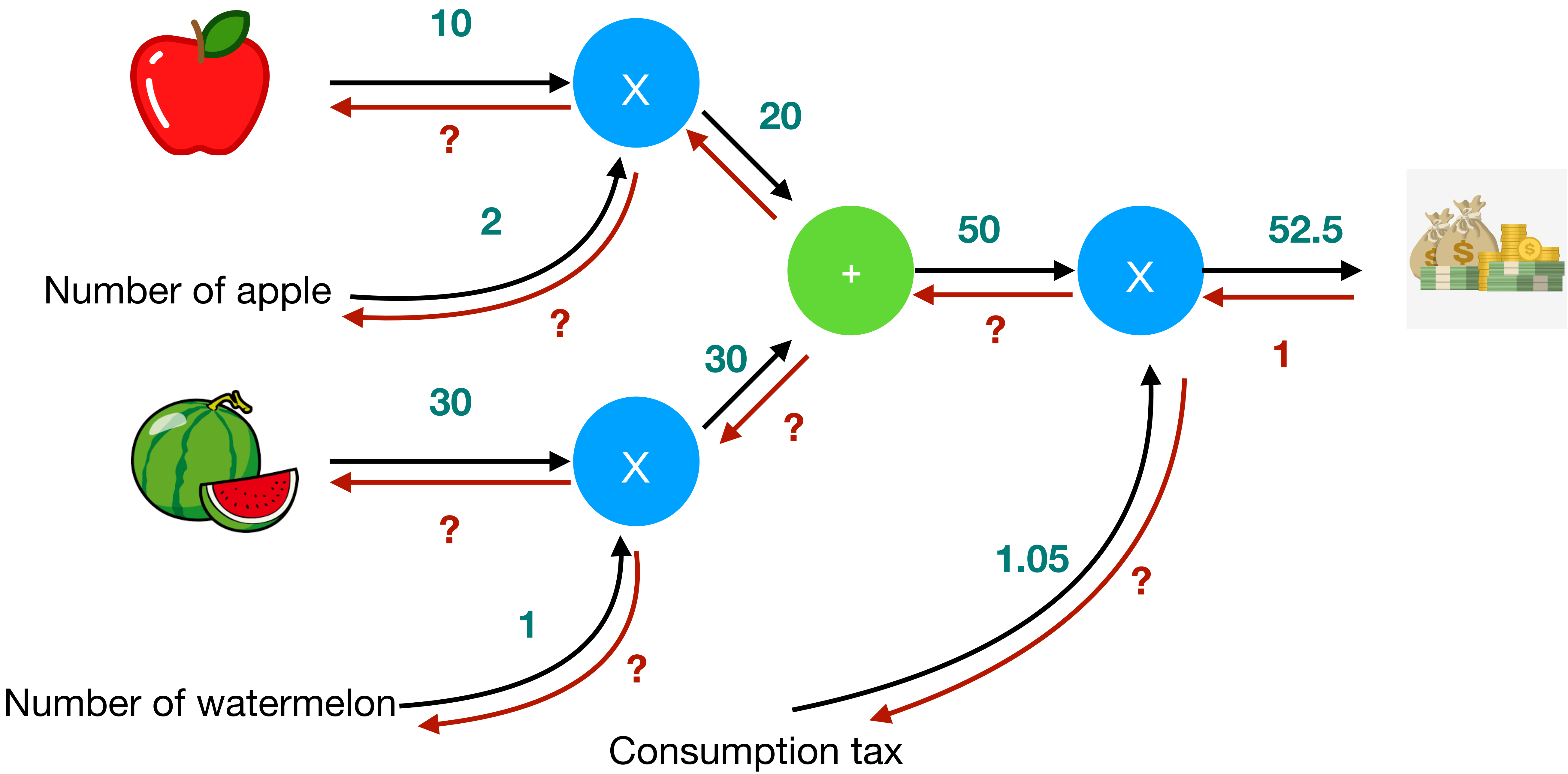
Code

```python
class Sigmoid:
    def __init__(self):
        self.out = None

    def forward(self, x):
        out = sigmoid(x)
        self.out = out
        return out

    def backward(self, dout):
        dx = dout * (1.0 - self.out) * self.out

        return dx
```

D.Application

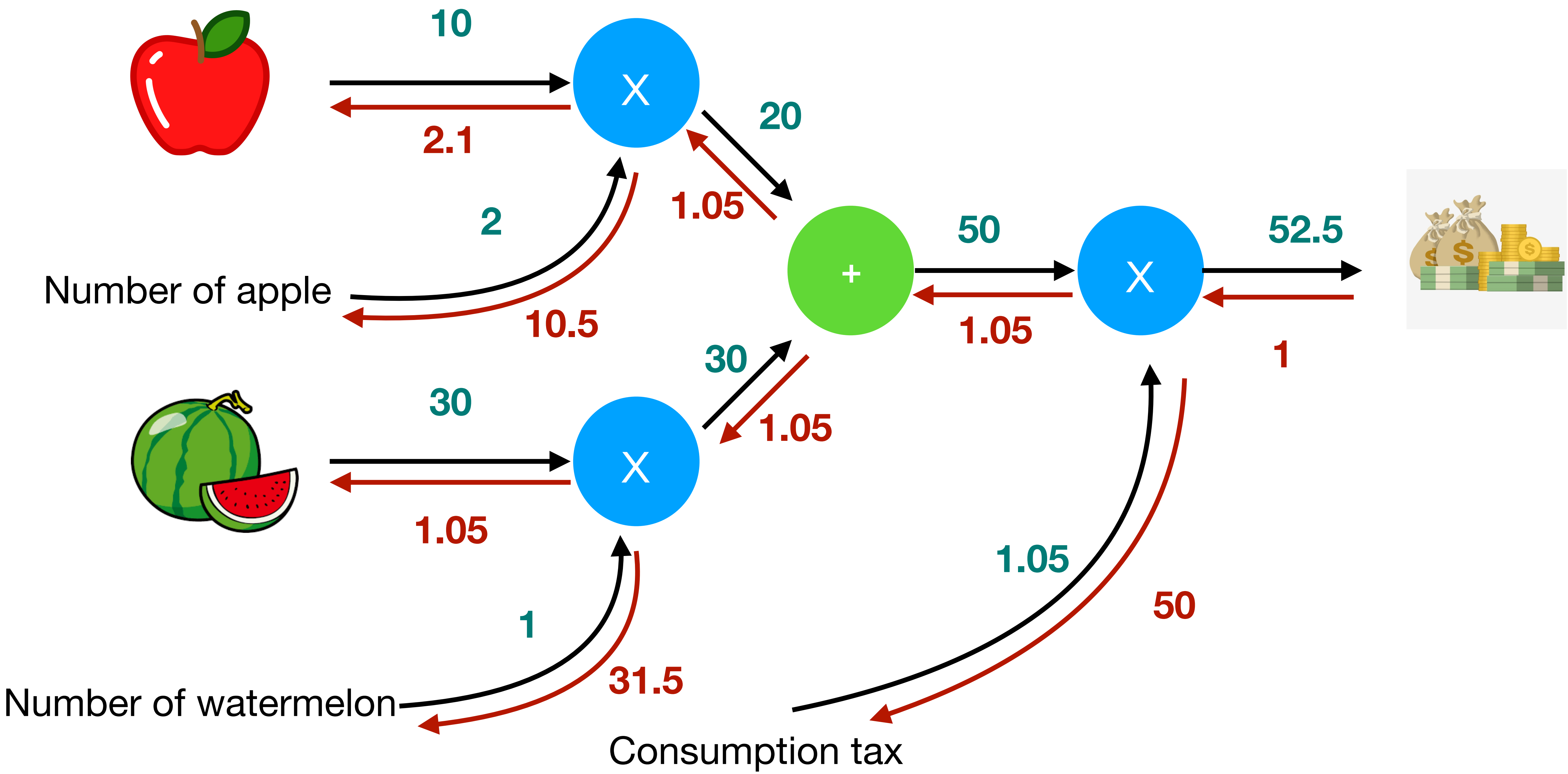
Exercise 1

Using compute graph to calculate gradient. (5-10 minutes)



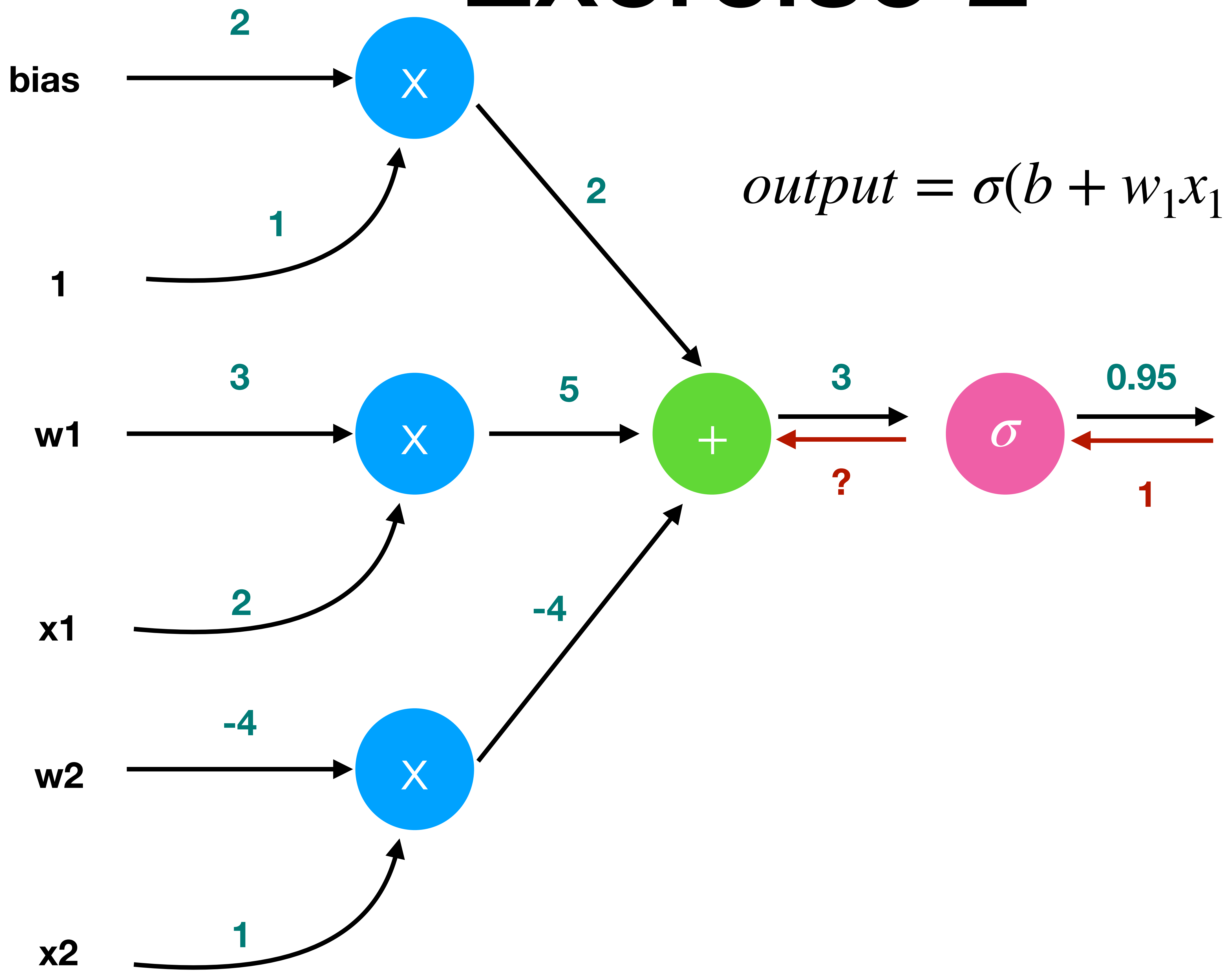
Exercise 1

Using compute graph to calculate gradient. (5-10minutes)

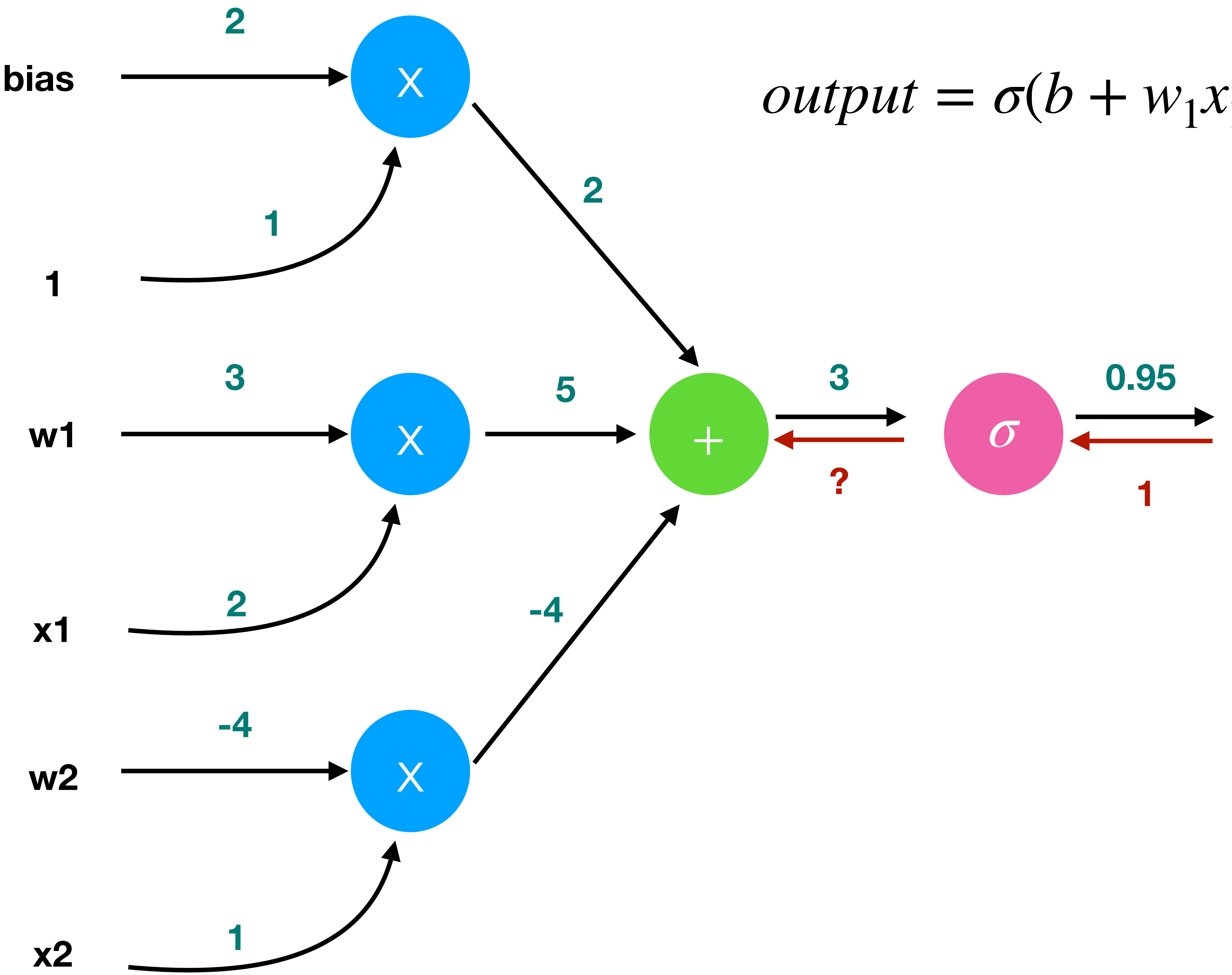


Exercise 2

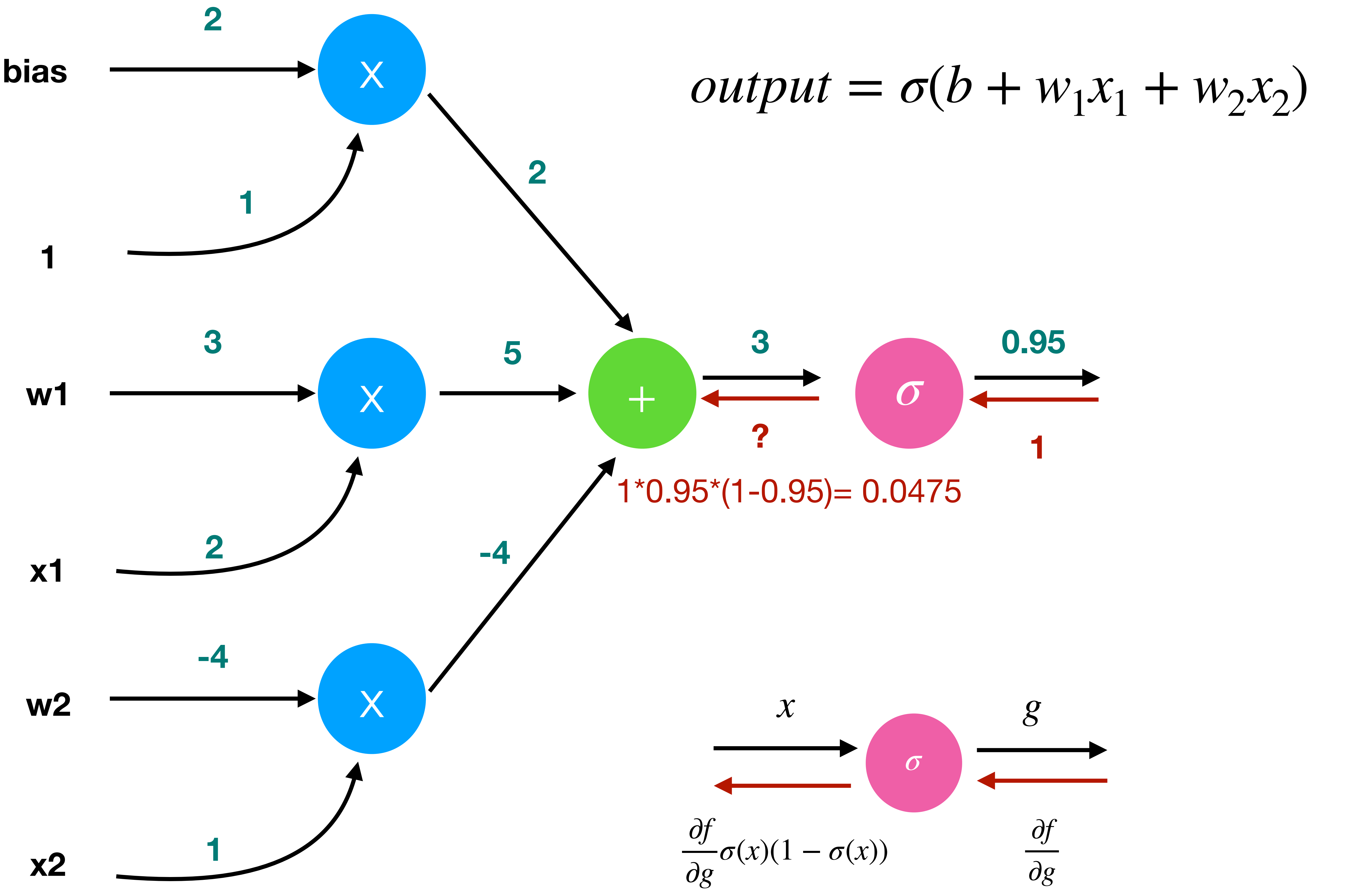
$$output = \sigma(b + w_1x_1 + w_2x_2)$$

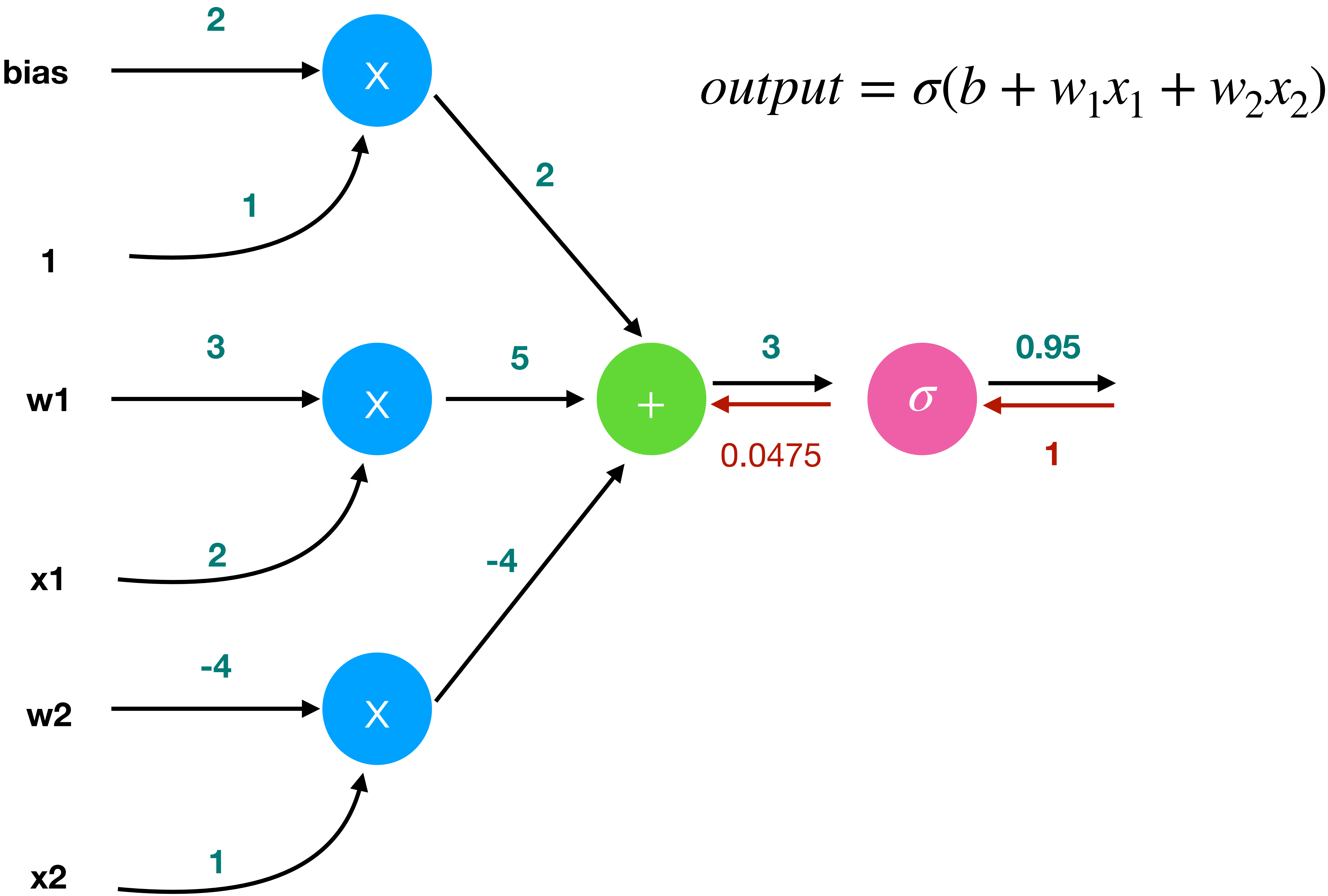


$$output = \sigma(b + w_1x_1 + w_2x_2)$$



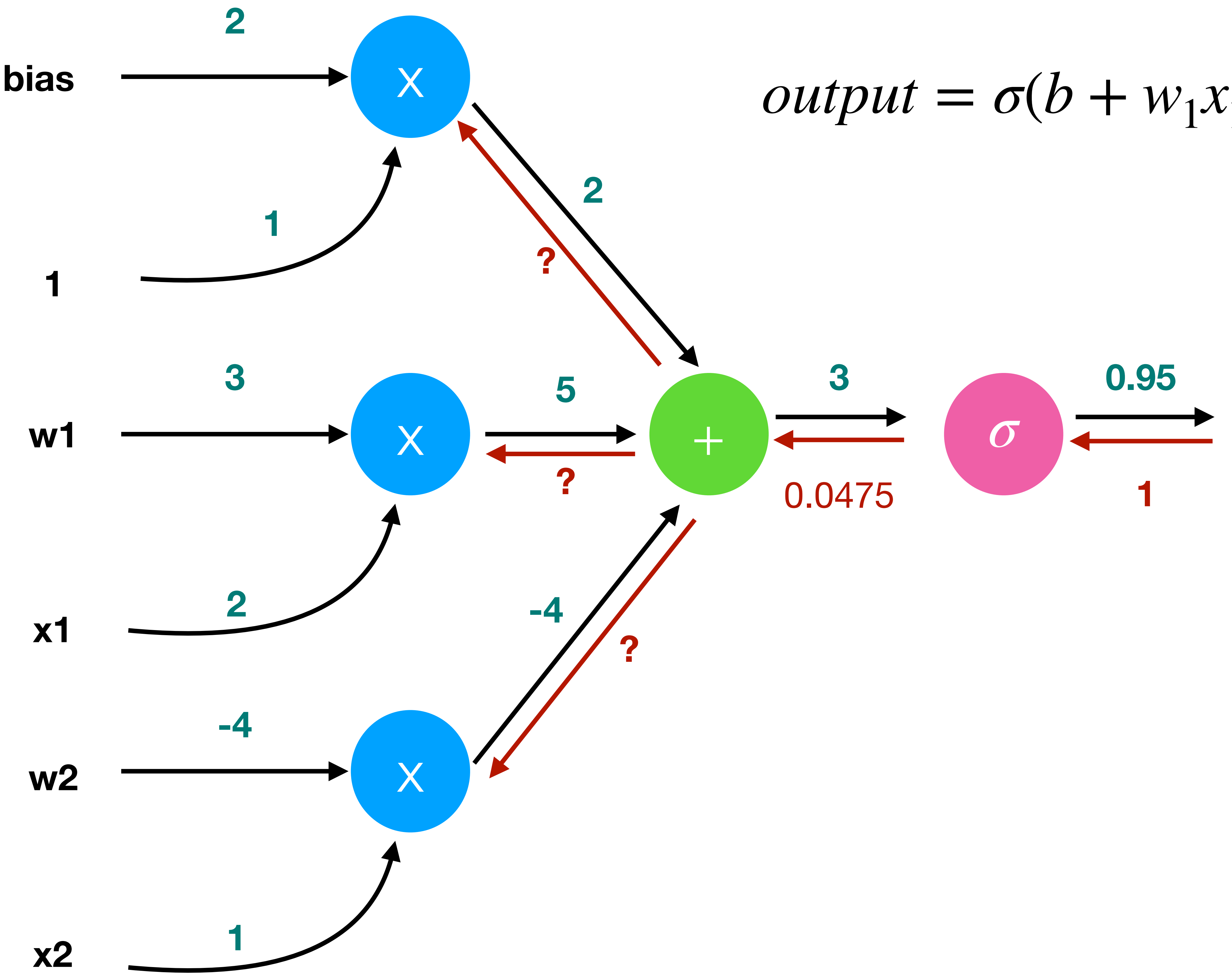
$$output = \sigma(b + w_1 x_1 + w_2 x_2)$$

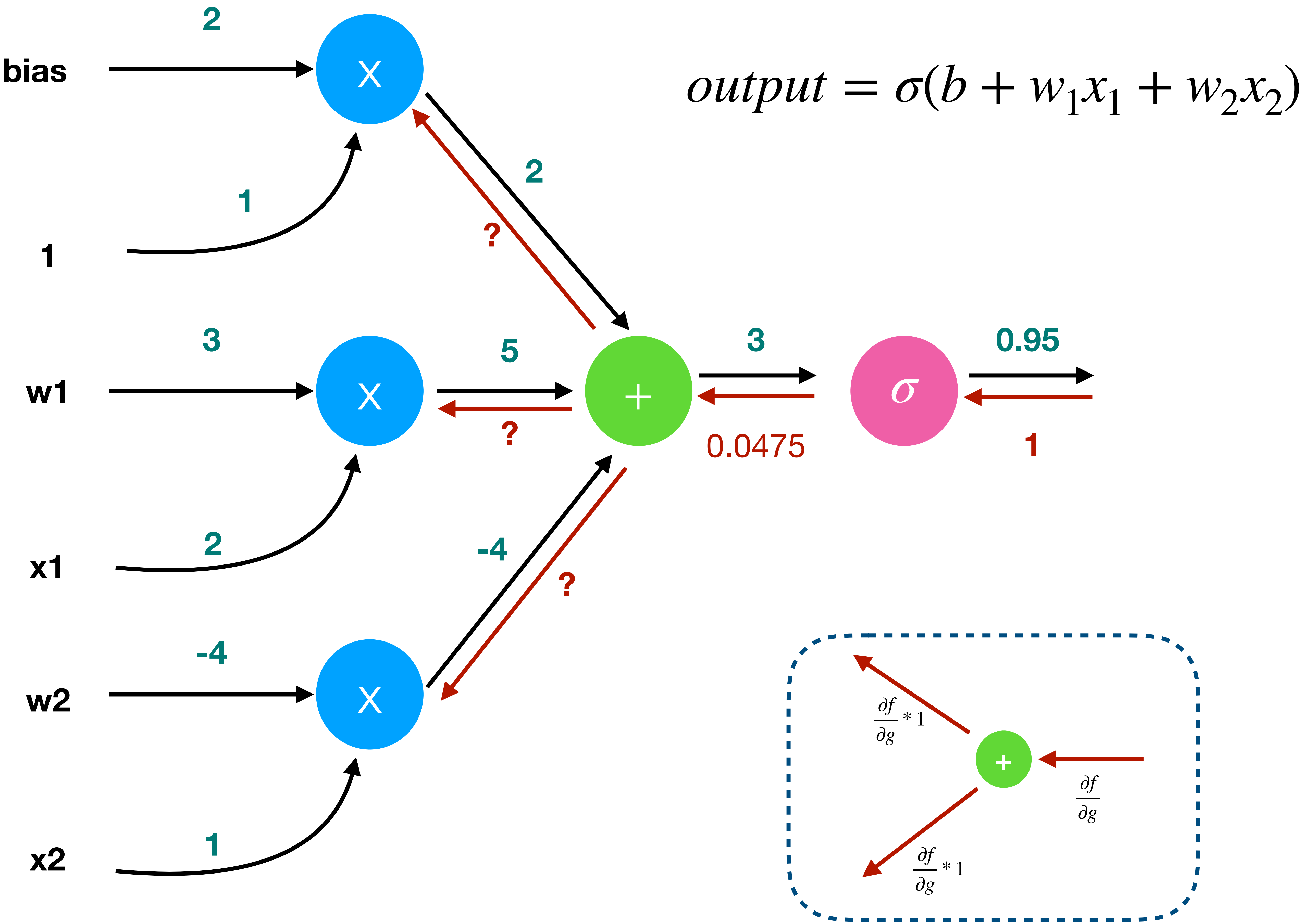




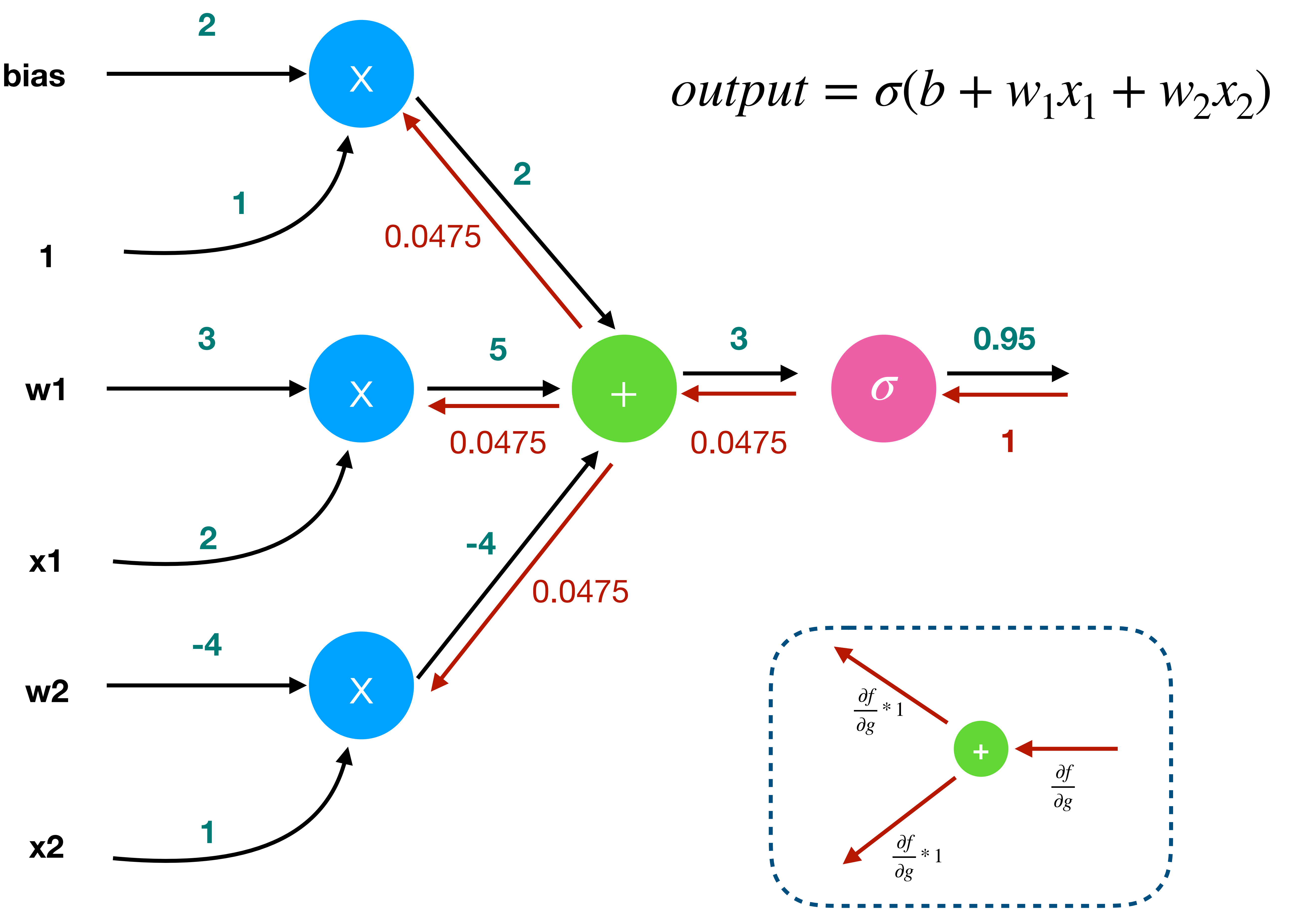
bias
1
w1
x1
w2
x2
2
1
3
2
-4
1
X
X
X
2
5
-4
+
3
0.0475
σ
0.95
1
$output = \sigma(b + w_1x_1 + w_2x_2)$

$$output = \sigma(b + w_1x_1 + w_2x_2)$$



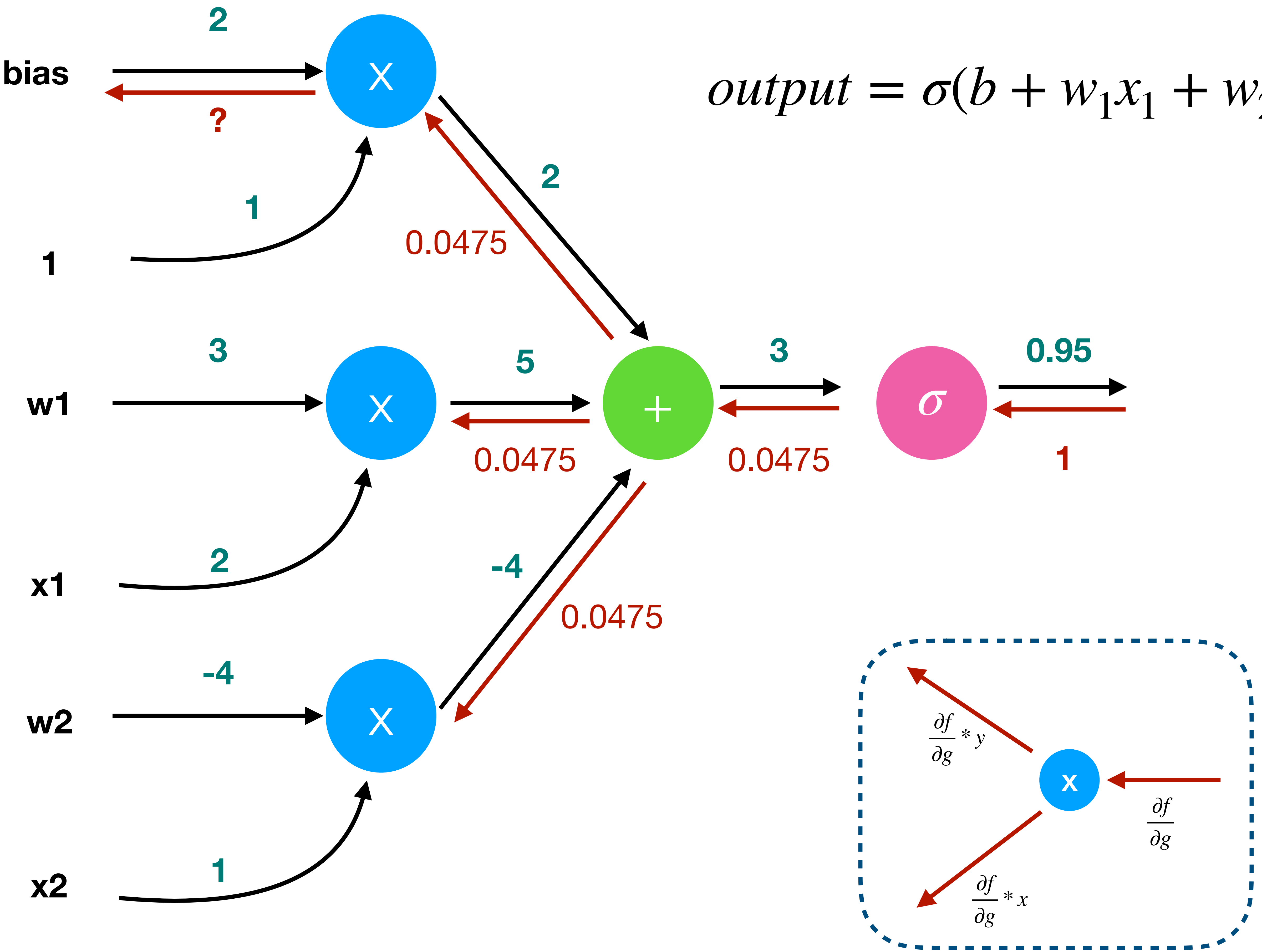


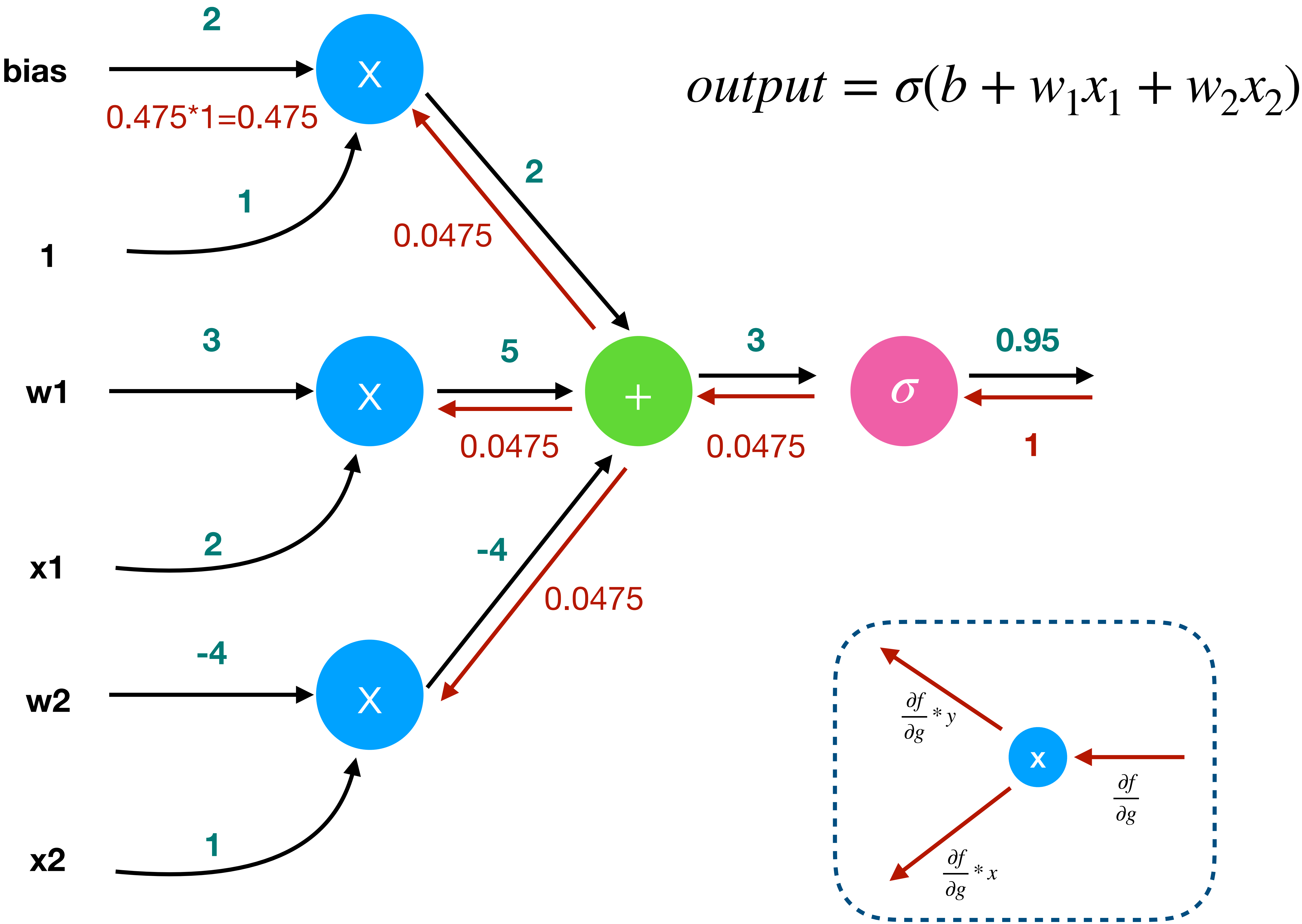
$output = \sigma(b + w_1x_1 + w_2x_2)$
bias
1
w1
x1
w2
x2
2
1
3
2
-4
1
X
X
X
2
5
-4
?
?
?
+
3
0.0475
σ
0.95
1
+
$\frac{\partial f}{\partial g} * 1$
$\frac{\partial f}{\partial g} * 1$
$\frac{\partial f}{\partial g}$



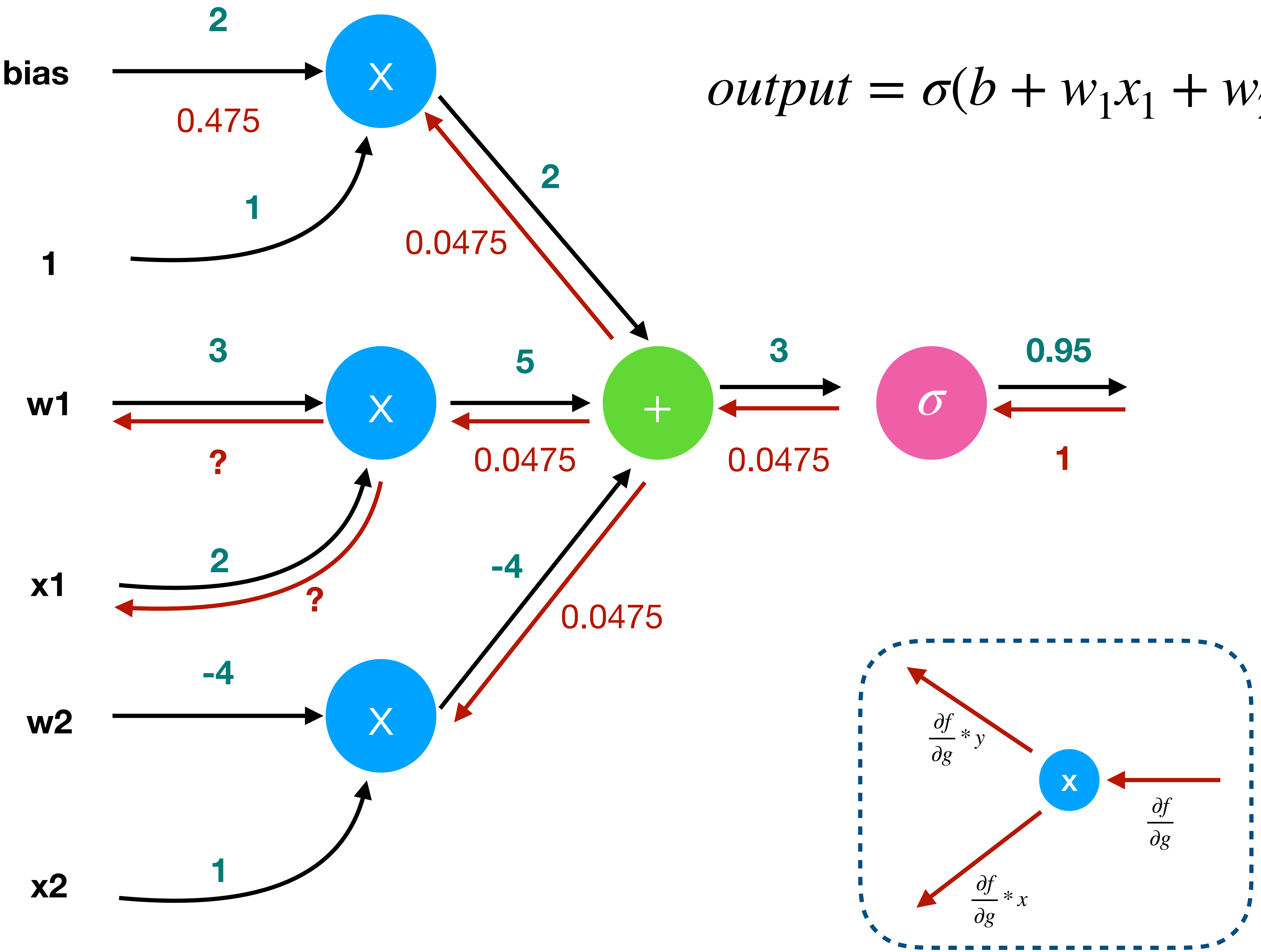
bias
1
w1
x1
w2
x2
2
1
3
2
-4
1
x
x
x
2
5
-4
0.0475
0.0475
0.0475
+
3
0.0475
σ
0.95
1
$output = \sigma(b + w_1x_1 + w_2x_2)$
$\frac{\partial f}{\partial g} * 1$
+
$\frac{\partial f}{\partial g}$
$\frac{\partial f}{\partial g} * 1$

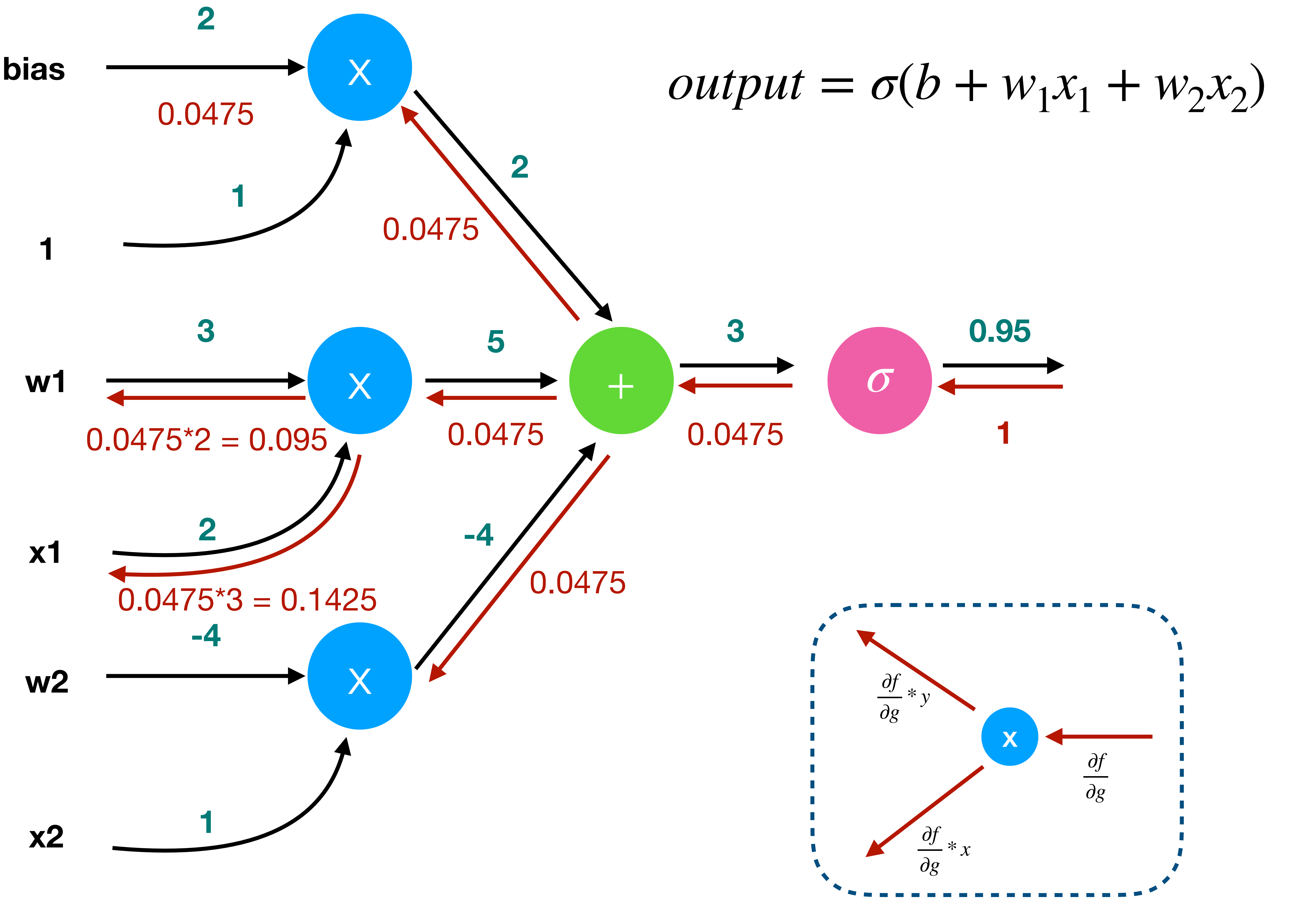
$$output = \sigma(b + w_1x_1 + w_2x_2)$$




$output = \sigma(b + w_1x_1 + w_2x_2)$
bias
1
w1
x1
w2
x2
2
1
3
2
-4
1
0.475*1=0.475
2
0.0475
5
0.0475
-4
0.0475
3
0.0475
0.95
1
$\frac{\partial f}{\partial g} * y$
$\frac{\partial f}{\partial g}$
$\frac{\partial f}{\partial g} * x$

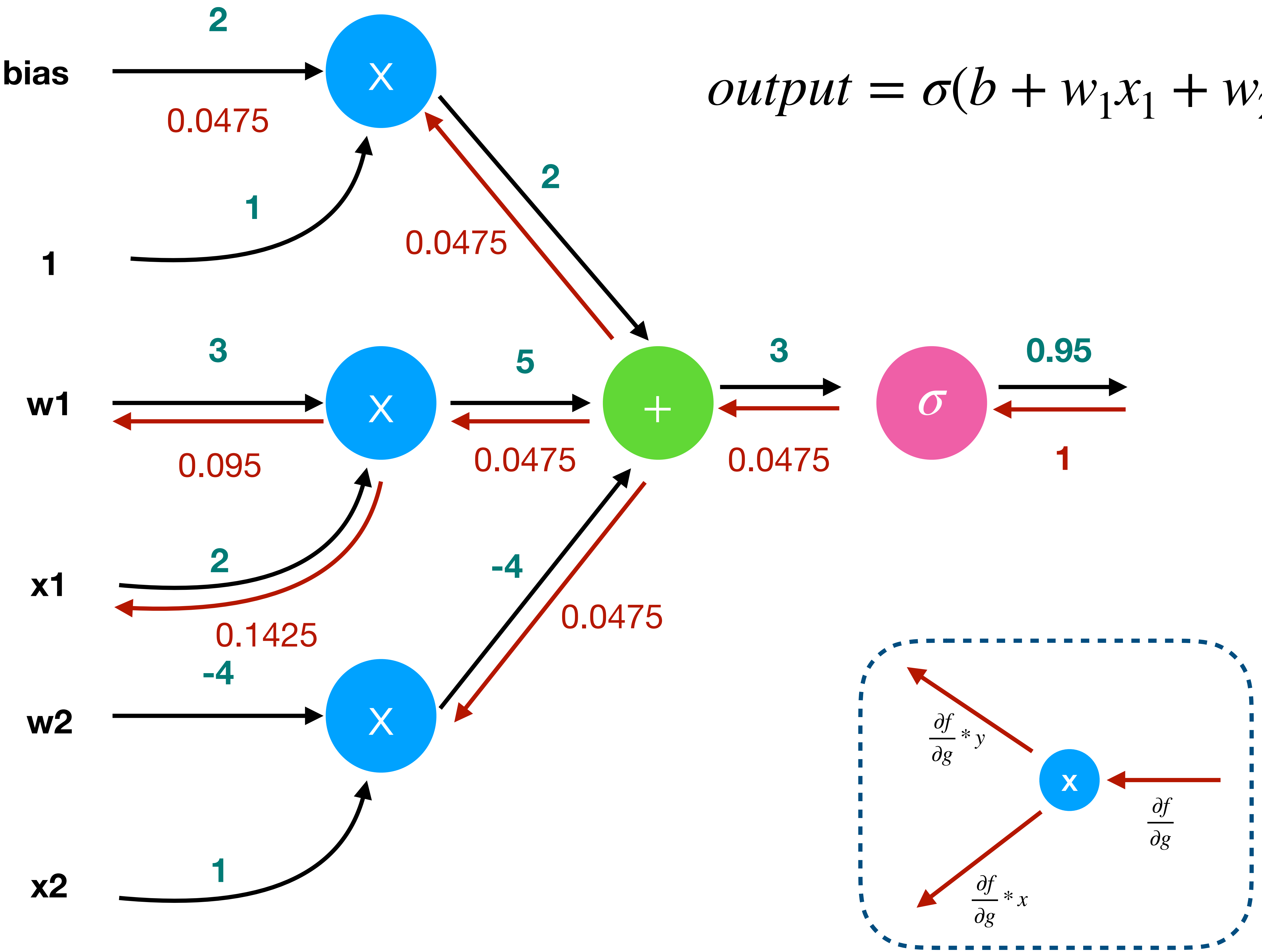
$$output = \sigma(b + w_1x_1 + w_2x_2)$$



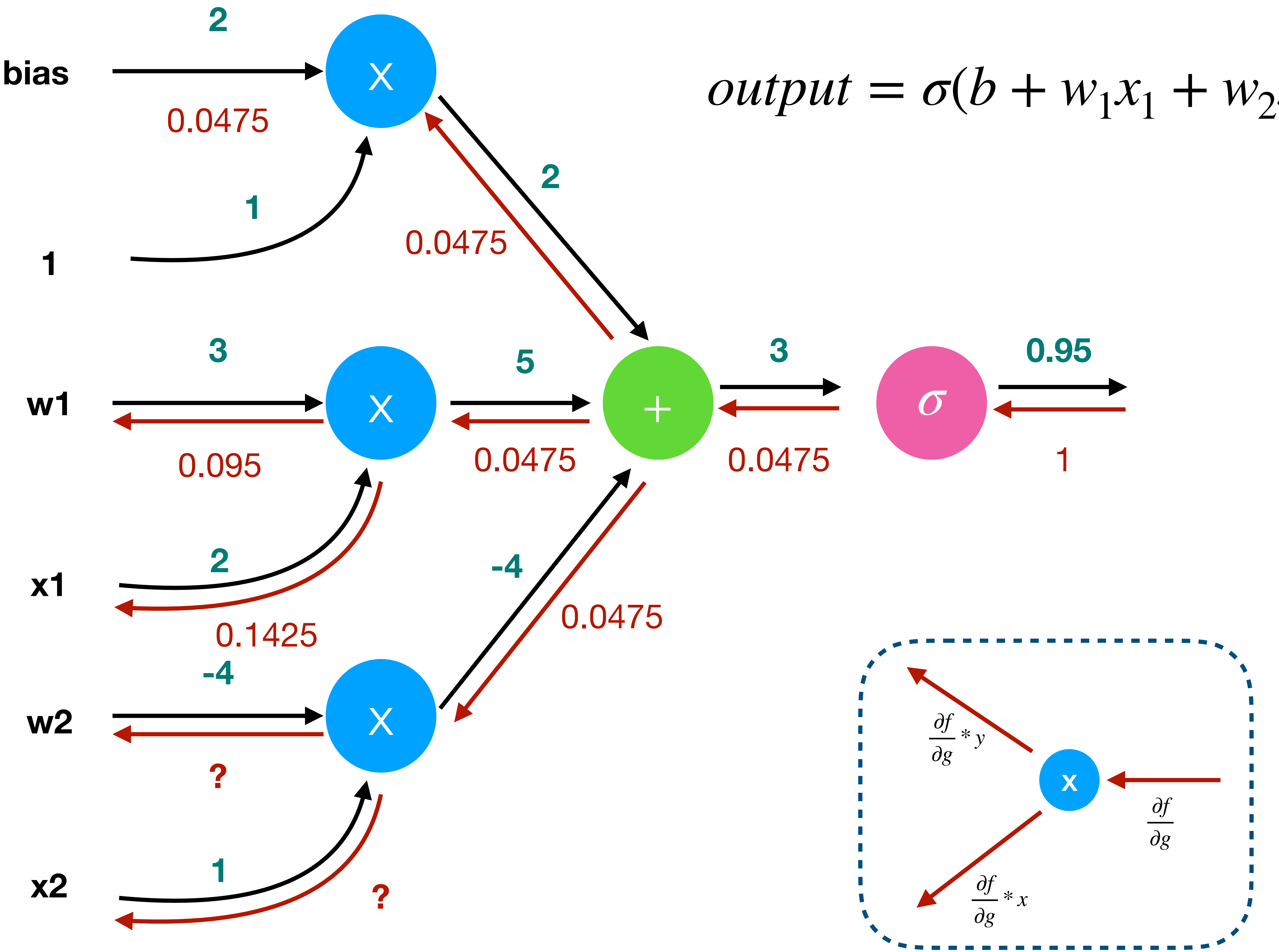


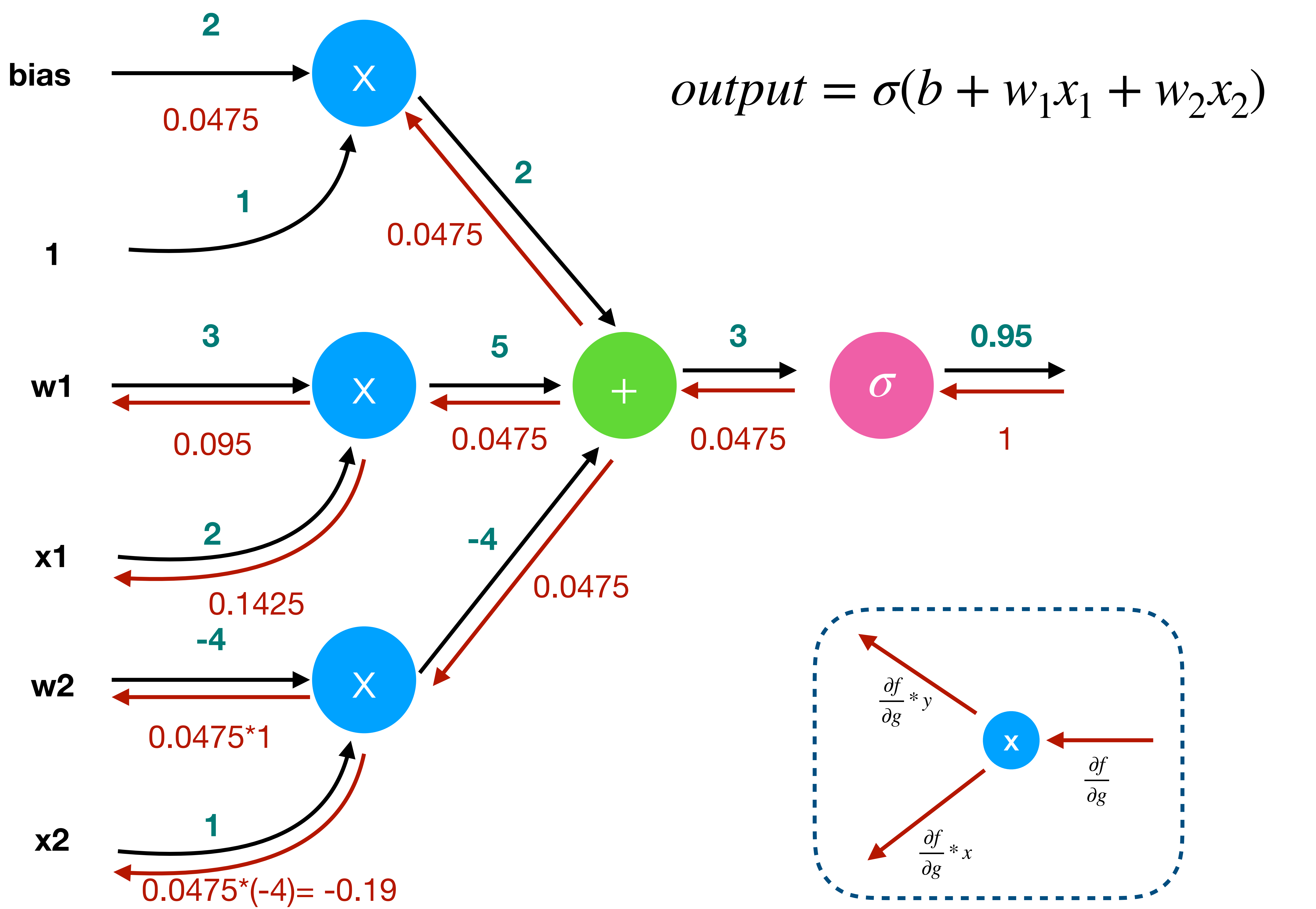
bias
1
w1
x1
w2
x2
2
0.0475
1
2
0.0475
3
5
0.0475*2 = 0.095
0.0475
2
0.0475*3 = 0.1425
-4
-4
0.0475
3
0.0475
0.95
1
$output = \sigma(b + w_1x_1 + w_2x_2)$
$\frac{\partial f}{\partial g} * y$
$\frac{\partial f}{\partial g}$
$\frac{\partial f}{\partial g} * x$

$$output = \sigma(b + w_1 x_1 + w_2 x_2)$$



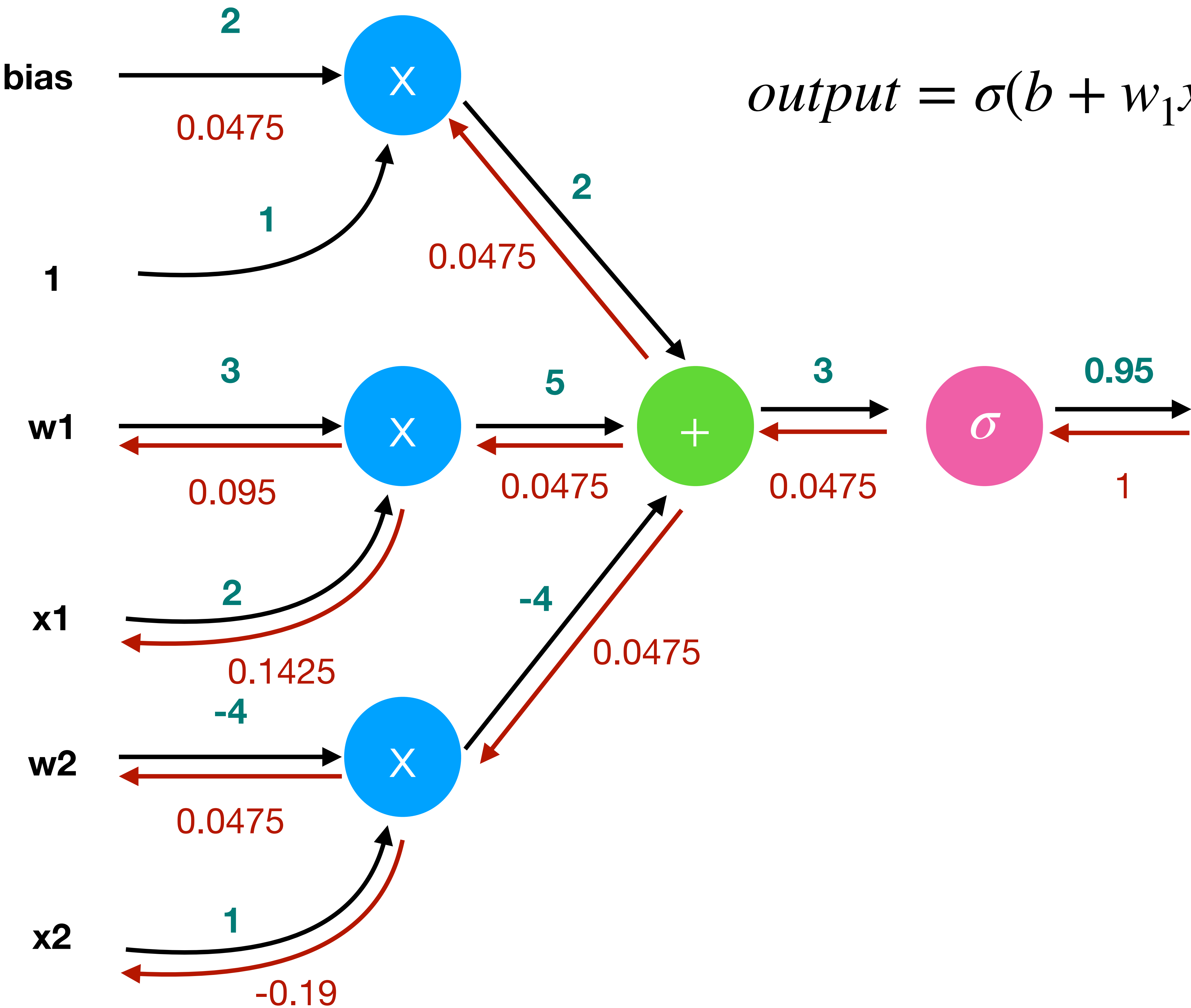
$$output = \sigma(b + w_1 x_1 + w_2 x_2)$$

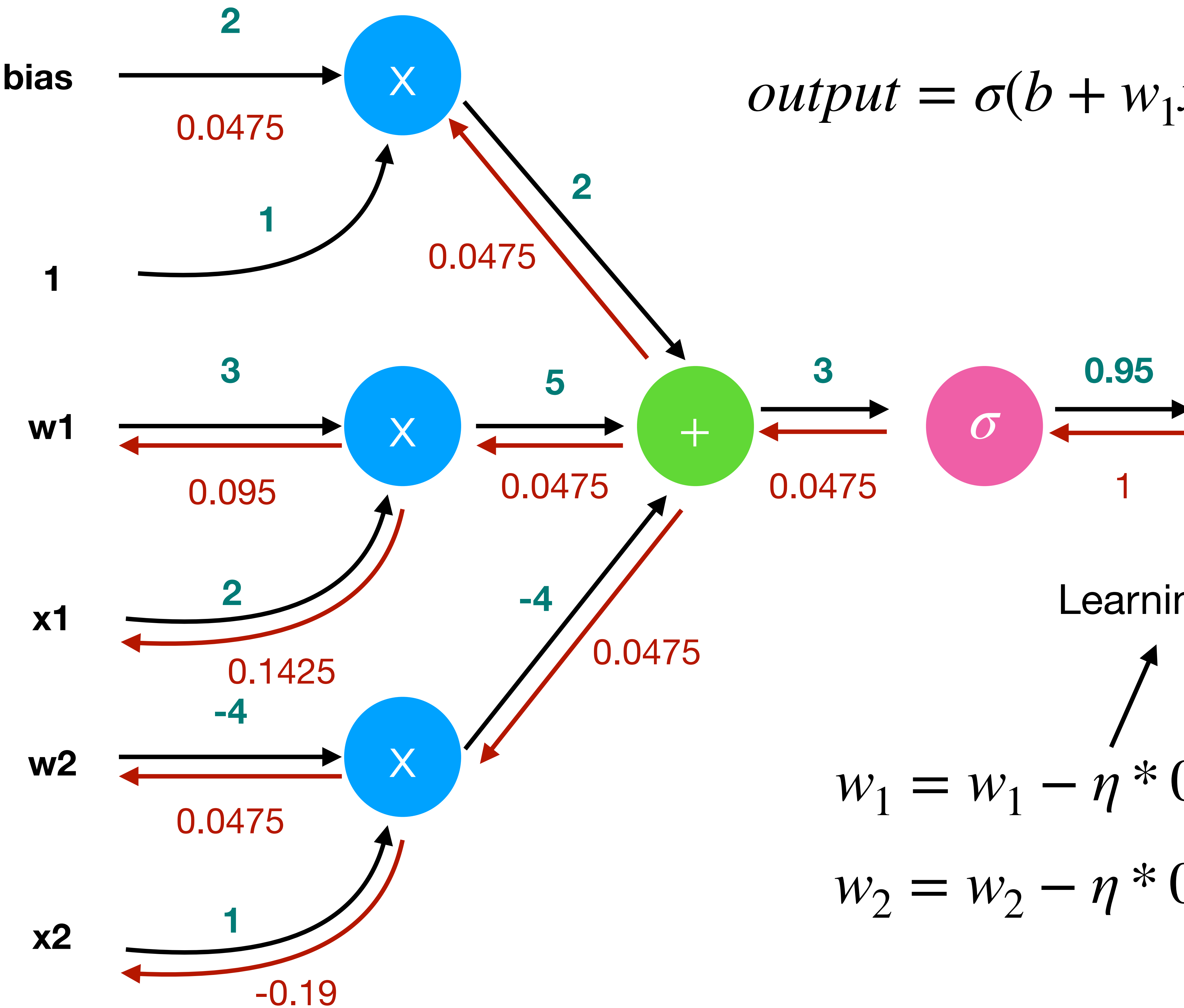




$output = \sigma(b + w_1x_1 + w_2x_2)$
bias
2
0.0475
1
1
2
0.0475
w1
3
0.095
5
0.0475
3
0.0475
0.95
1
x1
2
0.1425
-4
0.0475
w2
-4
0.0475*1
x2
1
0.0475*(-4)= -0.19
$\frac{\partial f}{\partial g} * y$
$\frac{\partial f}{\partial g}$
$\frac{\partial f}{\partial g} * x$

$$output = \sigma(b + w_1x_1 + w_2x_2)$$





$$output = \sigma(b + w_1x_1 + w_2x_2)$$

Learning rate

$$w_1 = w_1 - \eta * 0.095$$

$$w_2 = w_2 - \eta * 0.0475$$

Summary

- Step 0. Init weights and bias
- Step 1. Forward propagation
- Step 2. Compute Loss
- Step 3. Compute gradient using back propagation
- Step 4. Update weights and bias
- Repeat Step 1- Step 4 several times.

Thanks!