IBM Research Data Science Course

srihari sridharan

Agenda

More Neural Network

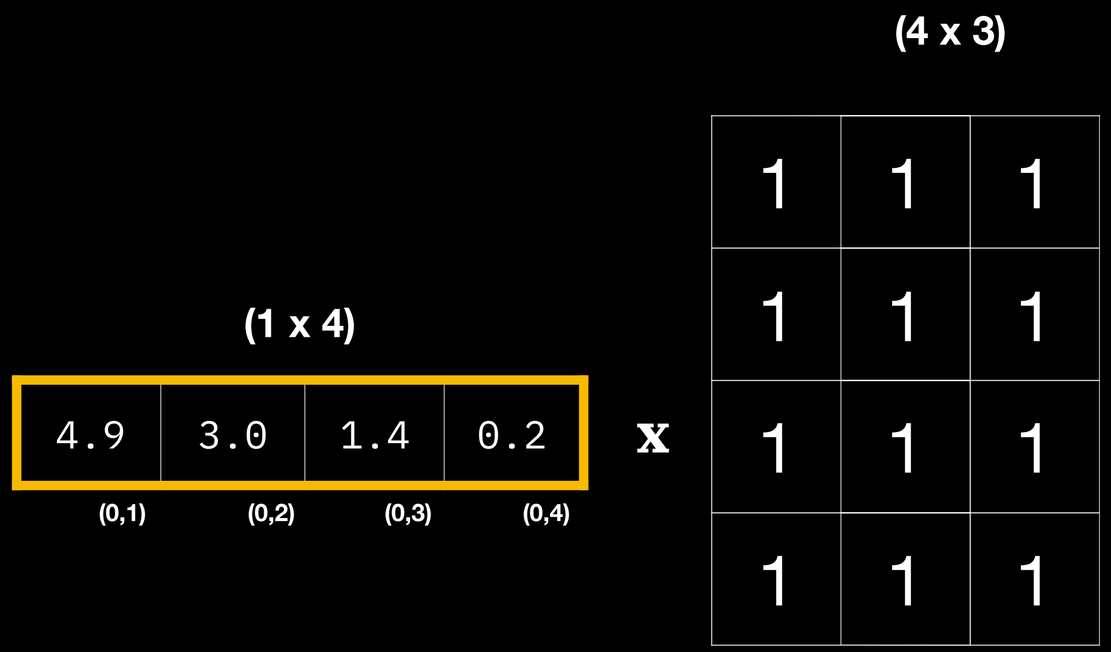
- Loss
- Gradient Descent
- Backpropagation
- Computational Graphs (Tensorflow 101)

Training Process (Backpropagation)

Step 1 : Initially weights and bias are randomly assigned.
Step 2 : Vector -> Matrix Multiplication -> Some Output (Vector)
Step 3 : Compare with the actual value (Label)
Step 4 : See the difference and pass it back to all the layers behind it.
Step 5 : Update the weights and repeat from Step 2

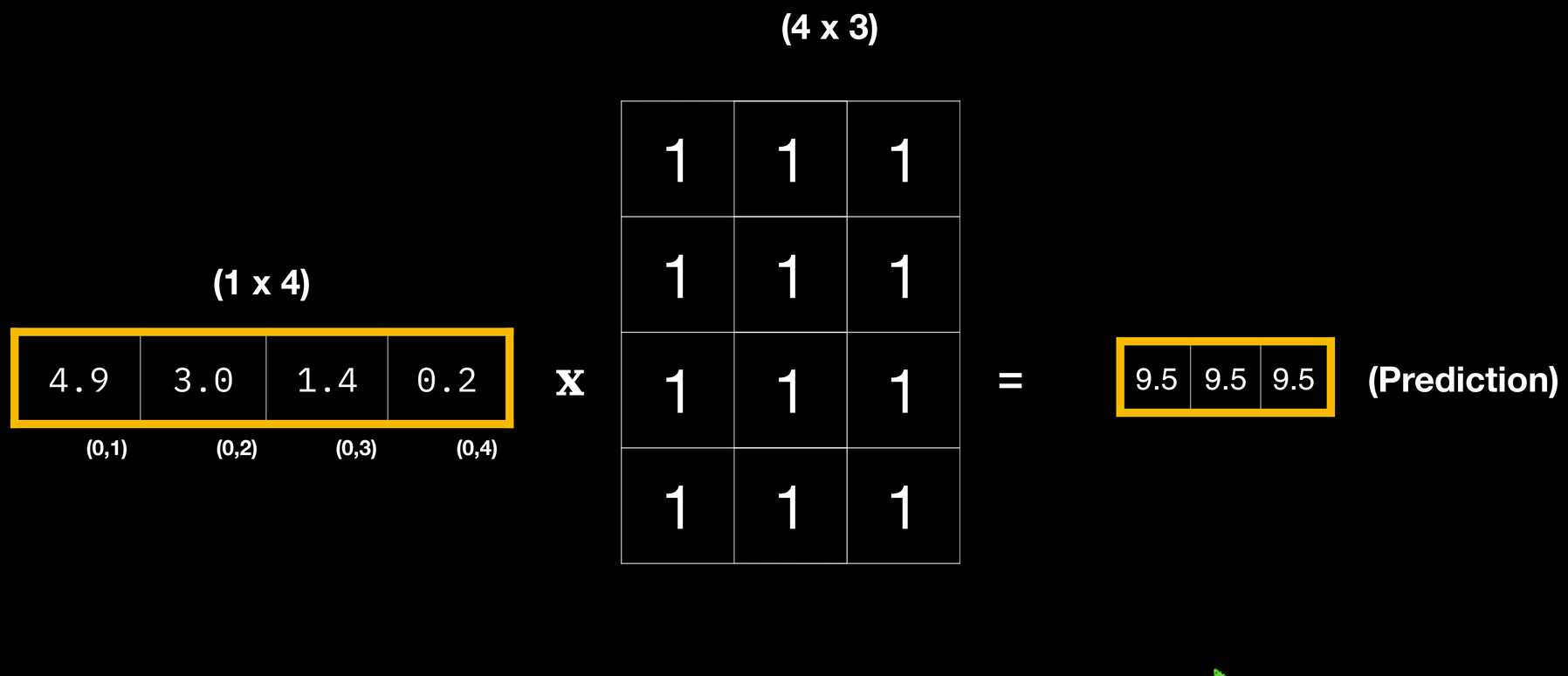
Training Process (Finding the values of W,b)

Step 1 : Initialise the weights to something



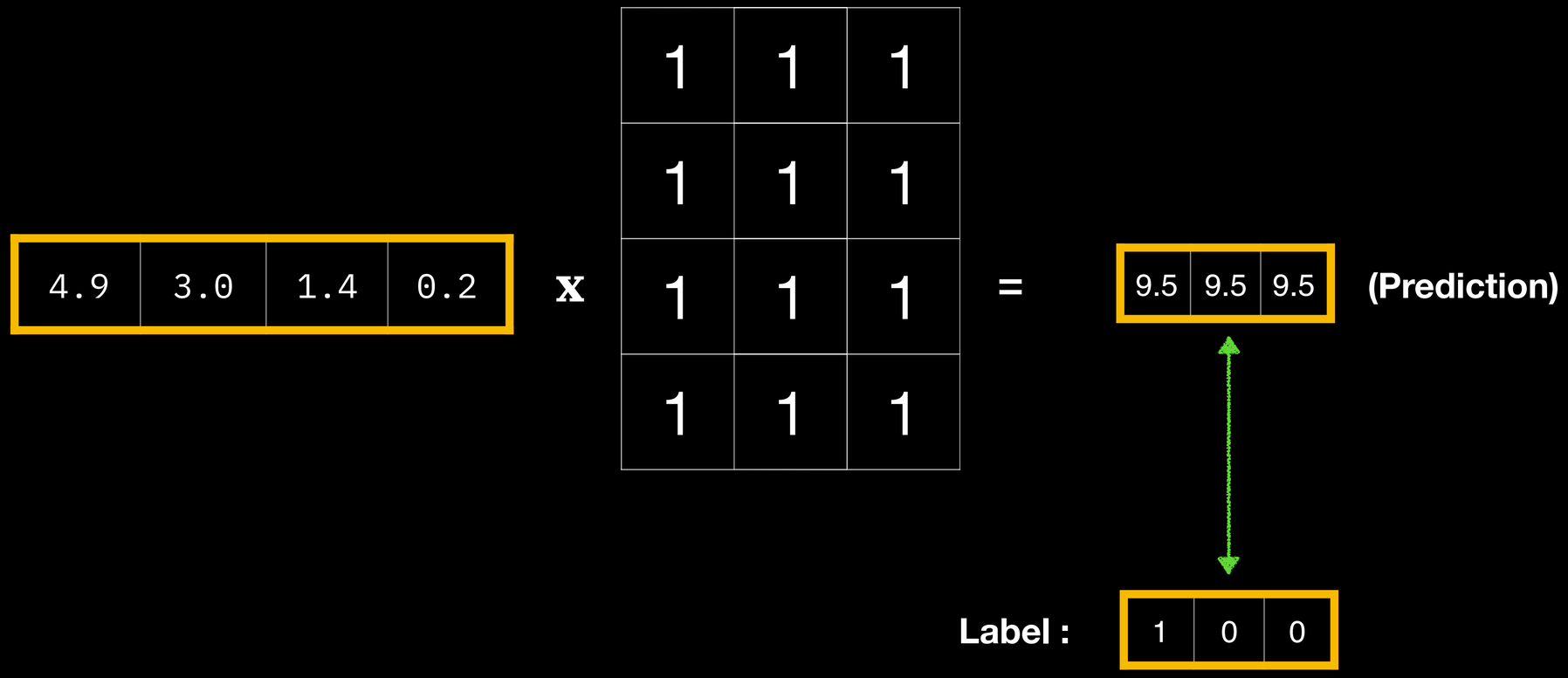
Training Process (Finding the values of W,b)

Step 2 : Do the matrix multiplication with initialised weights.



Training Process (Finding the values of W,b)

Step 3 : Compare with actual value (Label)



Training Process (Finding the values of W,b)

Step 4 : See the difference and pass back the changes (gradients) to previous layers

4.9	3.0	1.4	0.2

x

1	1	1
1	1	1
1	1	1
1	1	1

=

9.5	9.5	9.5

Square Error : Square of difference.

$$E_{total} = \sum \frac{1}{2}(target - output)^2$$

= 84.25

Label :

1	0	0

Training Process (Finding the values of W,b)

Step 4 : See the difference and pass back the changes (gradients) to previous layers

4.9	3.0	1.4	0.2

x

1	1	1
1	1	1
1	1	1
1	1	1

=

9.5	9.5	9.5

Error = 84.25

Label :

1	0	0

$$E_{total} = \sum \frac{1}{2}(target - output)^2$$

$$\frac{\partial E_{total}}{\partial out_{o1}} = 2 * \frac{1}{2}(target_{o1} - out_{o1})^{2-1} * -1 + 0$$

$$\frac{\partial E_{total}}{\partial out_{o1}} = -(target_{o1} - out_{o1})$$

Gradients (Derivatives)

Training Process (Finding the values of W,b)

Step 5 : Update the Weights

4.9	3.0	1.4	0.2

x

1-δ	1-δ	1-δ
1-δ	1-δ	1-δ
1-δ	1-δ	1-δ
1-δ	1-δ	1-δ

=

9.5	9.5	9.5

Error = 84.25

Label :

1	0	0

$$E_{total} = \sum \frac{1}{2}(target - output)^2$$

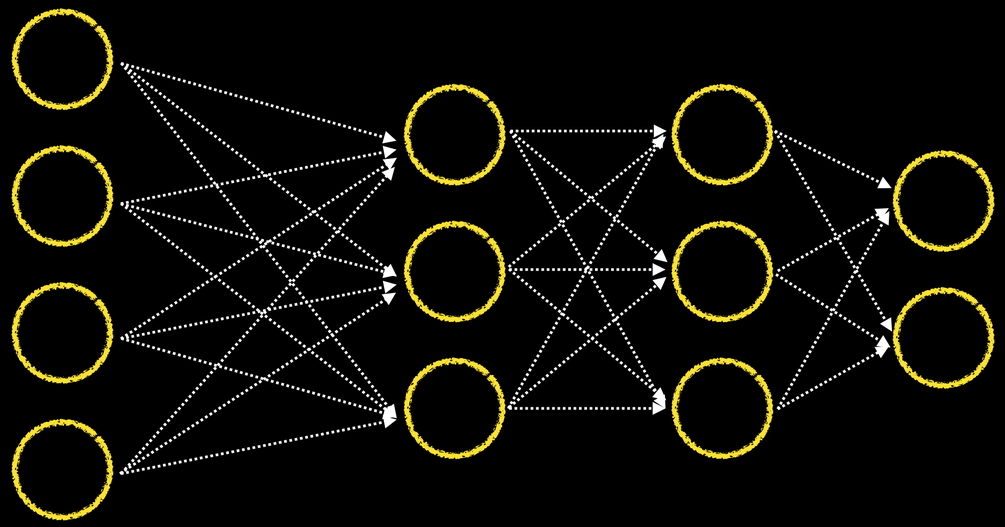
$$\frac{\partial E_{total}}{\partial out_{o1}} = 2 * \frac{1}{2}(target_{o1} - out_{o1})^{2-1} * -1 + 0$$

$$\frac{\partial E_{total}}{\partial out_{o1}} = -(target_{o1} - out_{o1})$$

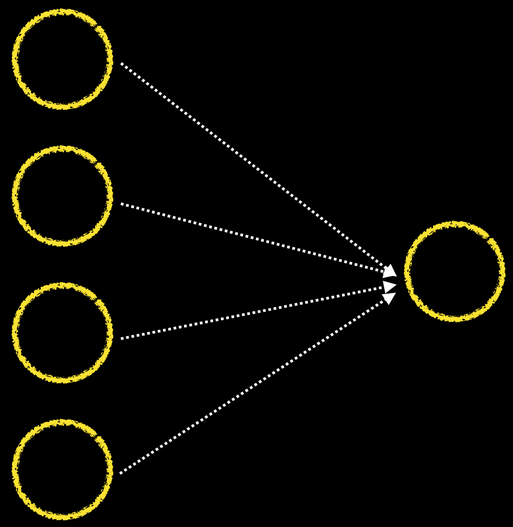
Gradients (Derivatives)

Subtract from each weight a small fraction of its corresponding derivative (Delta Rule)

Fully Connected Layer



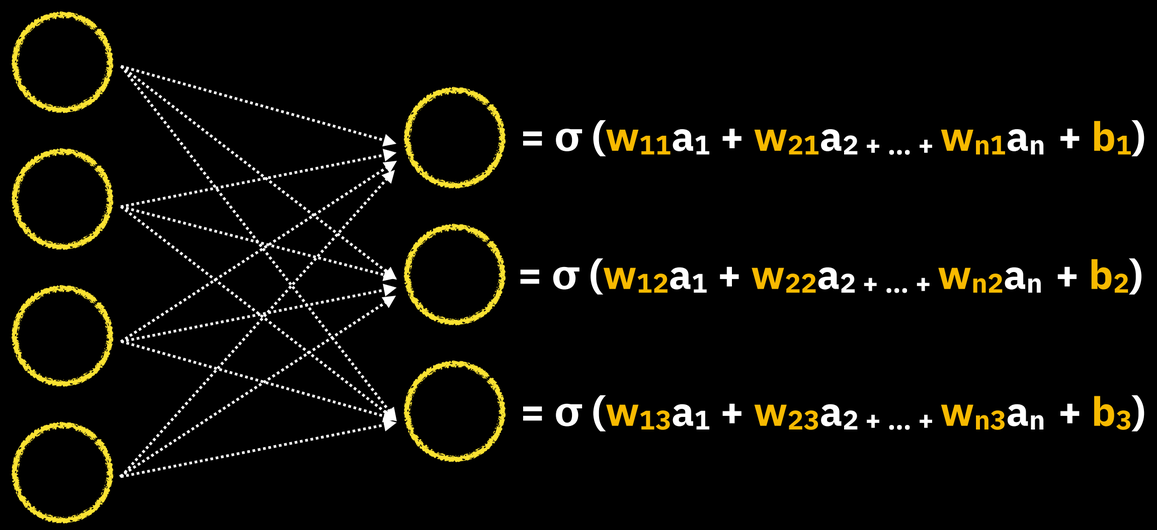
Fully Connected Layer



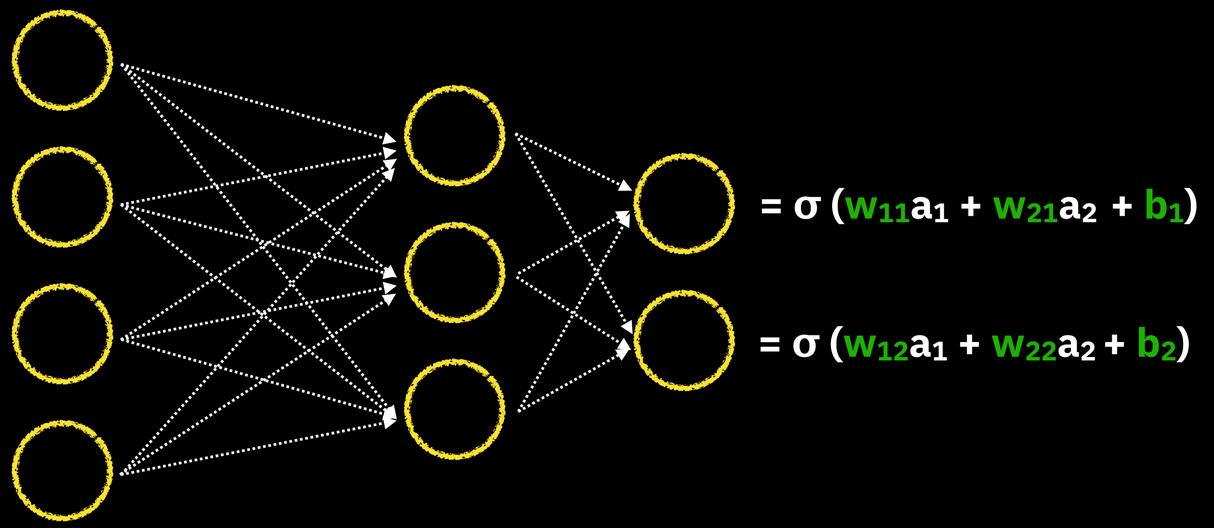
Neural Network Function

$= \sigma (w_1 a_1 + w_2 a_2 + \ldots + w_n a_n + b)$

Fully Connected Layer



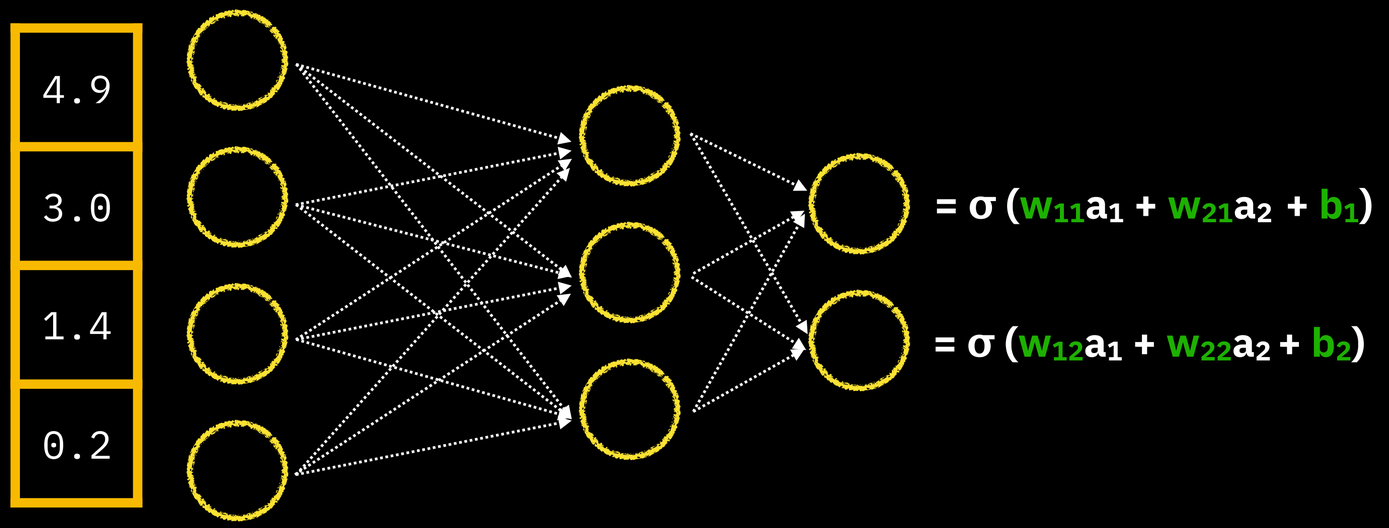
Fully Connected Layer



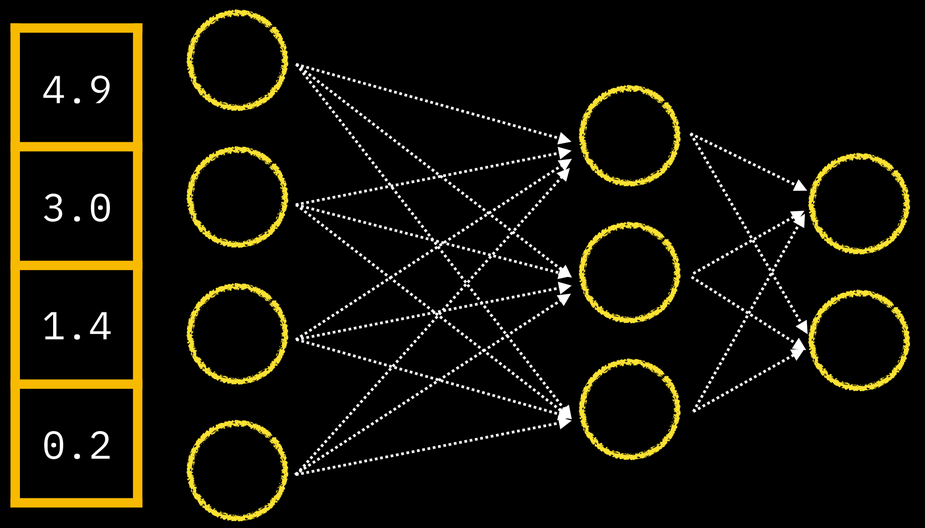
$= \sigma\,(w_{11}a_1 + w_{21}a_2 + b_1)$

$= \sigma\,(w_{12}a_1 + w_{22}a_2 + b_2)$

Fully Connected Layer



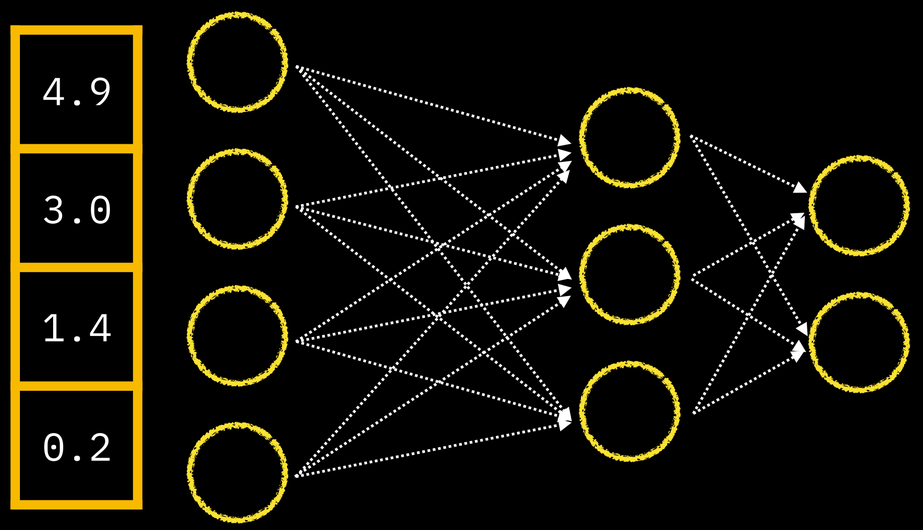
Fully Connected Layer



$= \sigma\,(w_{11}a_1 + w_{21}a_2 + b_1) = 0.34$

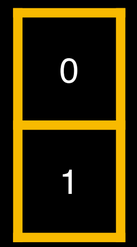
$= \sigma\,(w_{12}a_1 + w_{22}a_2 + b_2) = 0.11$

Fully Connected Layer



$= \sigma\,(w_{11}a_1 + w_{21}a_2 + b_1) = 0.34$

$= \sigma\,(w_{12}a_1 + w_{22}a_2 + b_2) = 0.11$



$(\sigma\,(w_1a_1 + w_2a_{2\,+\,\ldots\,+}\,w_na_n + b) - \text{True})$

Cost/Loss Function

Fully Connected Layer

Neural Function

Input : 4
Output : 3
Parameters : ((4 x 3) + 3) + ((3 x 2) + 2) = 23

Fully Connected Layer

Neural Function

Input : 4
Output : 3
Parameters : ((4 x 3) + 3) + ((3 x 2) + 2) = 23

Cost Function

Input : 23
Output : 1 (Cost)
Parameters : Your training examples

Cost Function

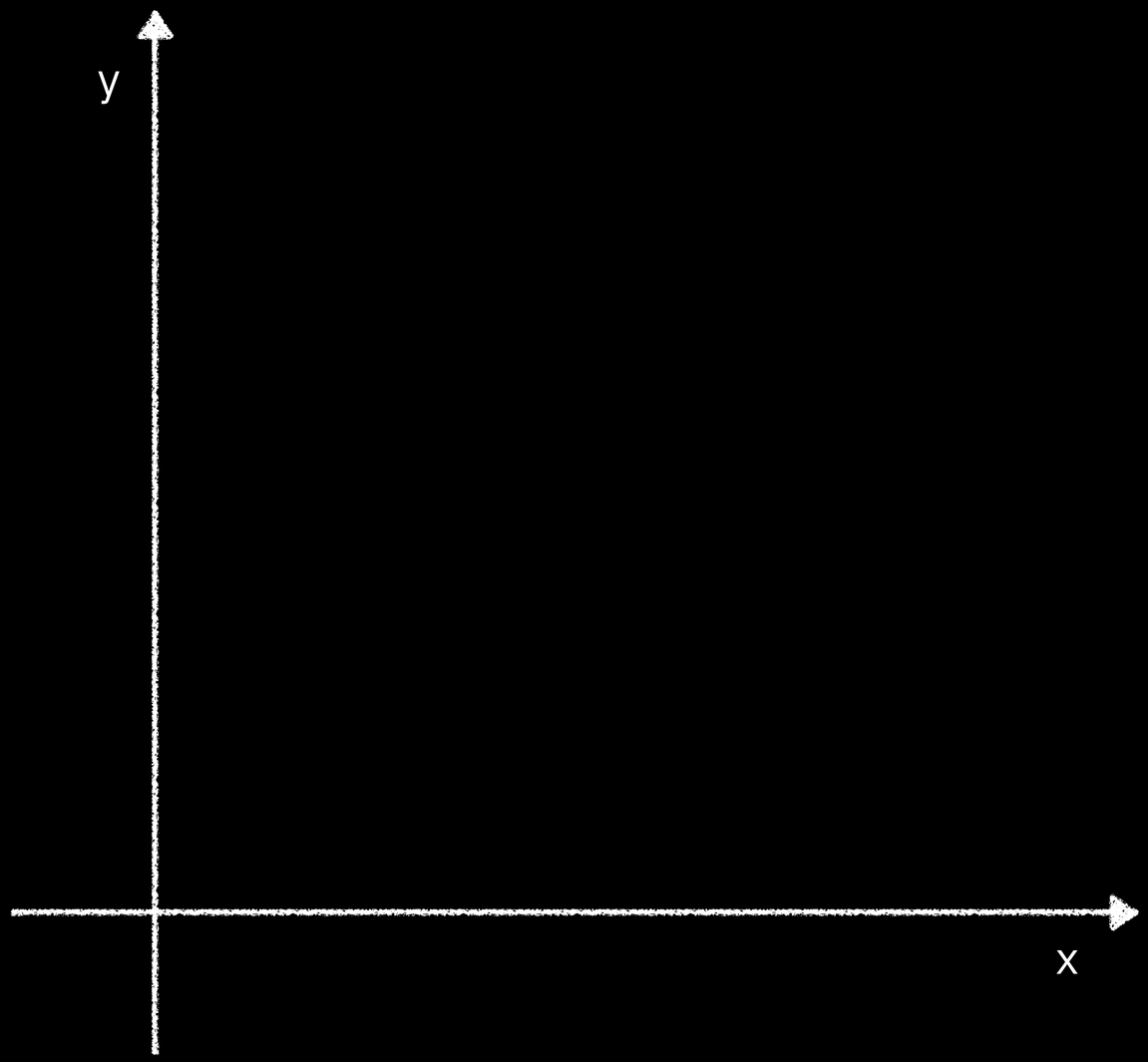
Input : 23
Output : 1 (Cost)
Parameters : Your training examples

$$C(w_1, w_2 \dots w_n)$$

Weights and Biases

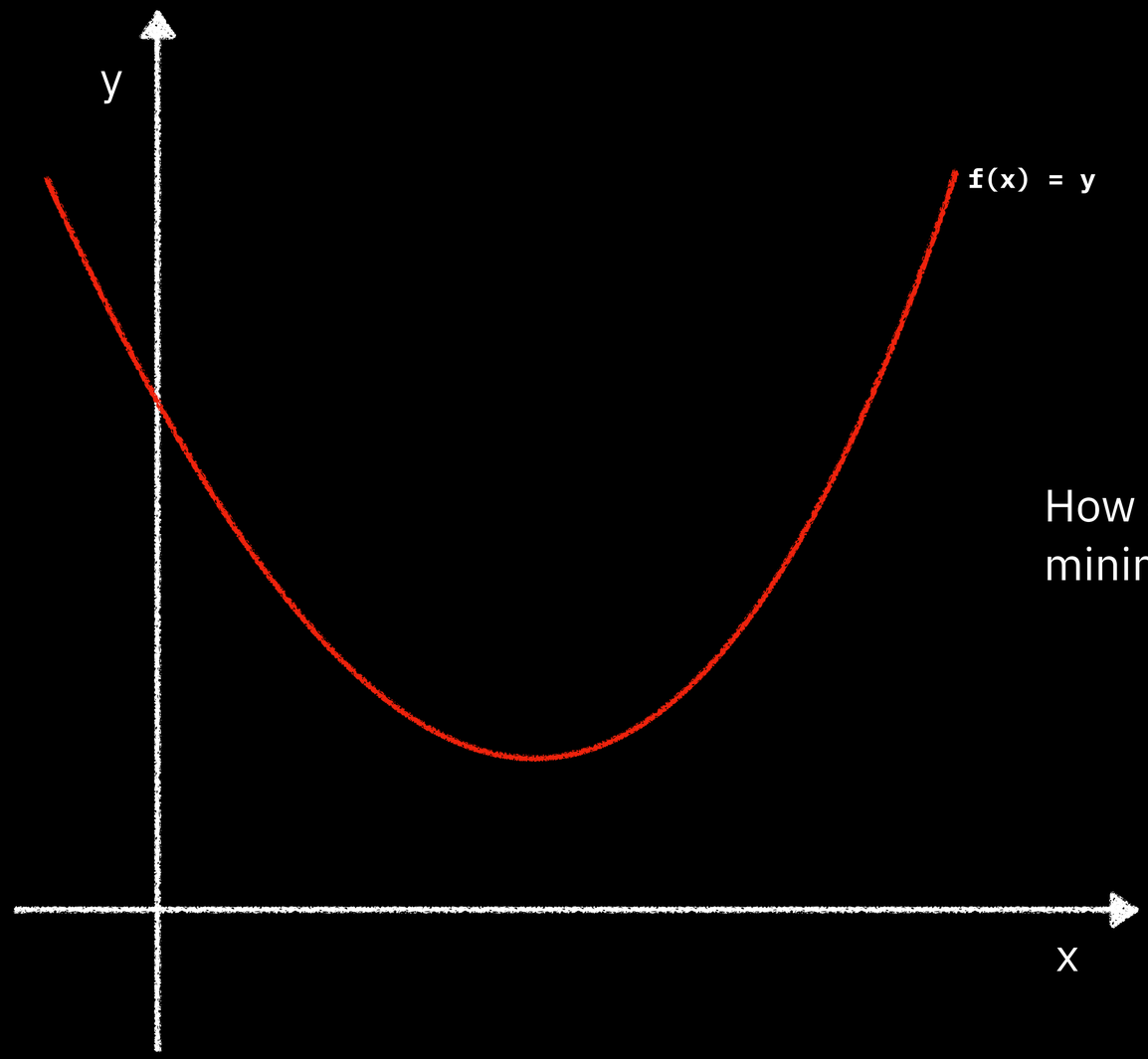
Cost Function

$C(w)$



Cost Function

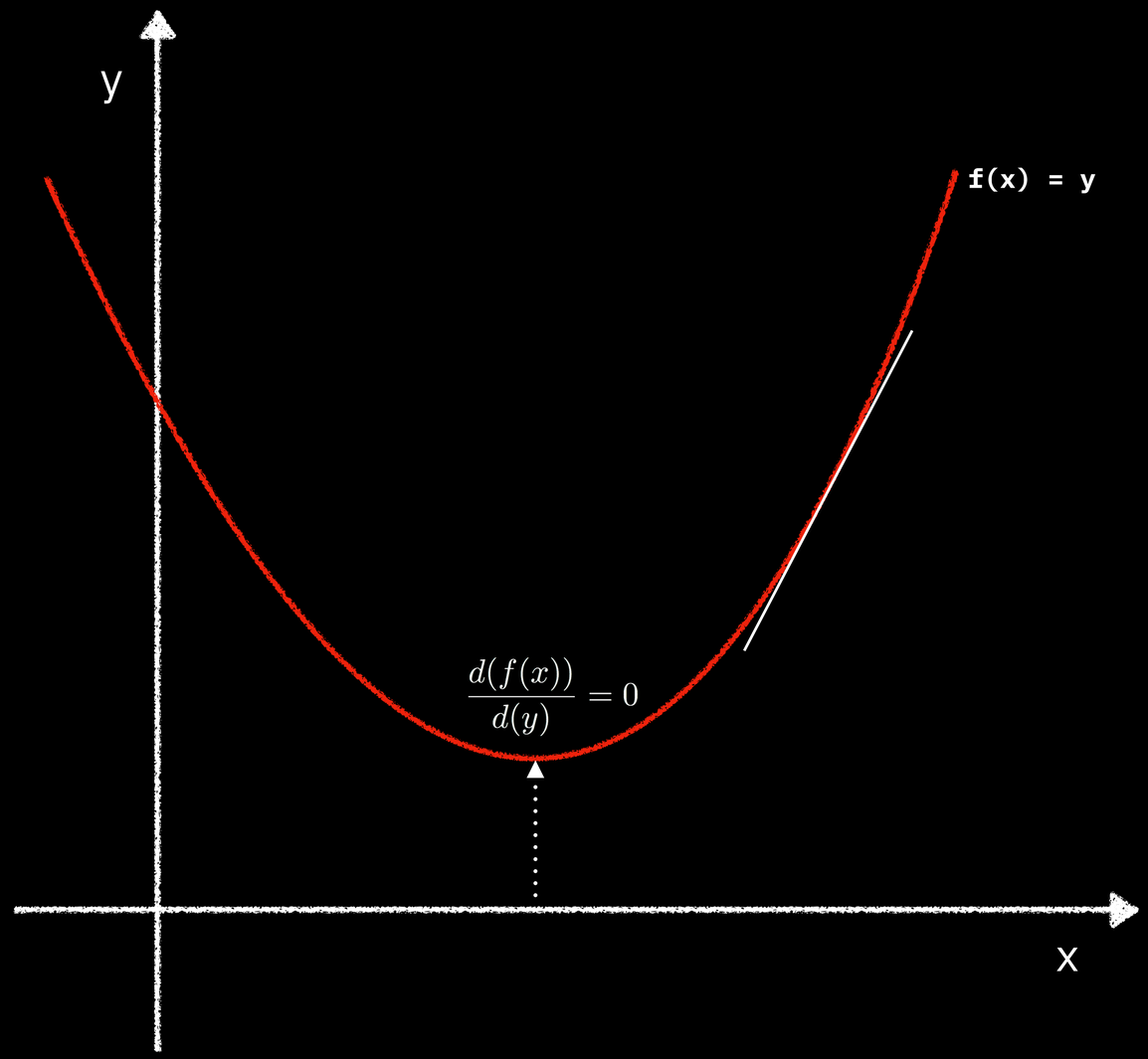
C(w)



How do you find the minima in a function?

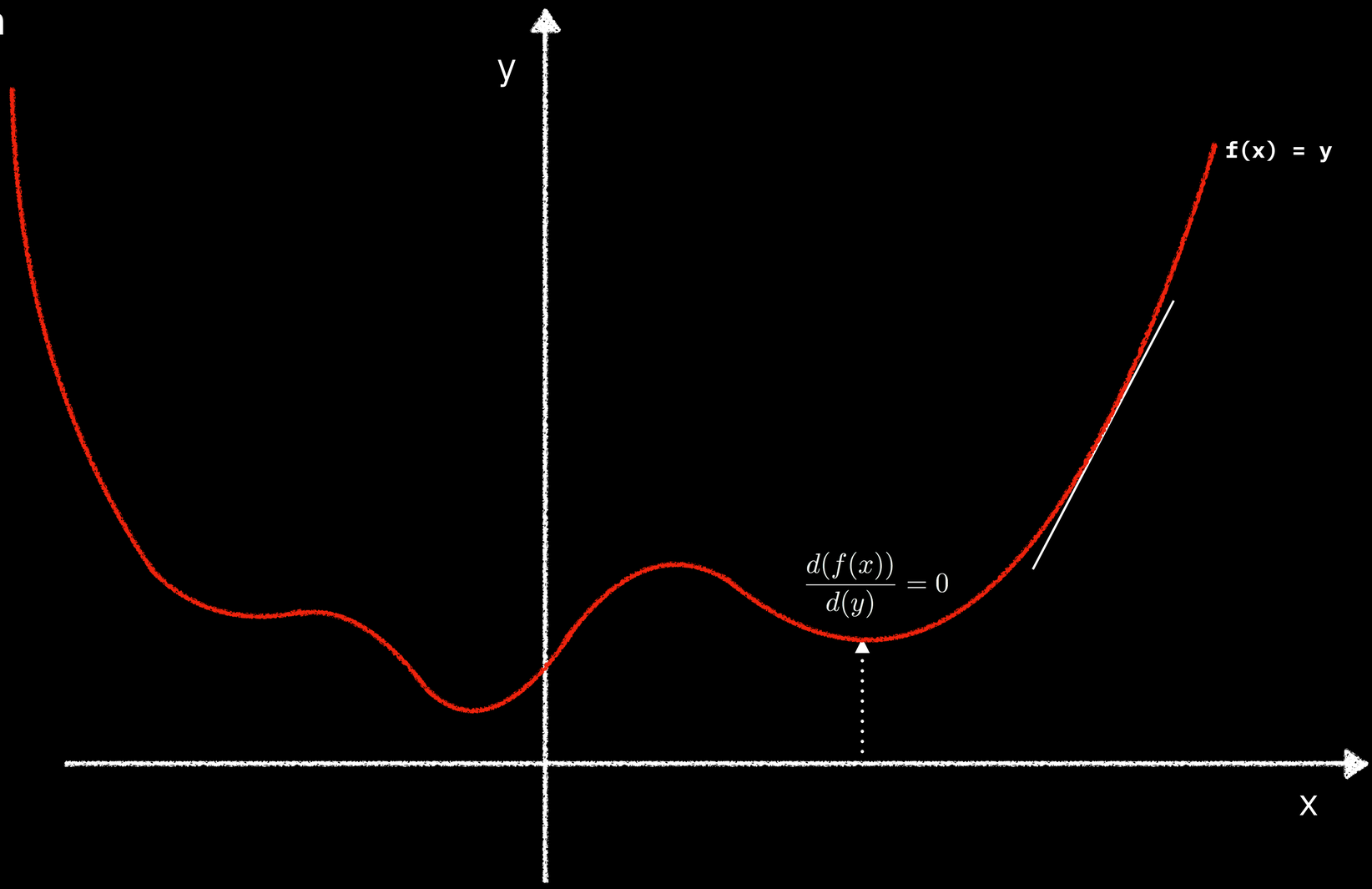
Cost Function

C(w)



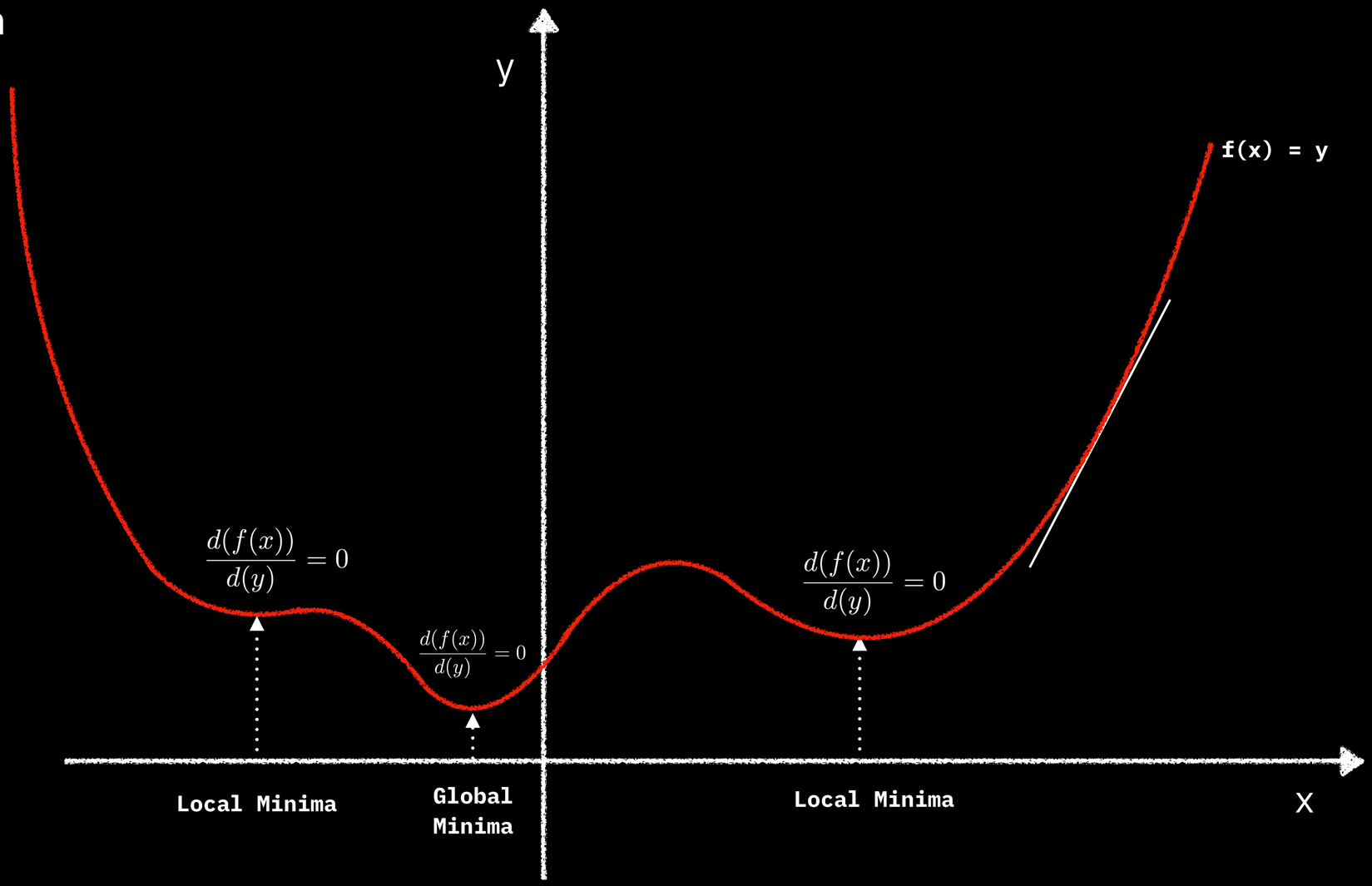
Cost Function

C(w)



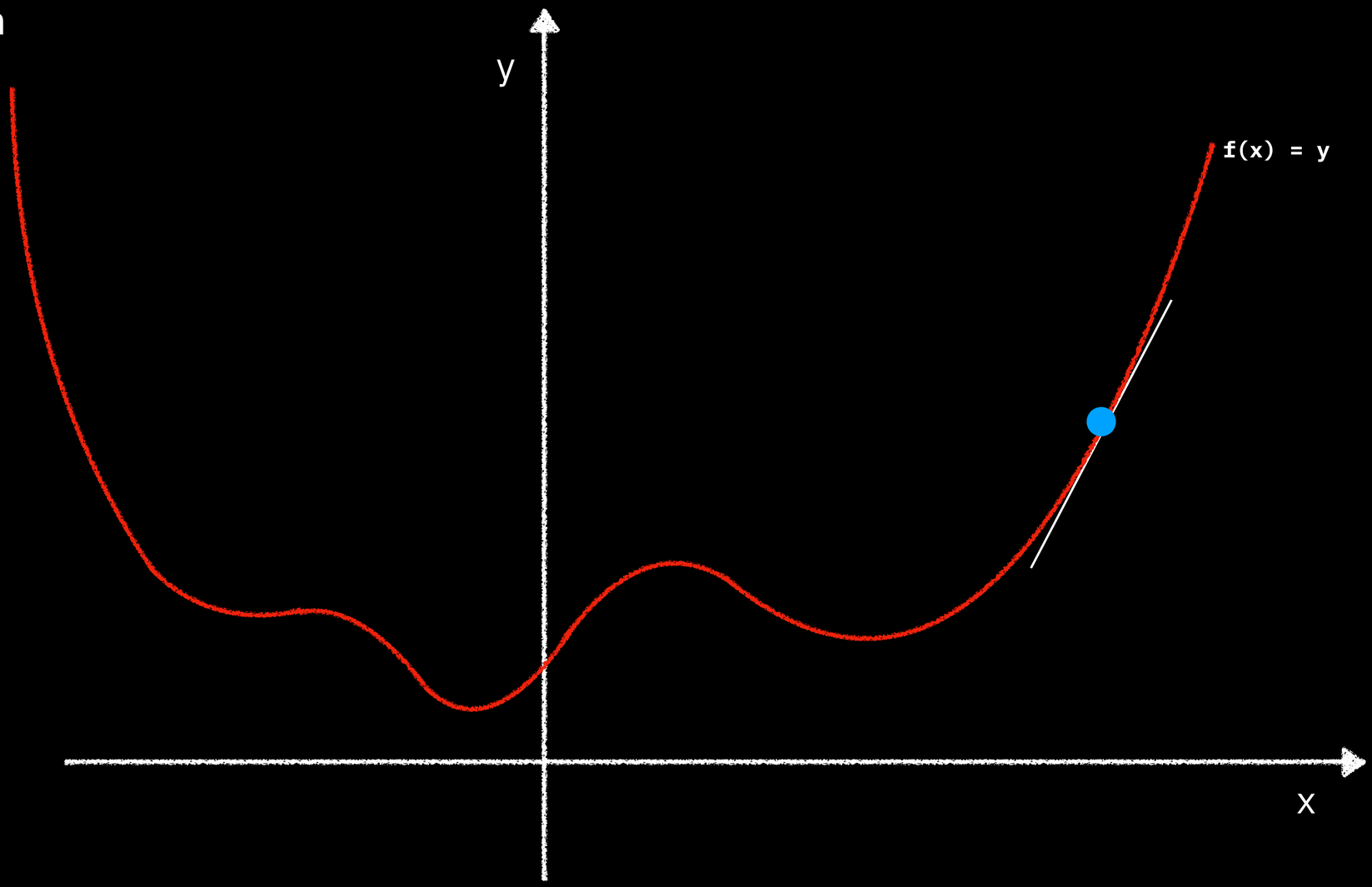
Cost Function

C(w)



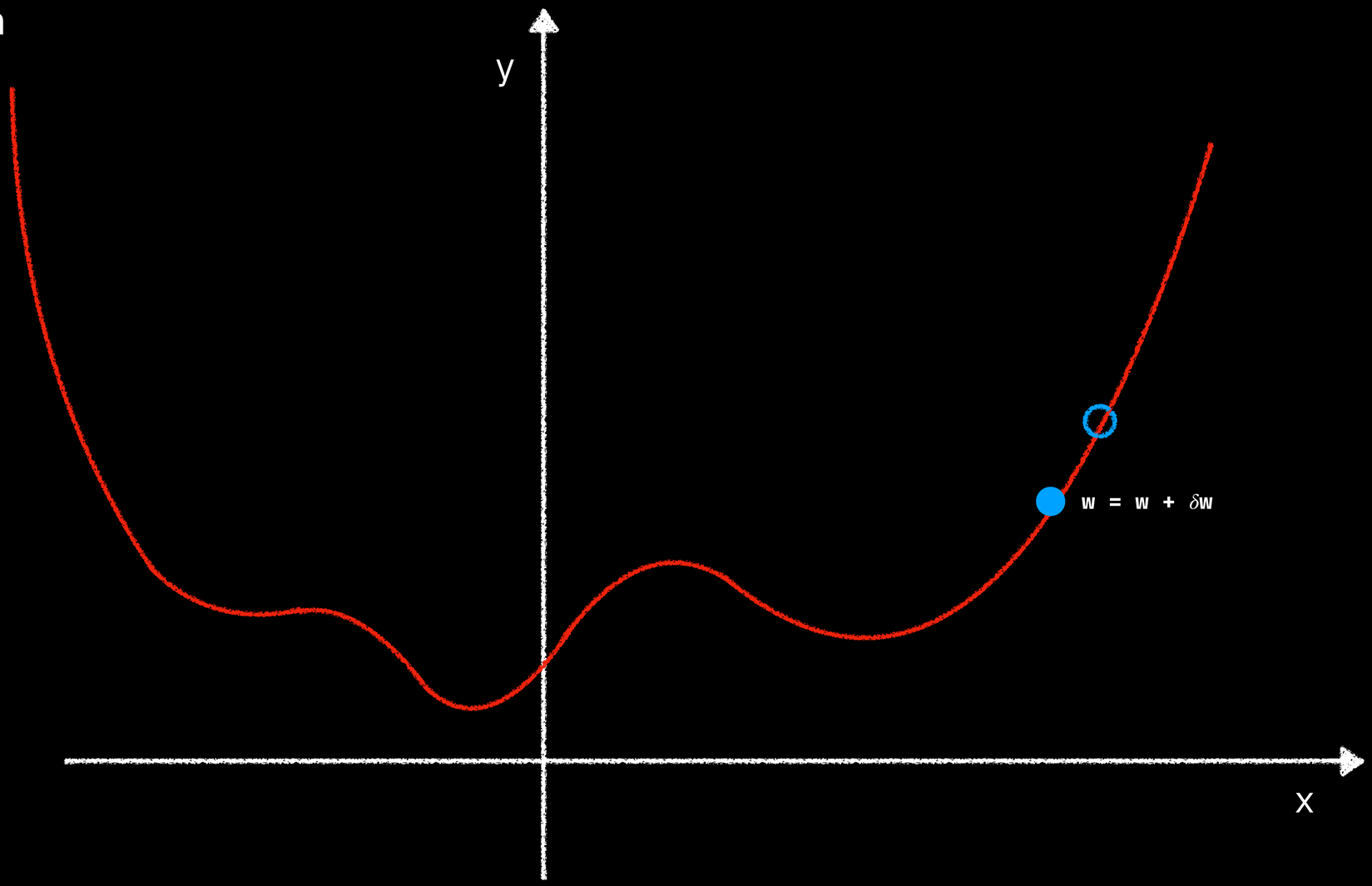
Cost Function

$C(w)$



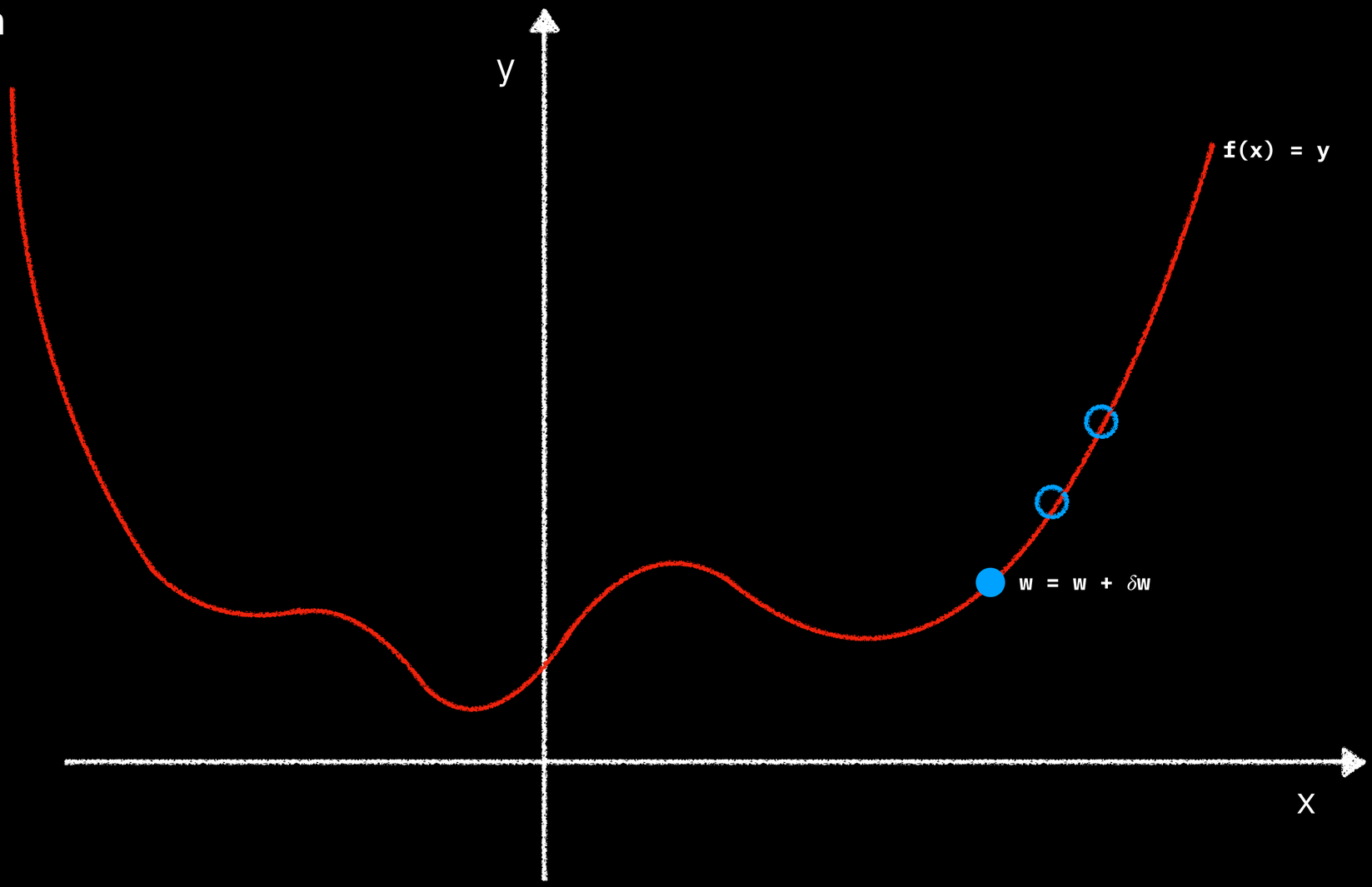
Cost Function

$C(w)$



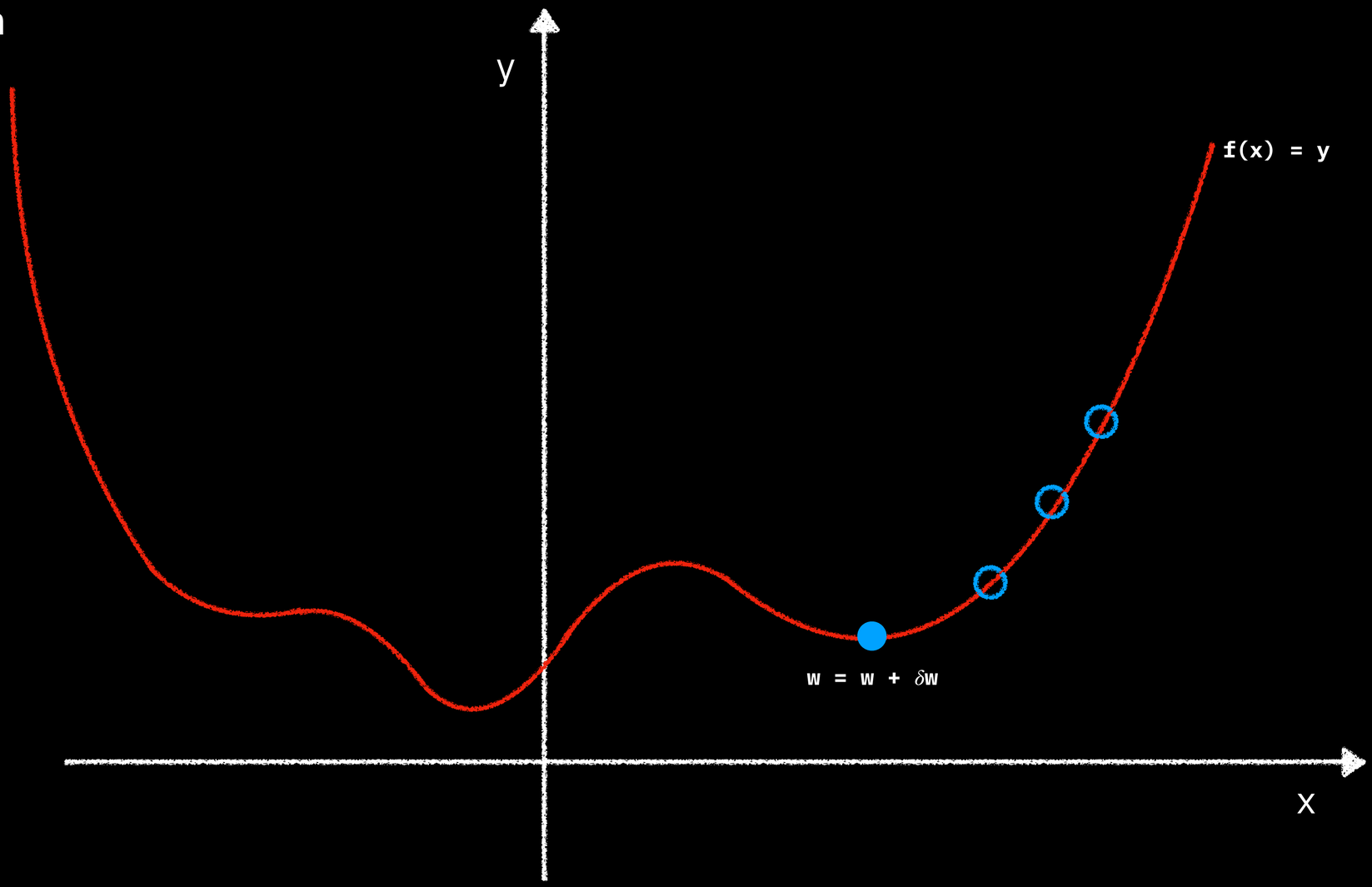
Cost Function

$C(w)$



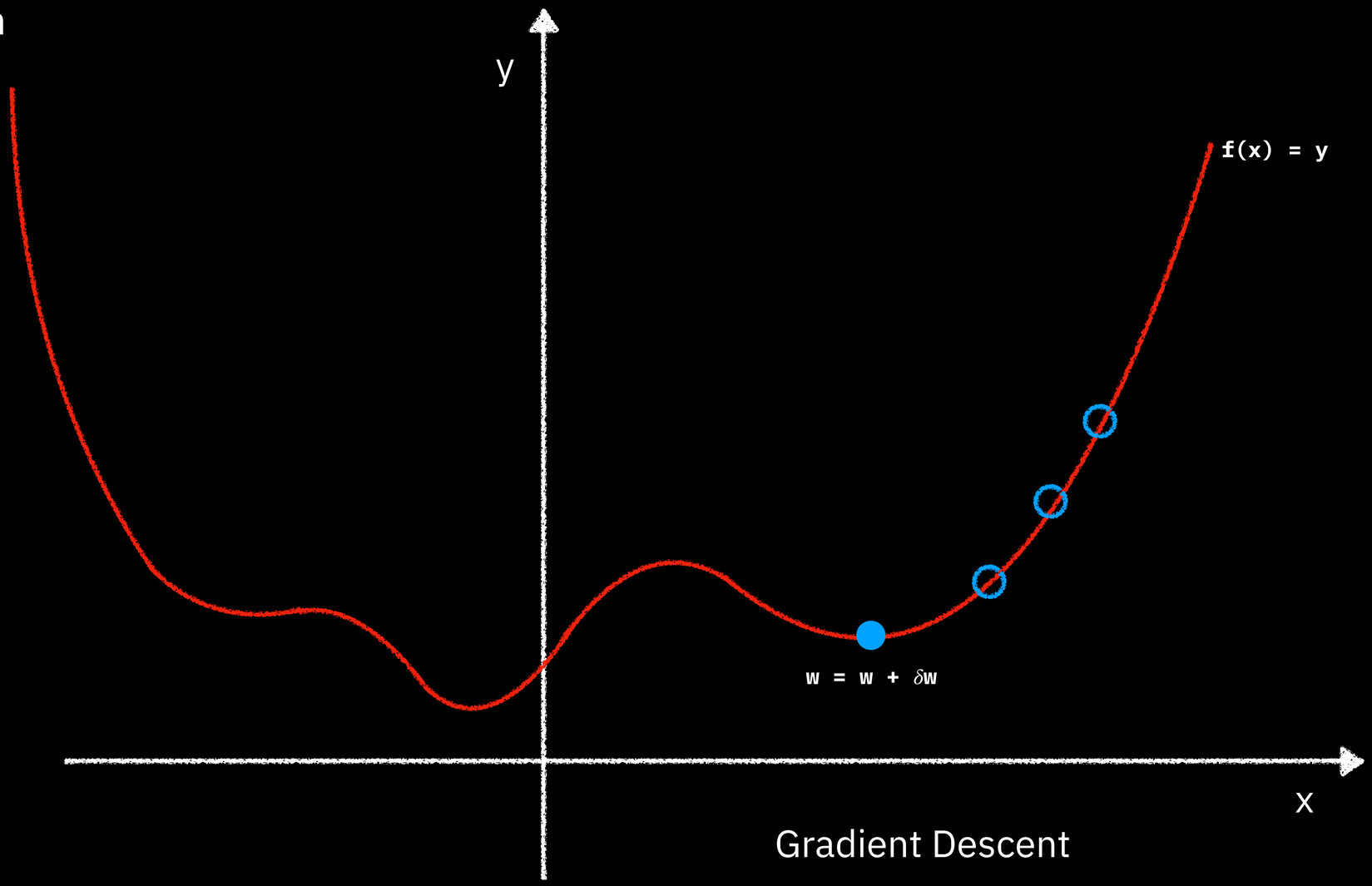
Cost Function

$C(w)$



Cost Function

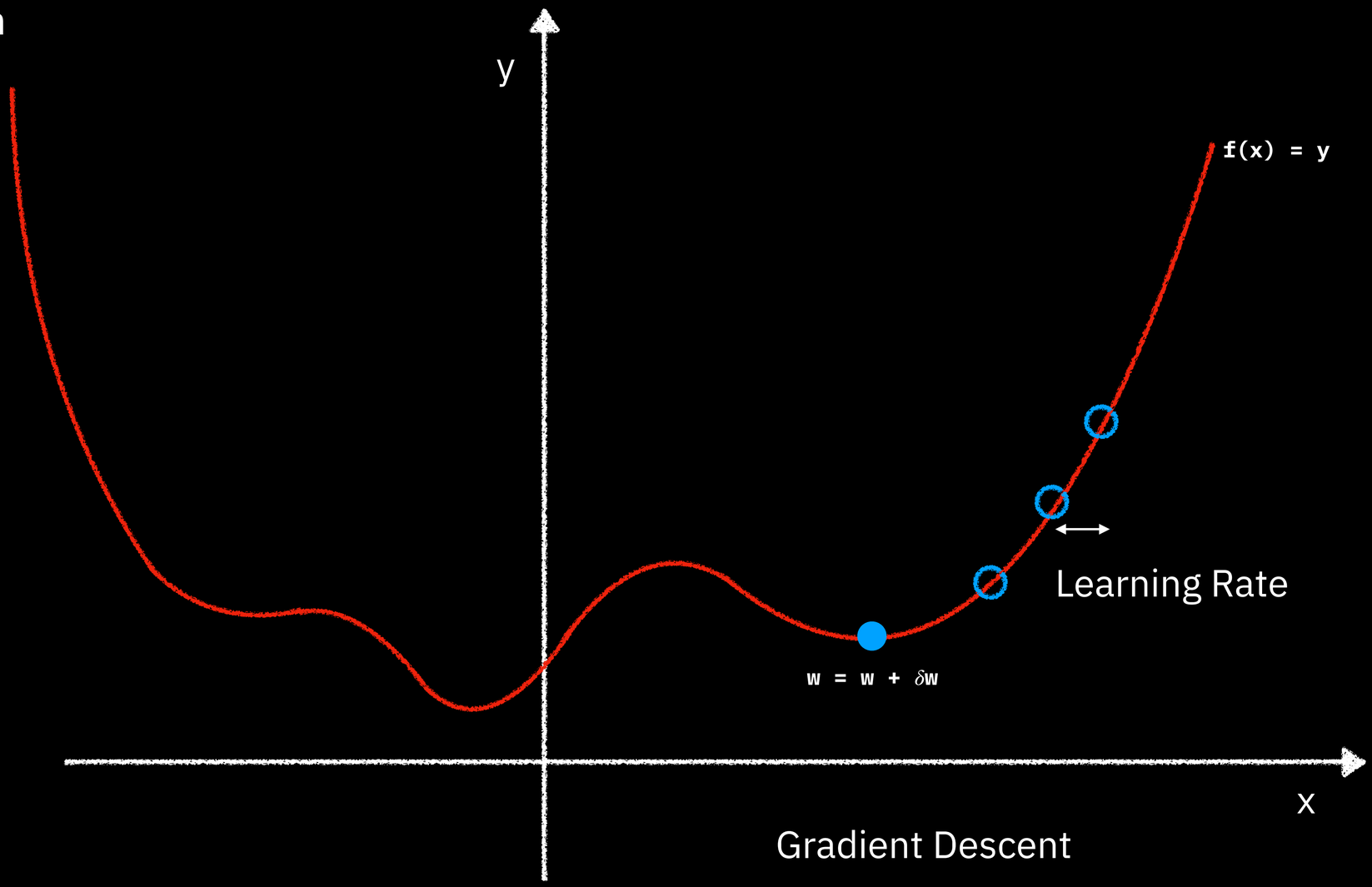
$C(w)$



Gradient Descent

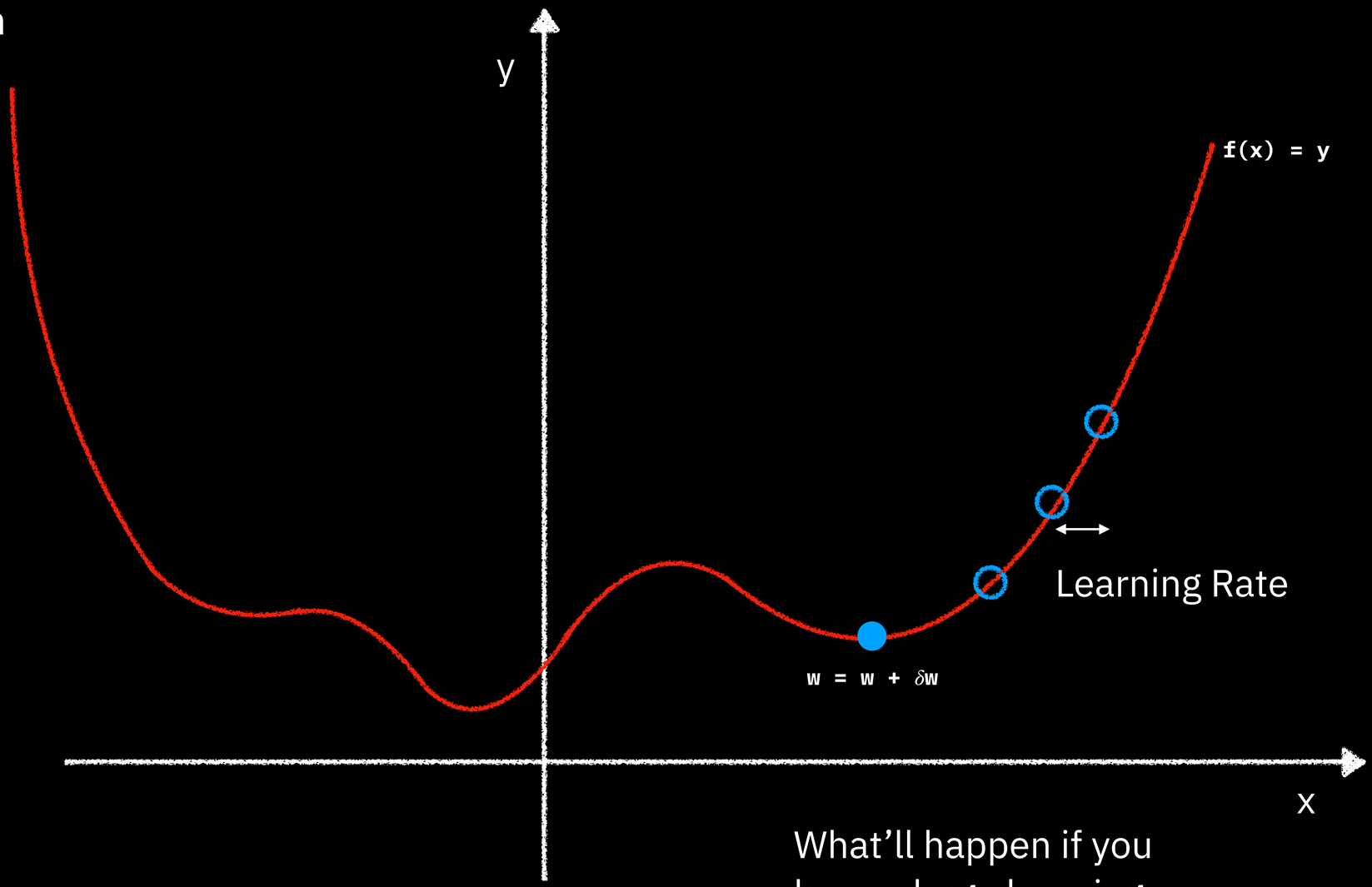
Cost Function

$C(w)$



Cost Function

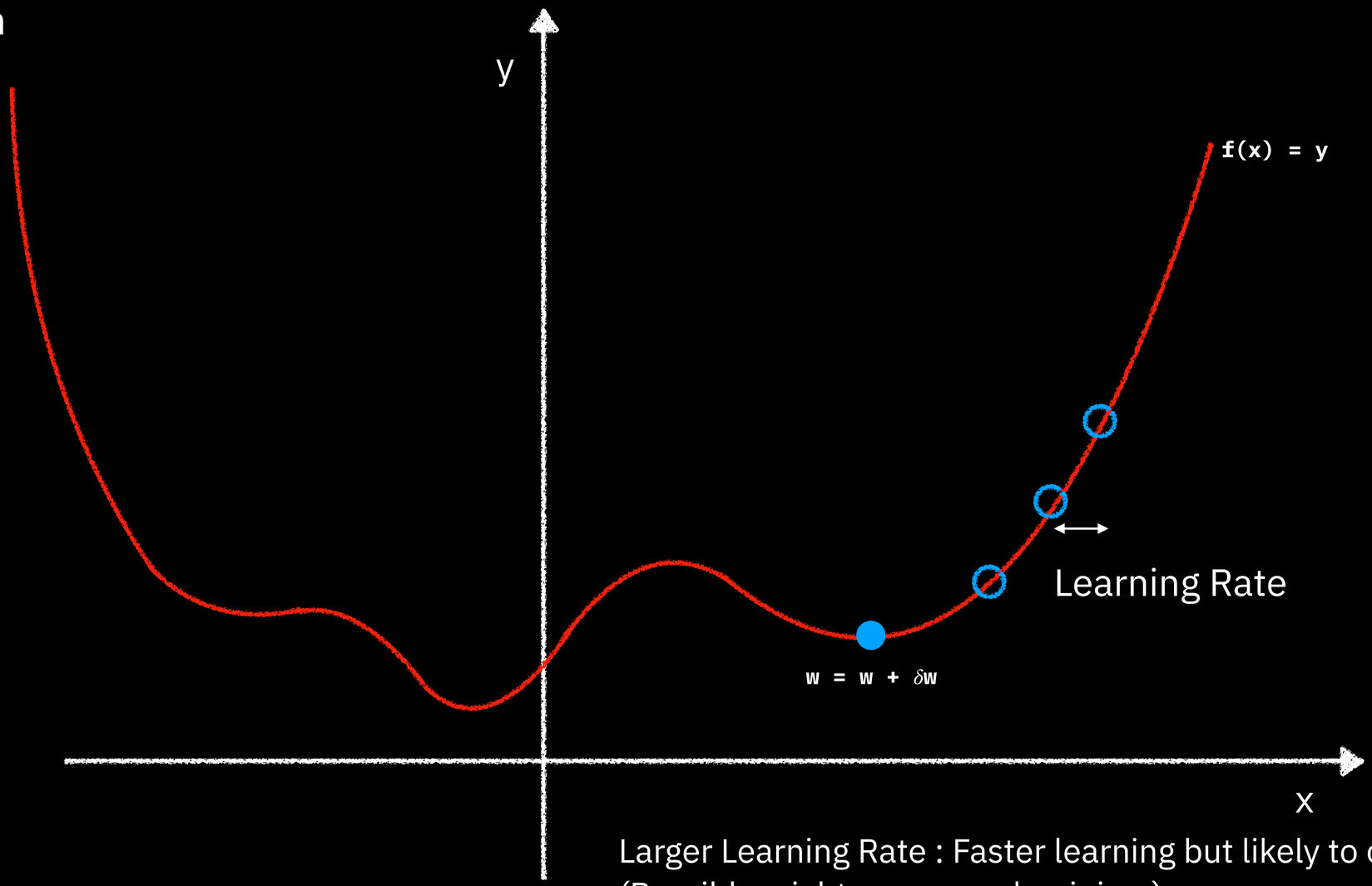
C(w)



What'll happen if you have a large learning rate value?

Cost Function

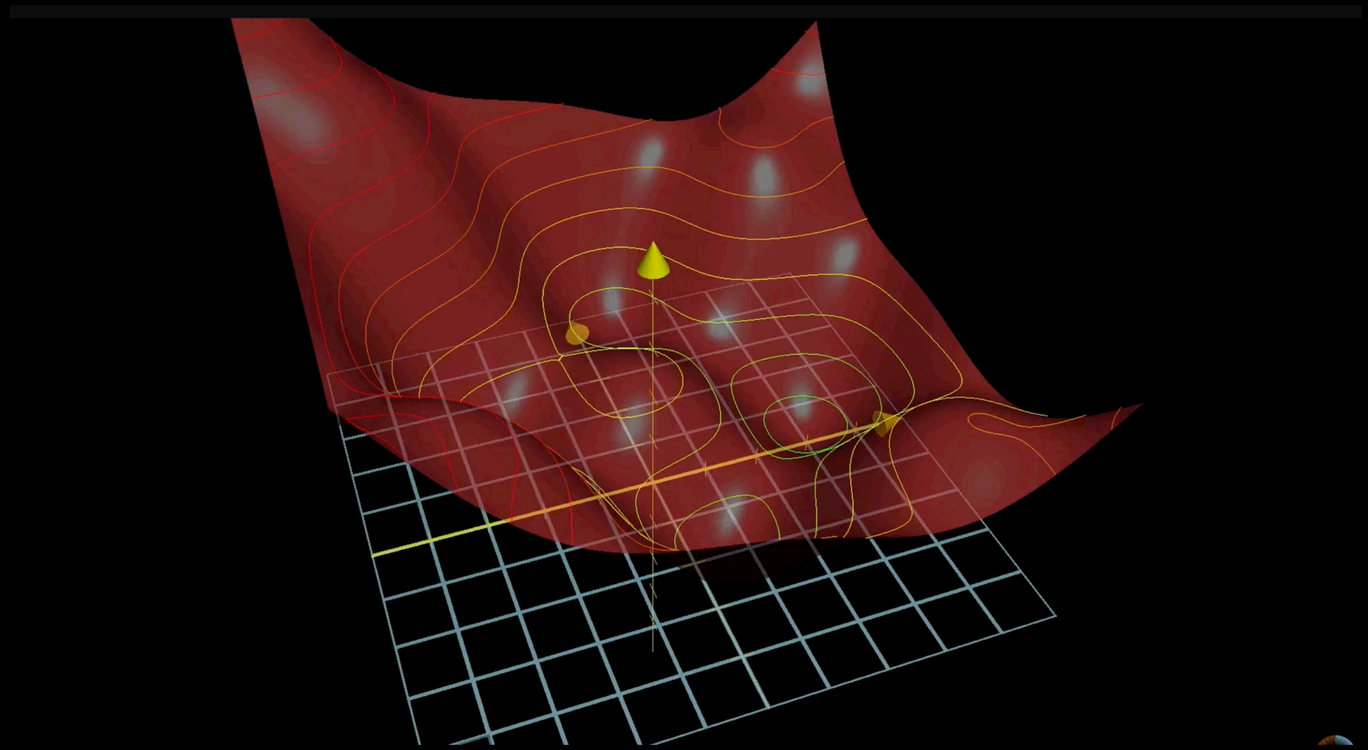
$C(w)$



Larger Learning Rate : Faster learning but likely to overshoot the minima.
(Possibly might never reach minima)
Small learning rate : Maybe not even learn before your lifetime.

Cost Function

Neural Network Learning = Minimising a Cost Function
Method to do that : Gradient Descent



// Youtube Video : 3Blue1Brown Gradient Descent

Improving Neural Networks

Consider a function : Input 1 ; Output 0
Let’s play with different initial weights.
Learning Rate constant.

```
// http://neuralnetworksanddeeplearning.com/chap3.html
```

Loss Function

Quadratic Loss function

$$\text{Loss} = \sum \frac{1}{2}(\texttt{target} - \texttt{output})^2$$

Loss Function

Quadratic Loss function

$$\text{Loss} = \sum \frac{1}{2}(\texttt{target} - \texttt{output})^2$$

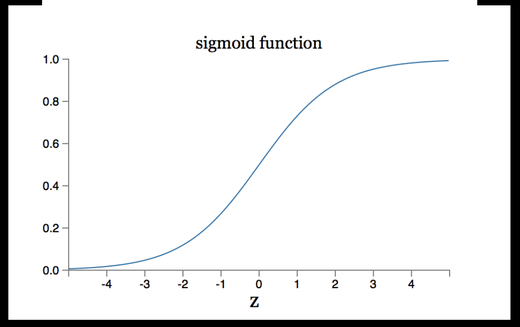
$z = wx + b$

$a = \sigma(z)$

$$C = \frac{(y-a)^2}{2}$$

$$\frac{\partial C}{\partial w} = (a-y)\sigma'(z)x = a\sigma'(z)$$

$$\frac{\partial C}{\partial b} = (a-y)\sigma'(z) = a\sigma'(z),$$



Loss Function

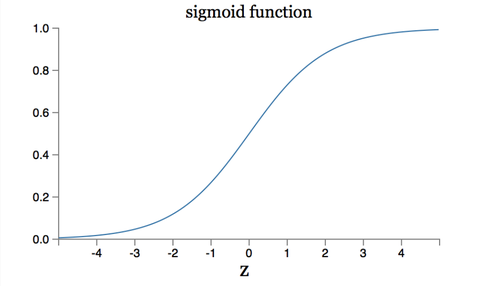
Quadratic Loss function

$$\text{Loss} = \sum \frac{1}{2}(\texttt{target} - \texttt{output})^2$$

z = wx + b
a = σ(z)

$$C = \frac{(y-a)^2}{2}$$

$$\frac{\partial C}{\partial w} = (a-y)\sigma'(z)x = a\sigma'(z)$$
$$\frac{\partial C}{\partial b} = (a-y)\sigma'(z) = a\sigma'(z),$$



Cross-entropy Loss function

$$C = -\frac{1}{n}\sum_x [y \ln a + (1-y)\ln(1-a)]$$

- Non-Negative (C > 0)
- Tends towards 0, if the desired output is close to actual output

$$\begin{aligned}\frac{\partial C}{\partial w_j} &= -\frac{1}{n}\sum_x \left(\frac{y}{\sigma(z)} - \frac{(1-y)}{1-\sigma(z)}\right)\frac{\partial \sigma}{\partial w_j} \\ &= -\frac{1}{n}\sum_x \left(\frac{y}{\sigma(z)} - \frac{(1-y)}{1-\sigma(z)}\right)\sigma'(z)x_j.\end{aligned}$$

$$\frac{\partial C}{\partial w_j} = \frac{1}{n}\sum_x x_j(\sigma(z) - y).$$

Loss Function

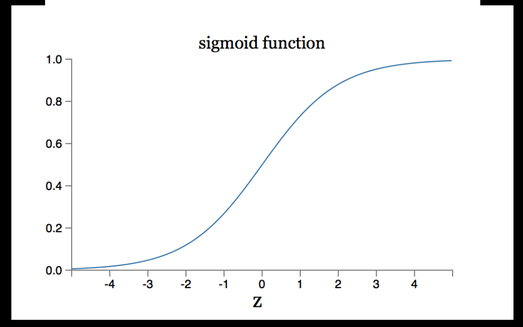
Quadratic Loss function

$$\text{Loss} = \sum \frac{1}{2}(\texttt{target} - \texttt{output})^2$$

z = wx + b
a = σ(z)

$$C = \frac{(y-a)^2}{2}$$

$$\frac{\partial C}{\partial w} = (a-y)\sigma'(z)x = a\sigma'(z)$$
$$\frac{\partial C}{\partial b} = (a-y)\sigma'(z) = a\sigma'(z),$$



Cross-entropy Loss function

$$C = -\frac{1}{n}\sum_x [y \ln a + (1-y)\ln(1-a)]$$

- Non-Negative (C > 0)
- Tends towards 0, if the desired output is close to actual output

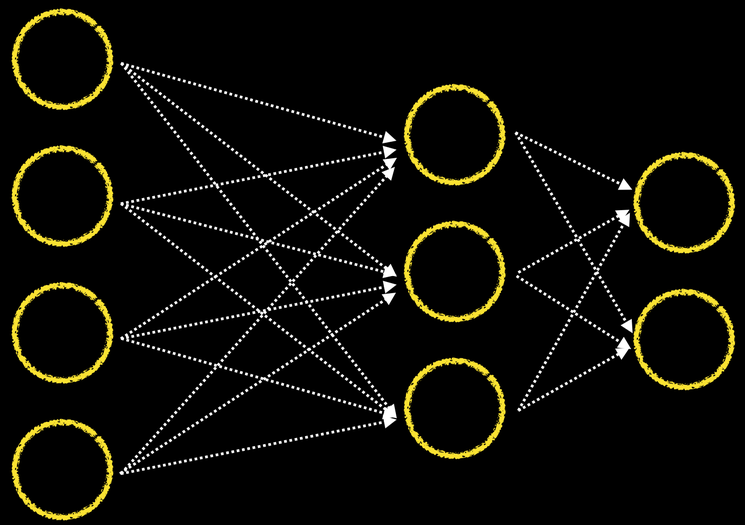
$$\begin{aligned}\frac{\partial C}{\partial w_j} &= -\frac{1}{n}\sum_x \left(\frac{y}{\sigma(z)} - \frac{(1-y)}{1-\sigma(z)}\right)\frac{\partial \sigma}{\partial w_j} \\ &= -\frac{1}{n}\sum_x \left(\frac{y}{\sigma(z)} - \frac{(1-y)}{1-\sigma(z)}\right)\sigma'(z)x_j.\end{aligned}$$

$$\frac{\partial C}{\partial w_j} = \frac{1}{n}\sum_x x_j(\sigma(z) - y).$$

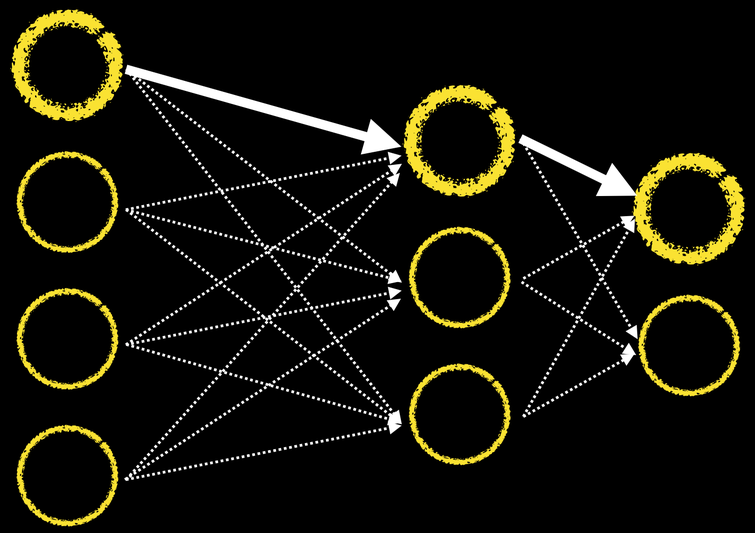
Homework :

- Think a little more about the origin of this problem.
- Come up with how cross-entropy term can derived from that problem.
- Find the derivative of the sigmoid function.

Computational Graphs (Backpropagation)



Computational Graphs (Backpropagation)

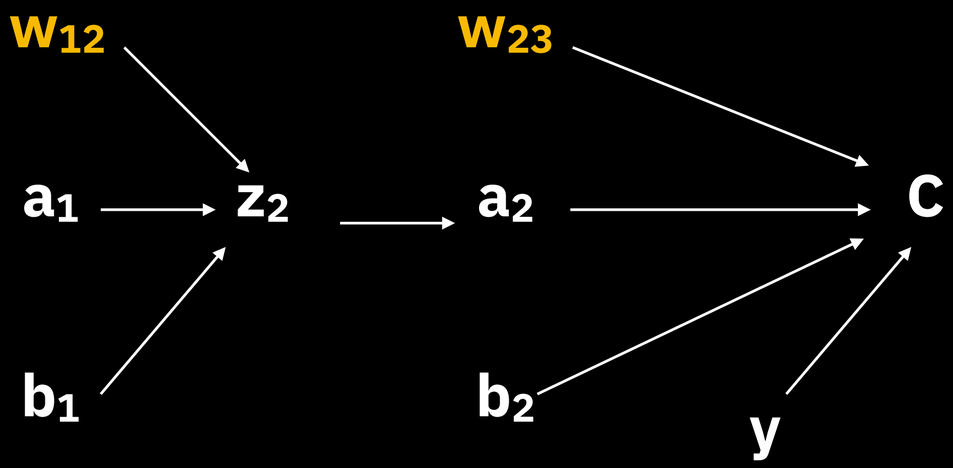


Computational Graphs (Backpropagation)

$\text{Cost} = C(a_3, y)$

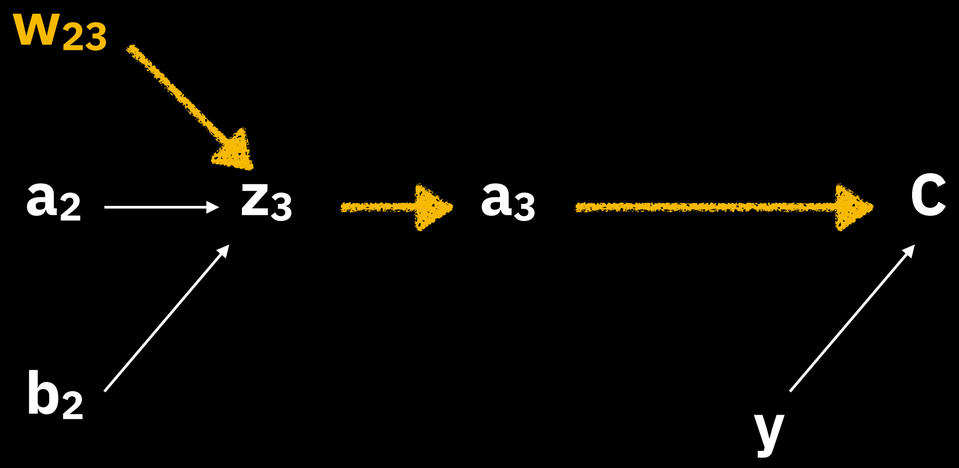
Computational Graphs (Backpropagation)

$\text{Cost} = C(a_3, y)$



Computational Graphs (Backpropagation)

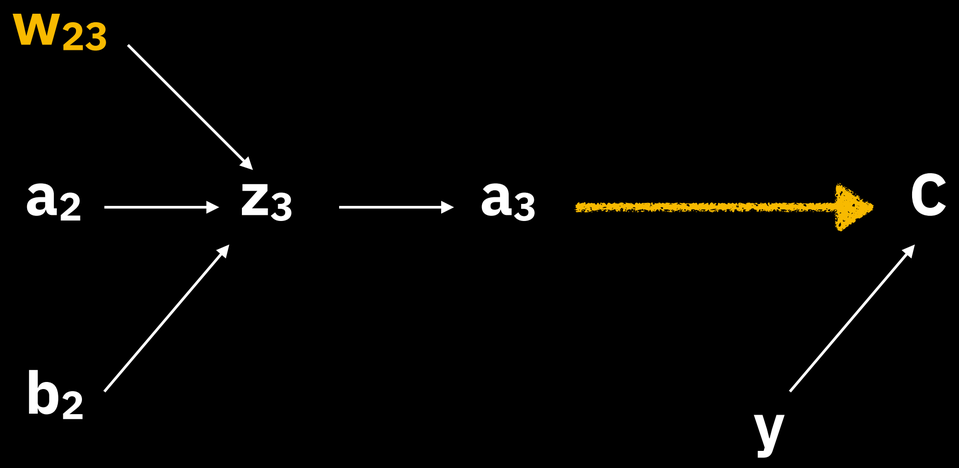
$Cost = C(a_3, y)$



$$\frac{\delta C}{\delta w_{23}}$$

Computational Graphs (Backpropagation)

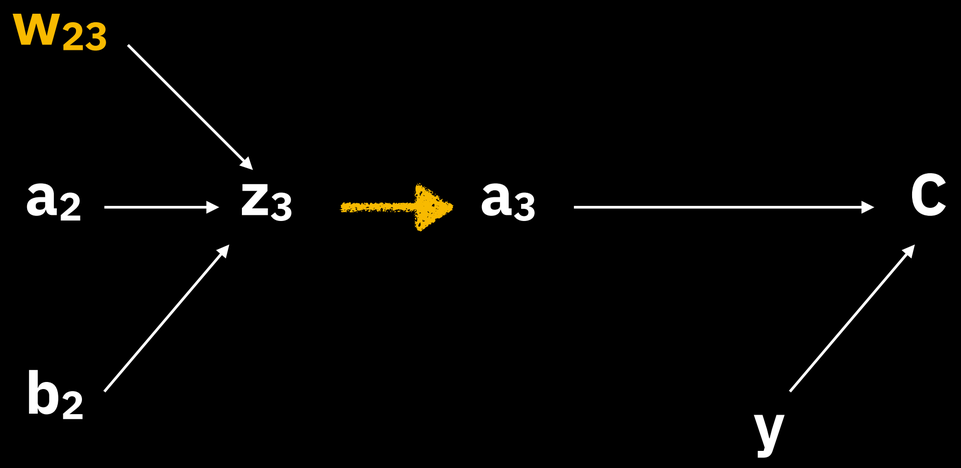
$Cost = C(a_3, y)$



$$\frac{\delta C}{\delta w_{23}} = \frac{\delta C}{\delta a_3}$$

Computational Graphs (Backpropagation)

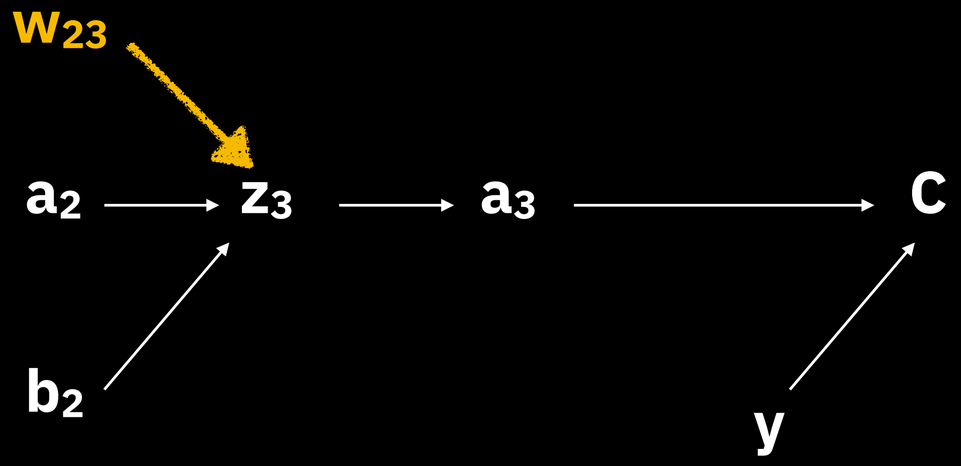
$Cost = C(a_3, y)$



$$\frac{\delta C}{\delta w_{23}} = \frac{\delta C}{\delta a_3} \frac{\delta a_3}{\delta z_3}$$

Computational Graphs (Backpropagation)

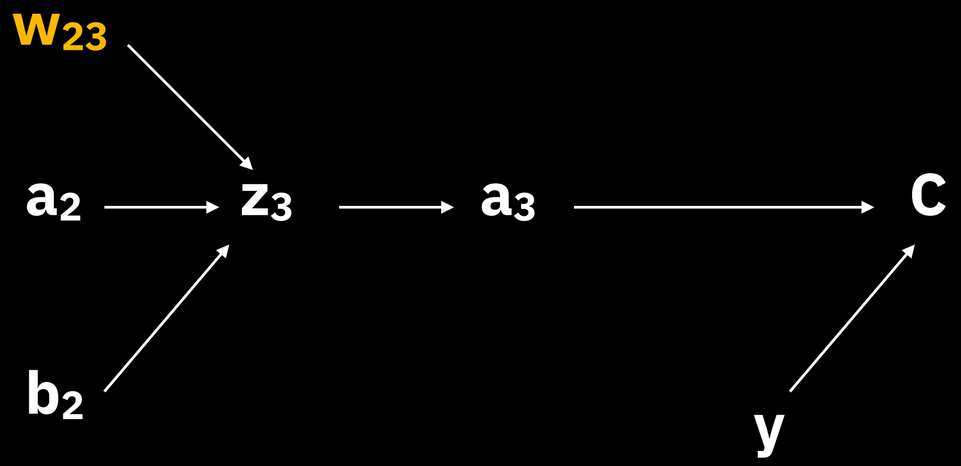
$Cost = C(a_3, y)$



$$\frac{\delta C}{\delta w_{23}} = \frac{\delta C}{\delta a_3} \frac{\delta a_3}{\delta z_3} \frac{\delta z_3}{\delta w_{23}}$$

Computational Graphs (Backpropagation)

Cost = $C(a_3, y)$

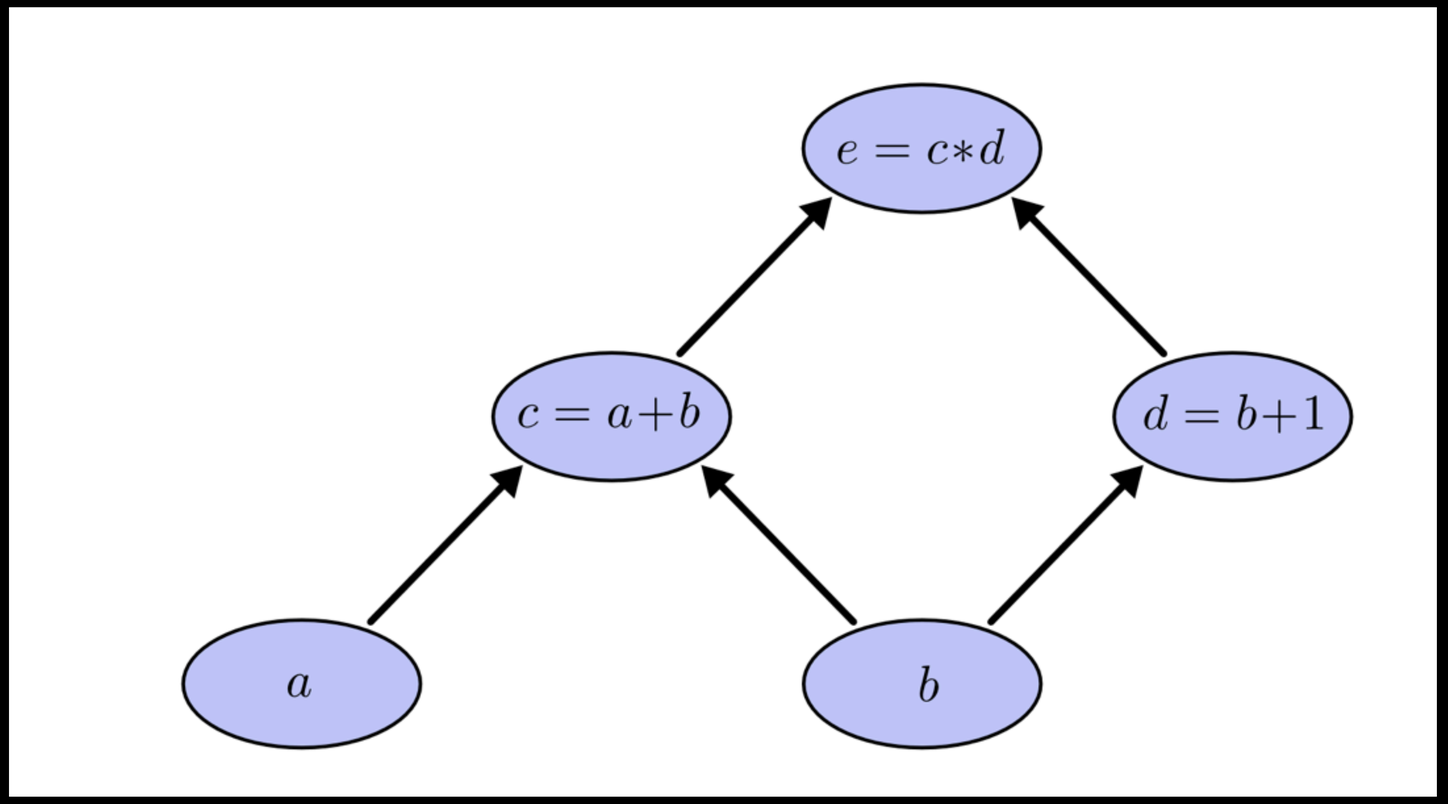


$$\frac{\delta C}{\delta w_{23}} = \frac{\delta C}{\delta a_3} \frac{\delta a_3}{\delta z_3} \frac{\delta z_3}{\delta w_{23}}$$

Chain Rule

Computational Graphs

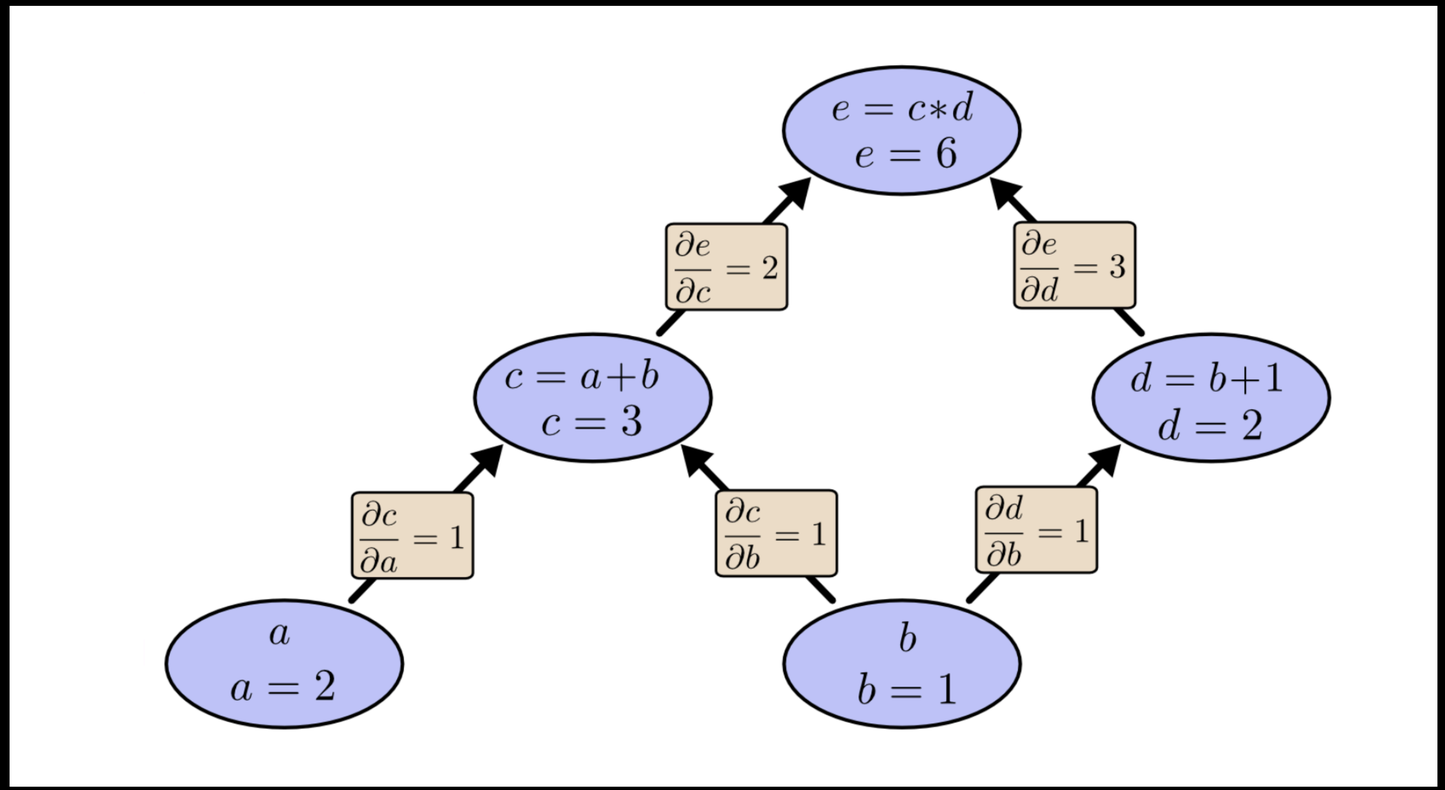
c = a + b
d = b + 1
e = c * d



Computational Graphs

Derivatives in Computational Graphs

$$\frac{\partial e}{\partial b} = 1 * 2 + 1 * 3$$



// coding time

Any Questions?

email id : harisris@gmail.com