

Artificial Neural Networks and Deep Learning

Week 3

Gradient descent and backpropagation

RECAP

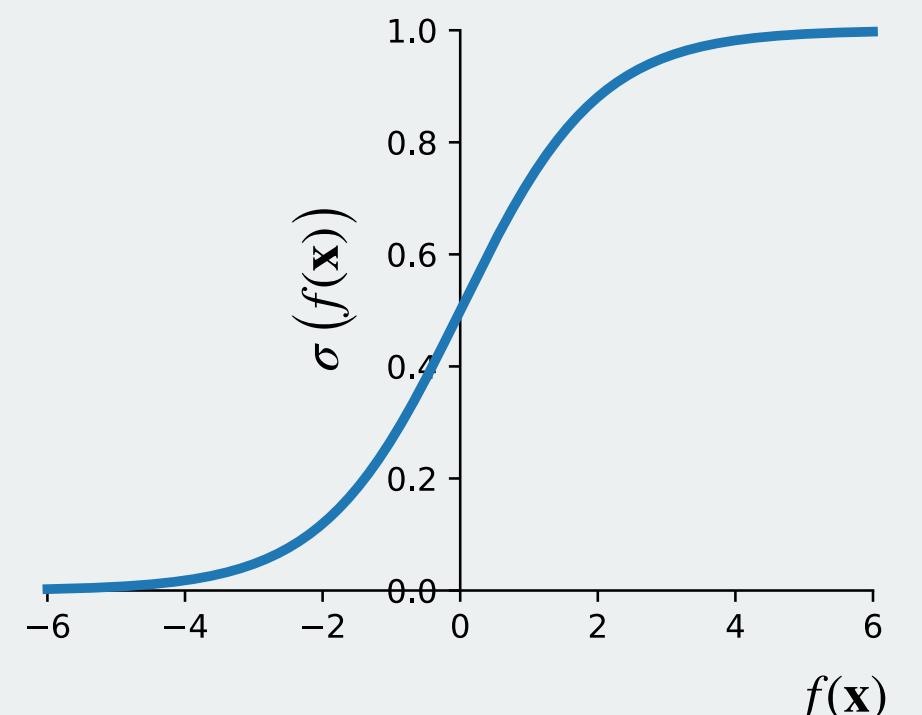
iLogistic regression! classifier

(Input)

$$w_0 + x_0w_1 + x_1w_2 + x_2w_3 = f(\mathbf{x})$$

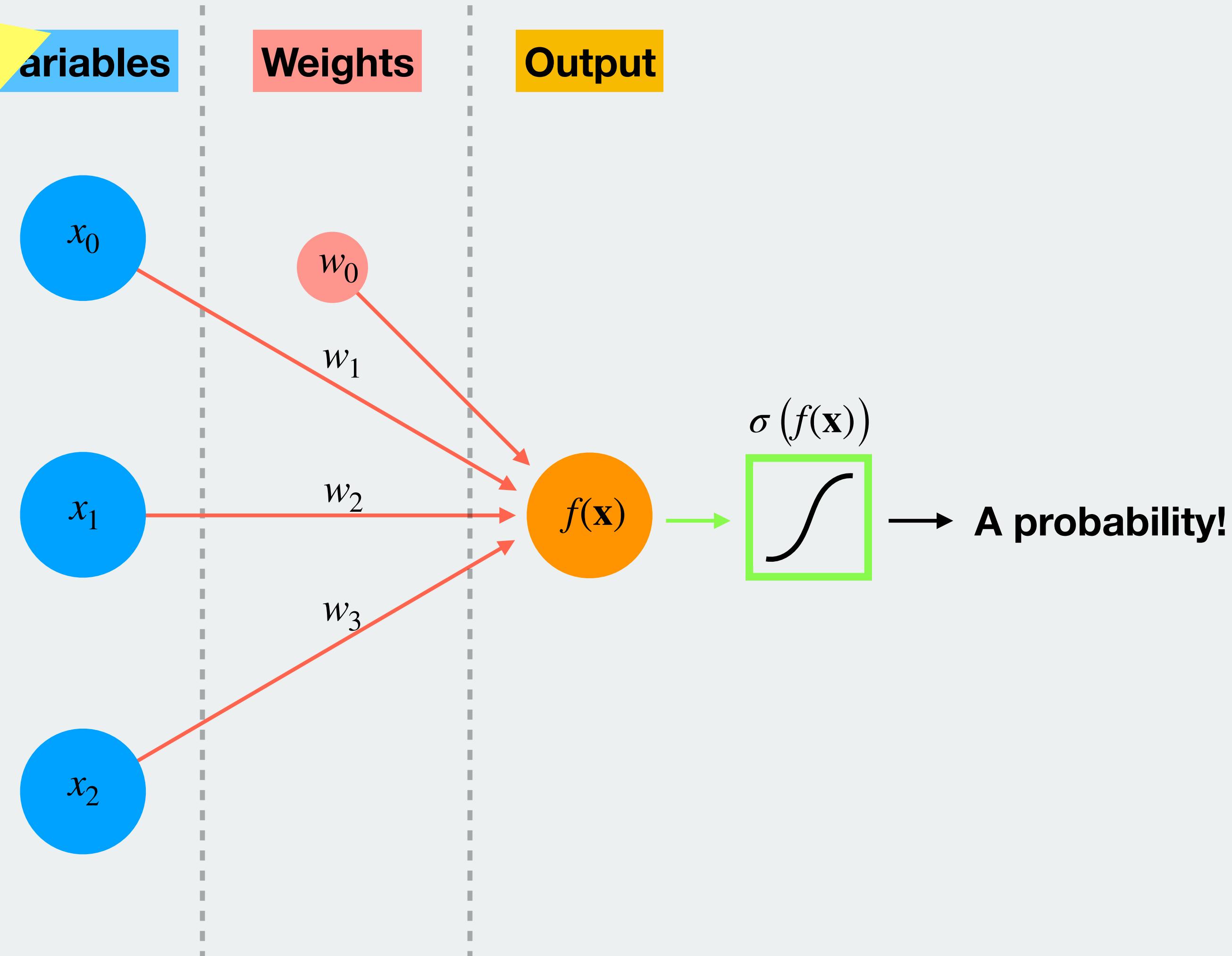
Output $\rightarrow \sigma(f(\mathbf{x})) = \frac{1}{1 + exp(-f(\mathbf{x}))}$

Activation function



iPerceptron!

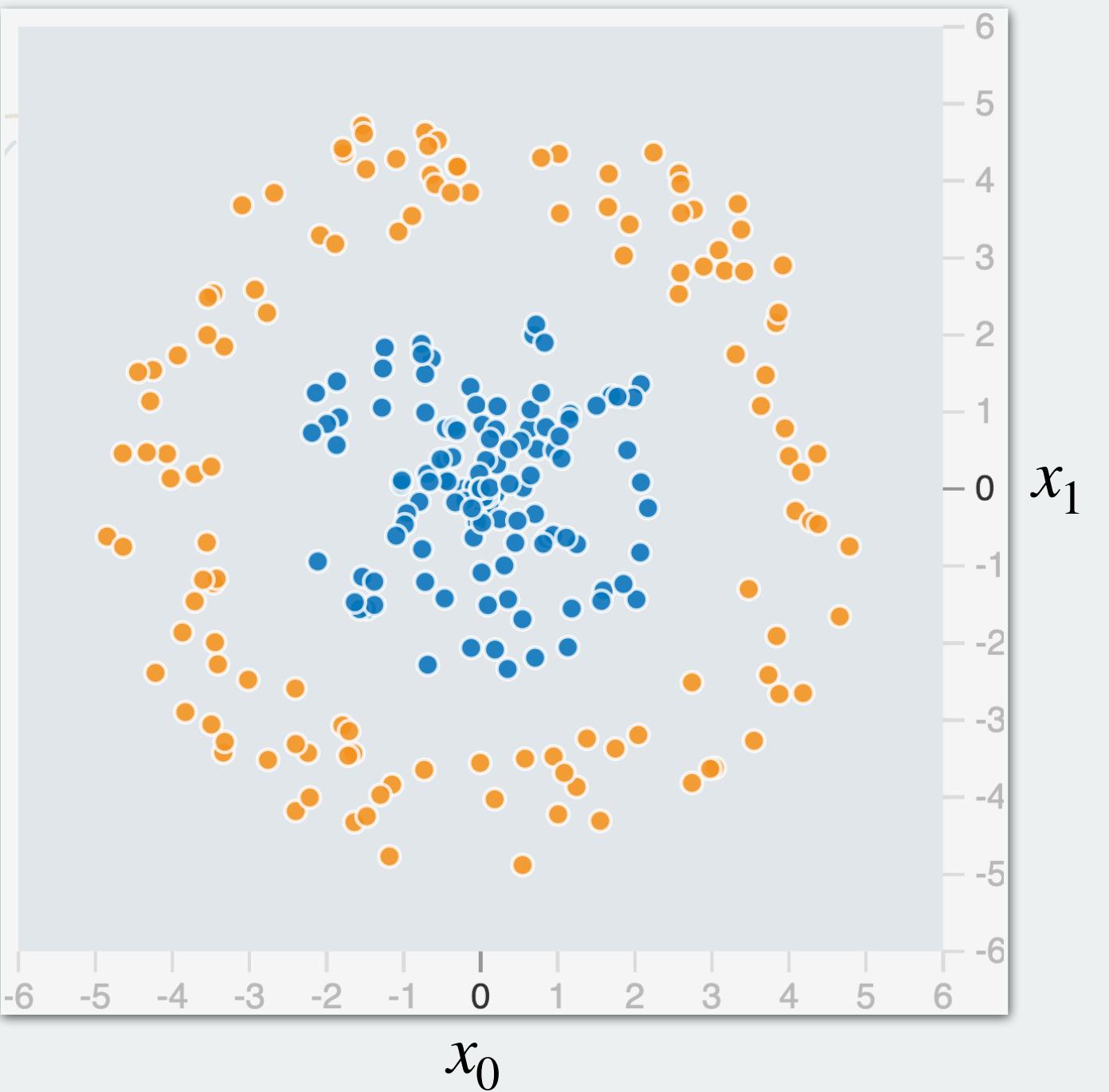
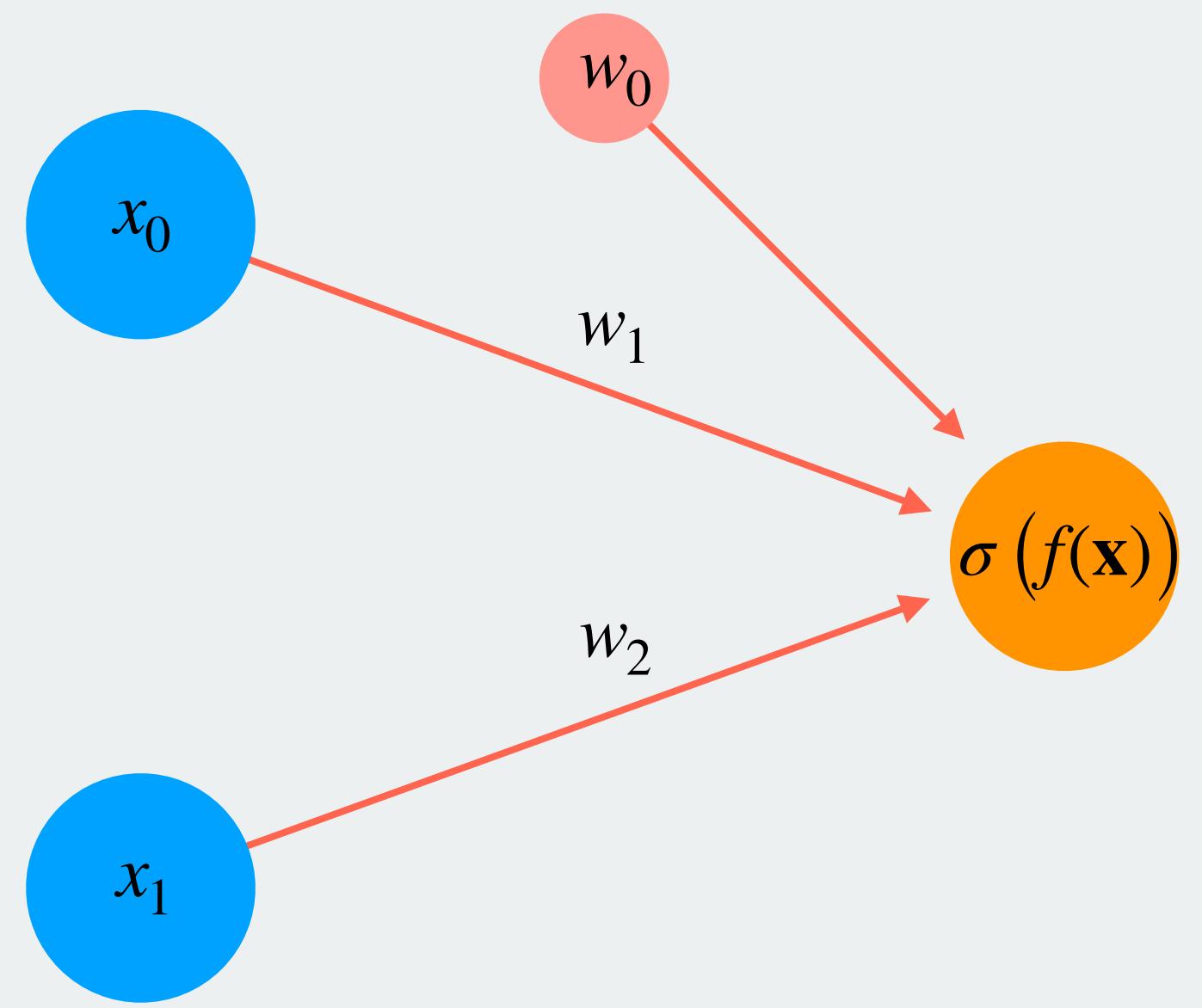
Invented by Frank Rosenblat in 1957



Logistic regression not so simple problem

> Find values of $\{w_0, w_1, w_2\}$ that minimizes $\sum_n (\tilde{y}_n - y_n)^2$

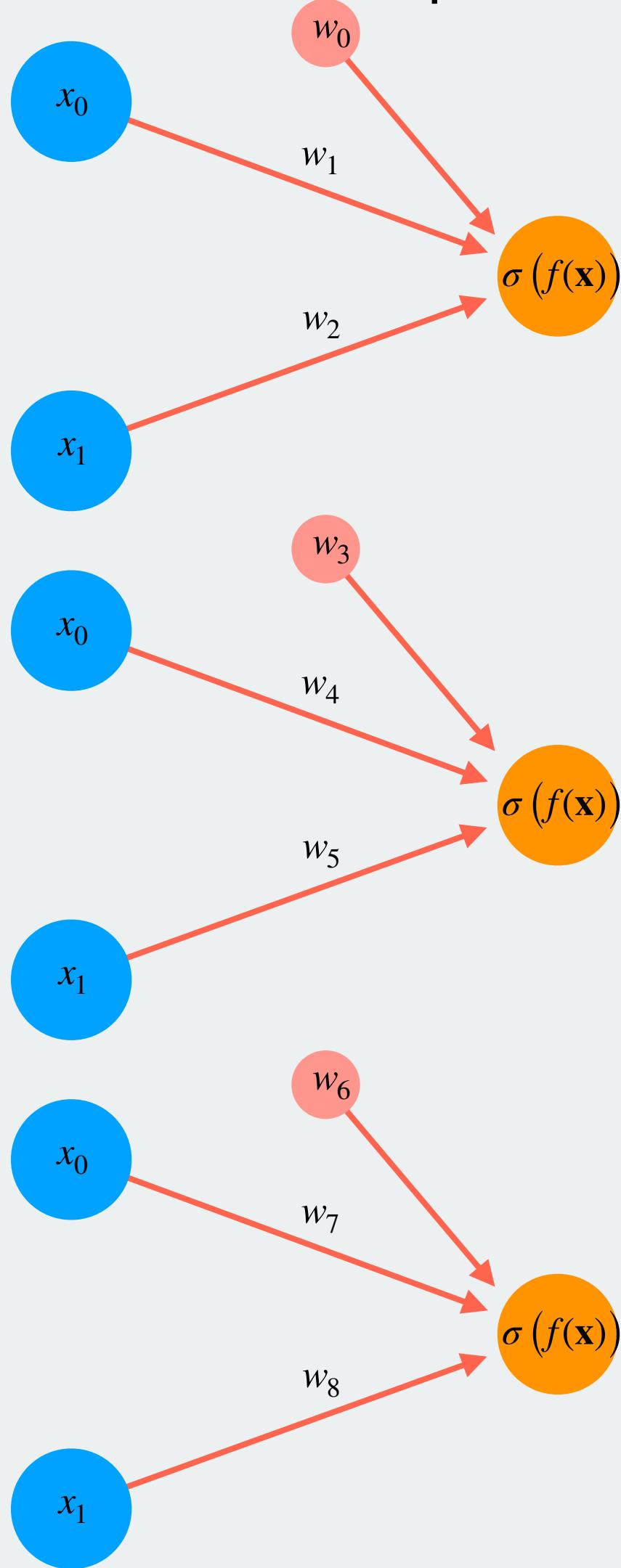
Model too simple



Reminder: $w_0 + x_0 w_1 + x_1 w_2 = f(\mathbf{x})$

Logistic regression not so simple problem

> Solution: Break problem into subproblems



$$\sigma(w_0 + x_0 w_1 + x_1 w_2) = z_0(\mathbf{x})$$

$$\sigma(w_3 + x_0 w_4 + x_1 w_5) = z_1(\mathbf{x})$$

$$\sigma(w_6 + x_0 w_7 + x_1 w_8) = z_2(\mathbf{x})$$

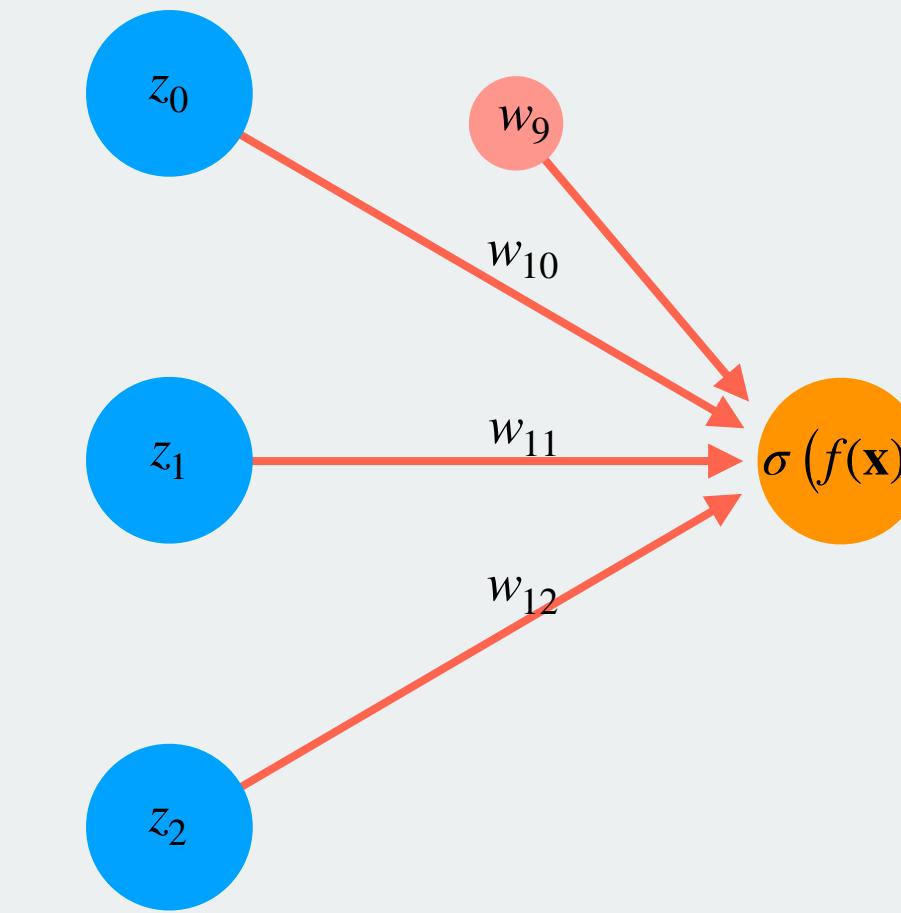
New problem: Given \mathbf{Z} , predict \mathbf{y}

z_0	z_1	z_2	y
0.3	0.75	0.78	0
0.25	0.1	0.95	1
...
0.79	0.99	0.3	1
0.34	0.6	0.1	0

$\mathbf{Z} =$

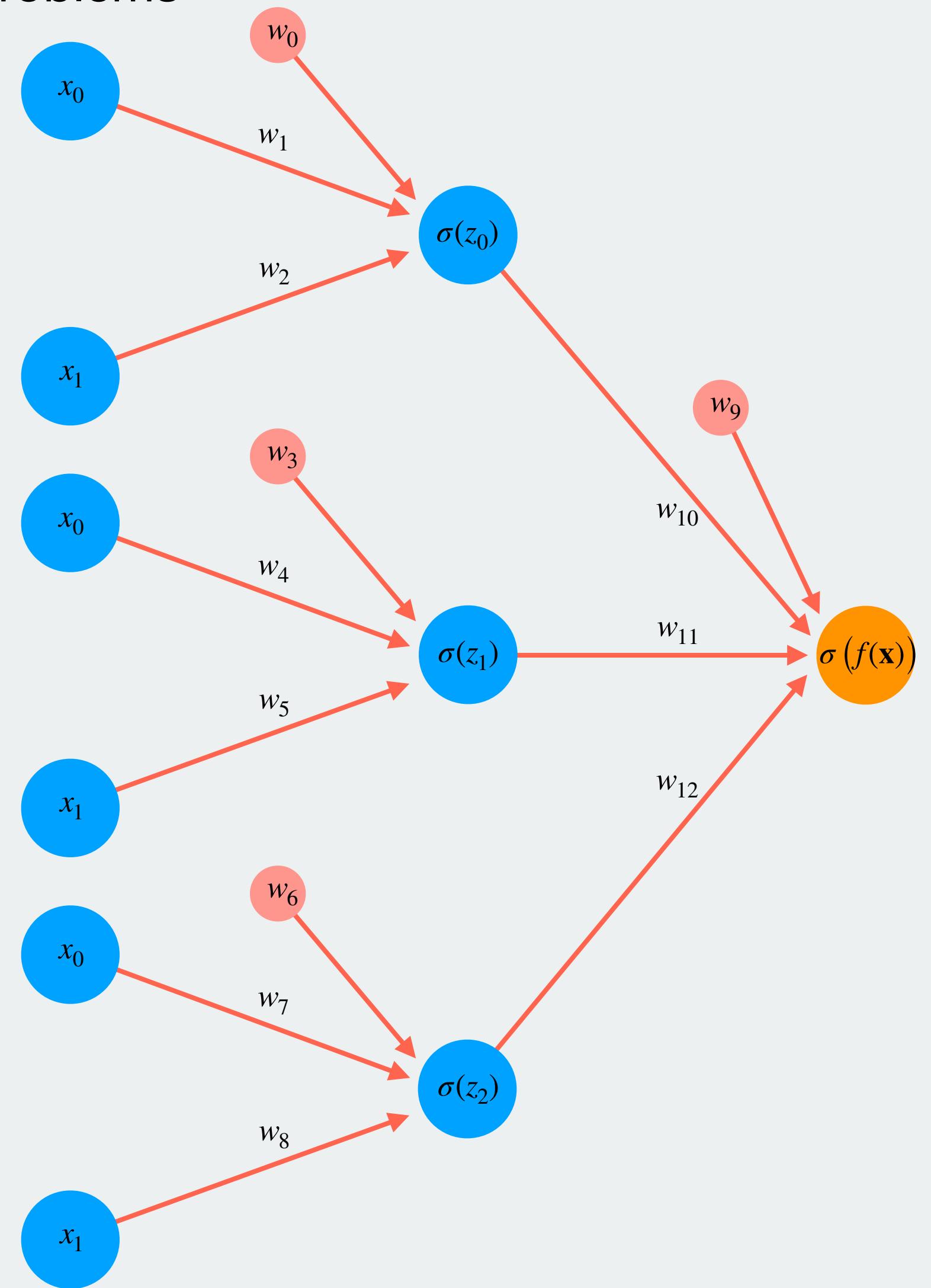
$\mathbf{y} =$

Solution: Why not use a logistic regression?



Logistic regression not so simple problem

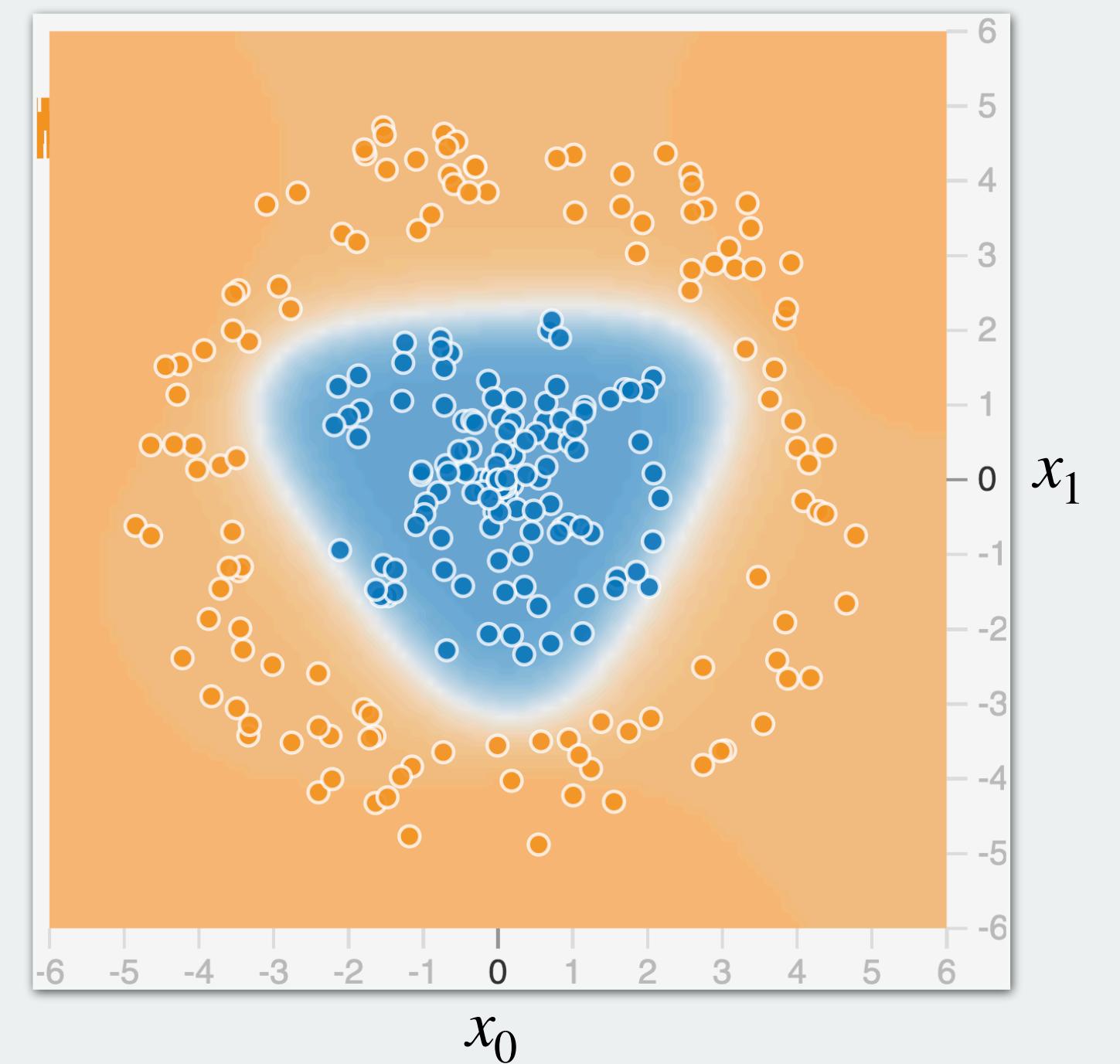
> Solution: Connect subproblems



Mathematical form

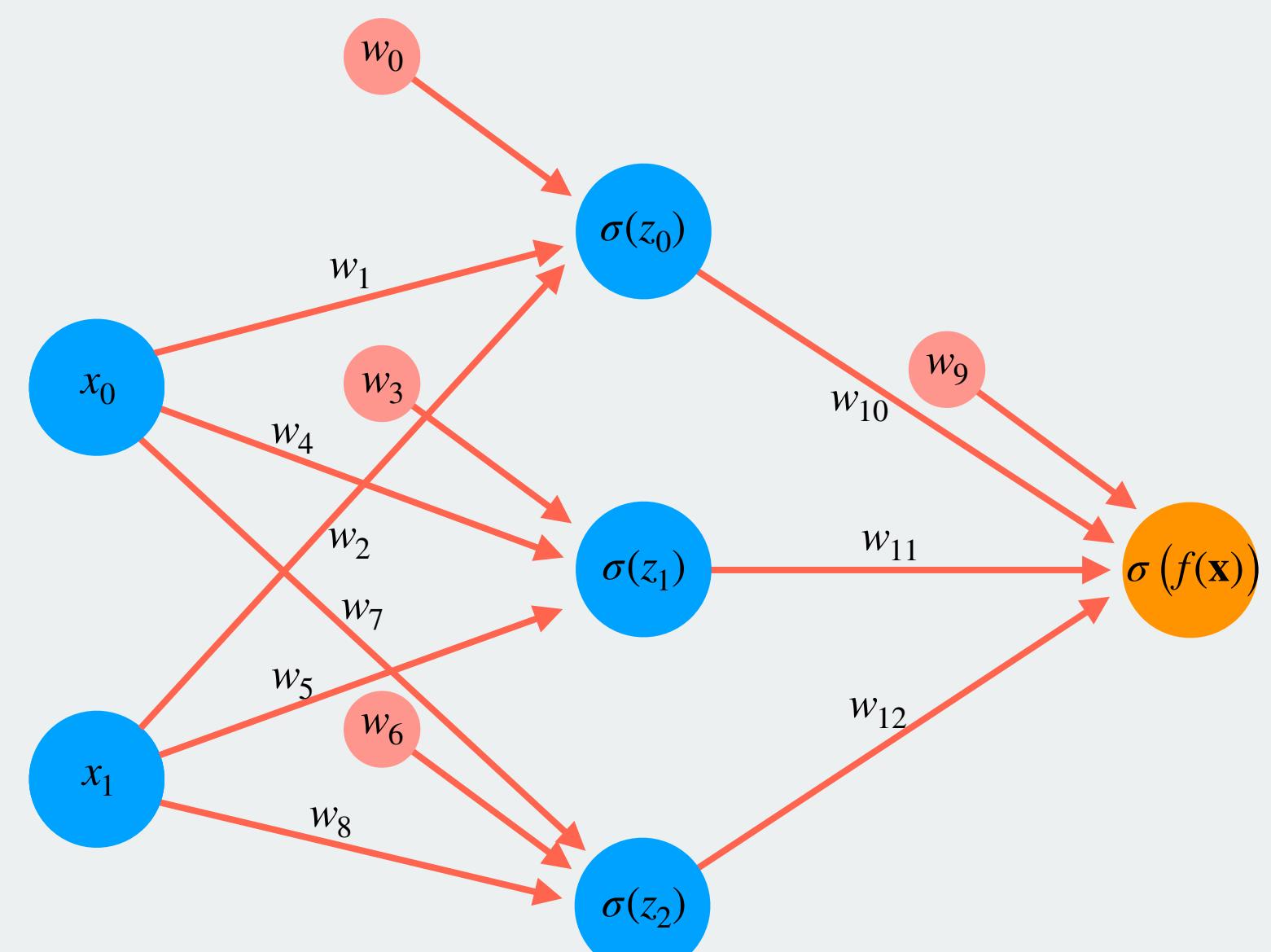
$$\begin{aligned}\sigma(w_9 + \sigma(w_0 + x_0 w_1 + x_1 w_2)w_{10} + \\ \sigma(w_3 + x_0 w_4 + x_1 w_5)w_{11} + \\ \sigma(w_6 + x_0 w_7 + x_1 w_8)w_{12}) = \sigma(f(\mathbf{x}))\end{aligned}$$

Solution



Logistic regression not so simple problem

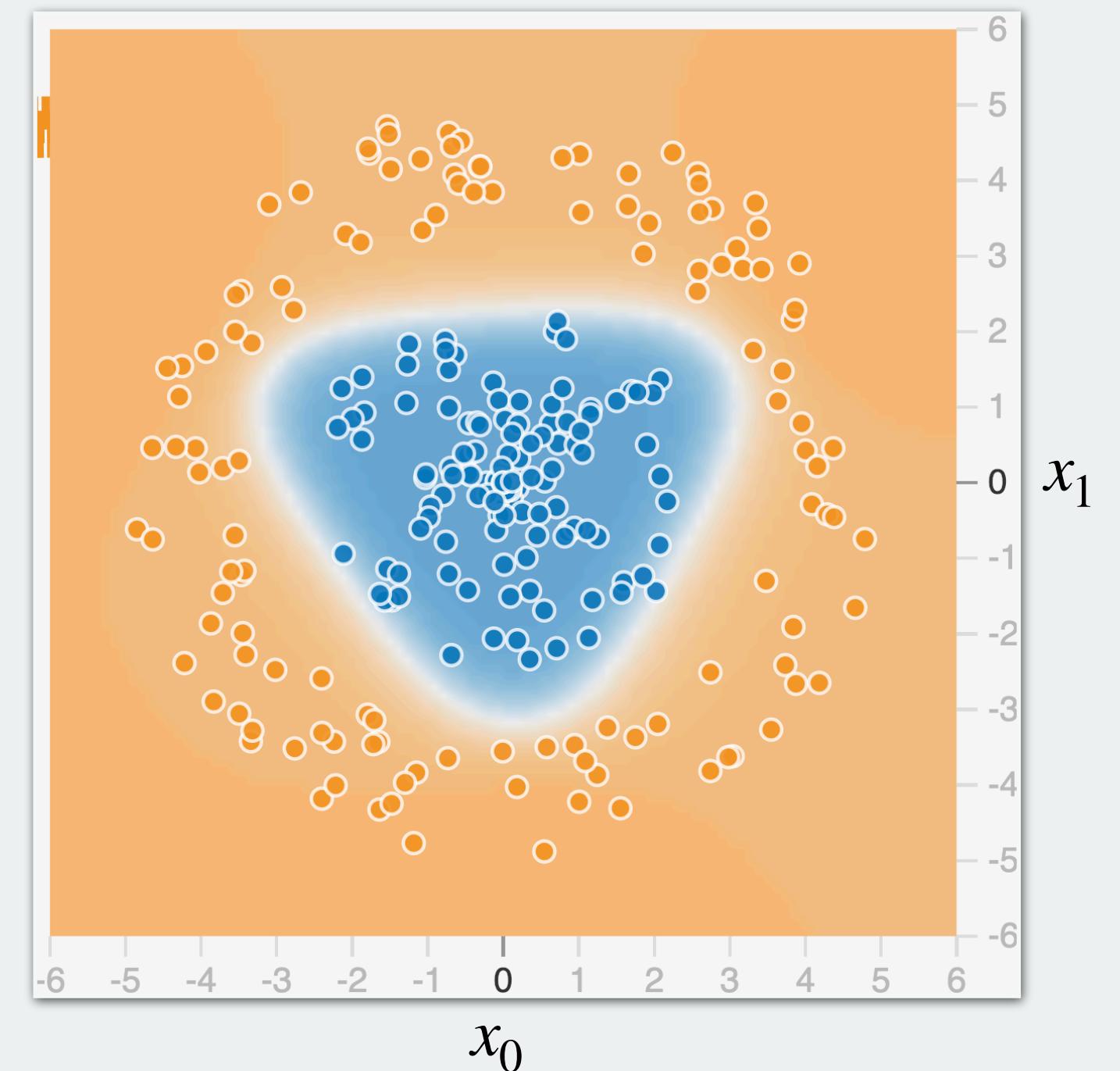
> Solution: Connect subproblems



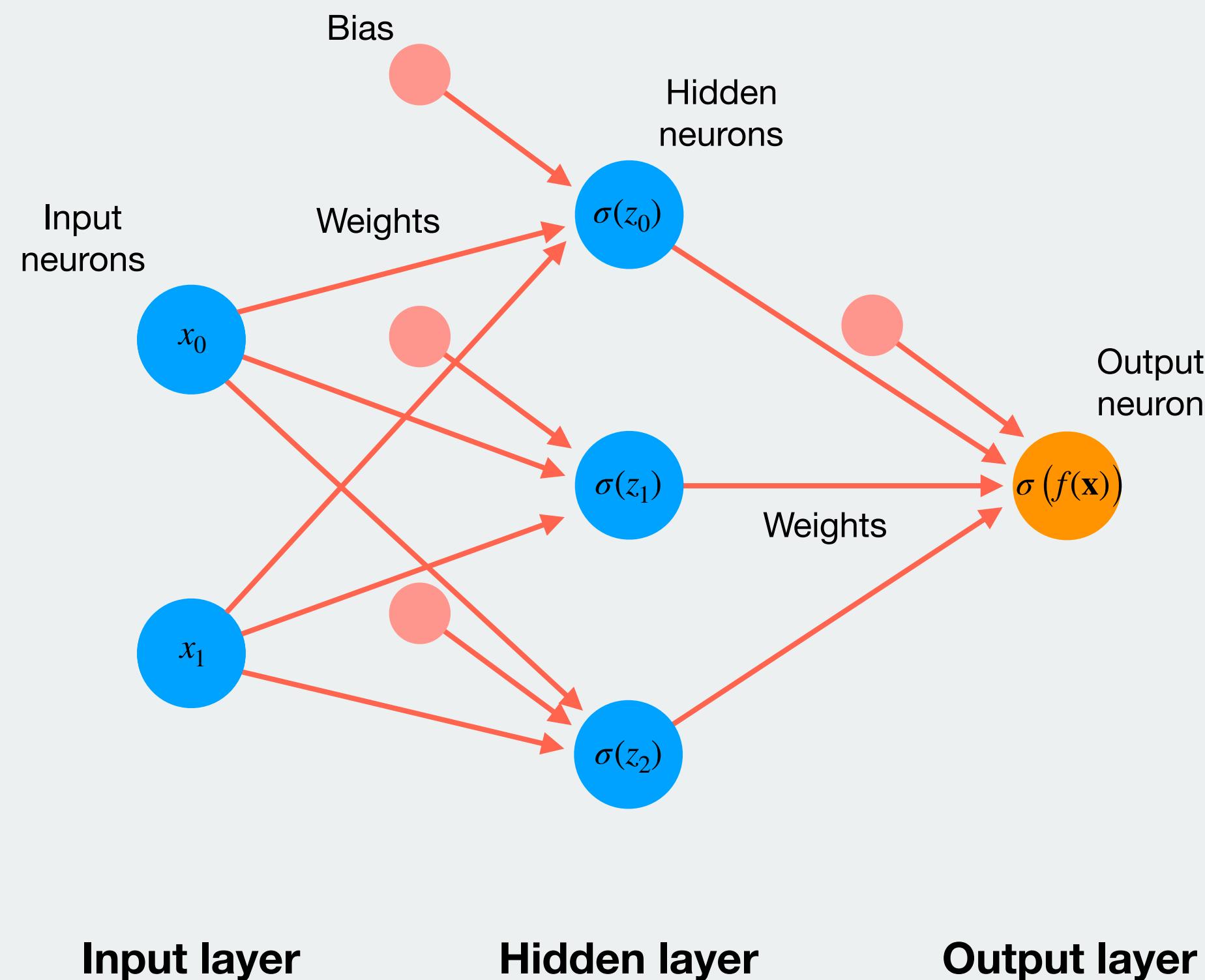
Mathematical form

$$\begin{aligned}\sigma(w_9 + \sigma(w_0 + x_0 w_1 + x_1 w_2) w_{10}) + \\ \sigma(w_3 + x_0 w_4 + x_1 w_5) w_{11} + \\ \sigma(w_6 + x_0 w_7 + x_1 w_8) w_{12}) = \sigma(f(\mathbf{x}))\end{aligned}$$

Solution



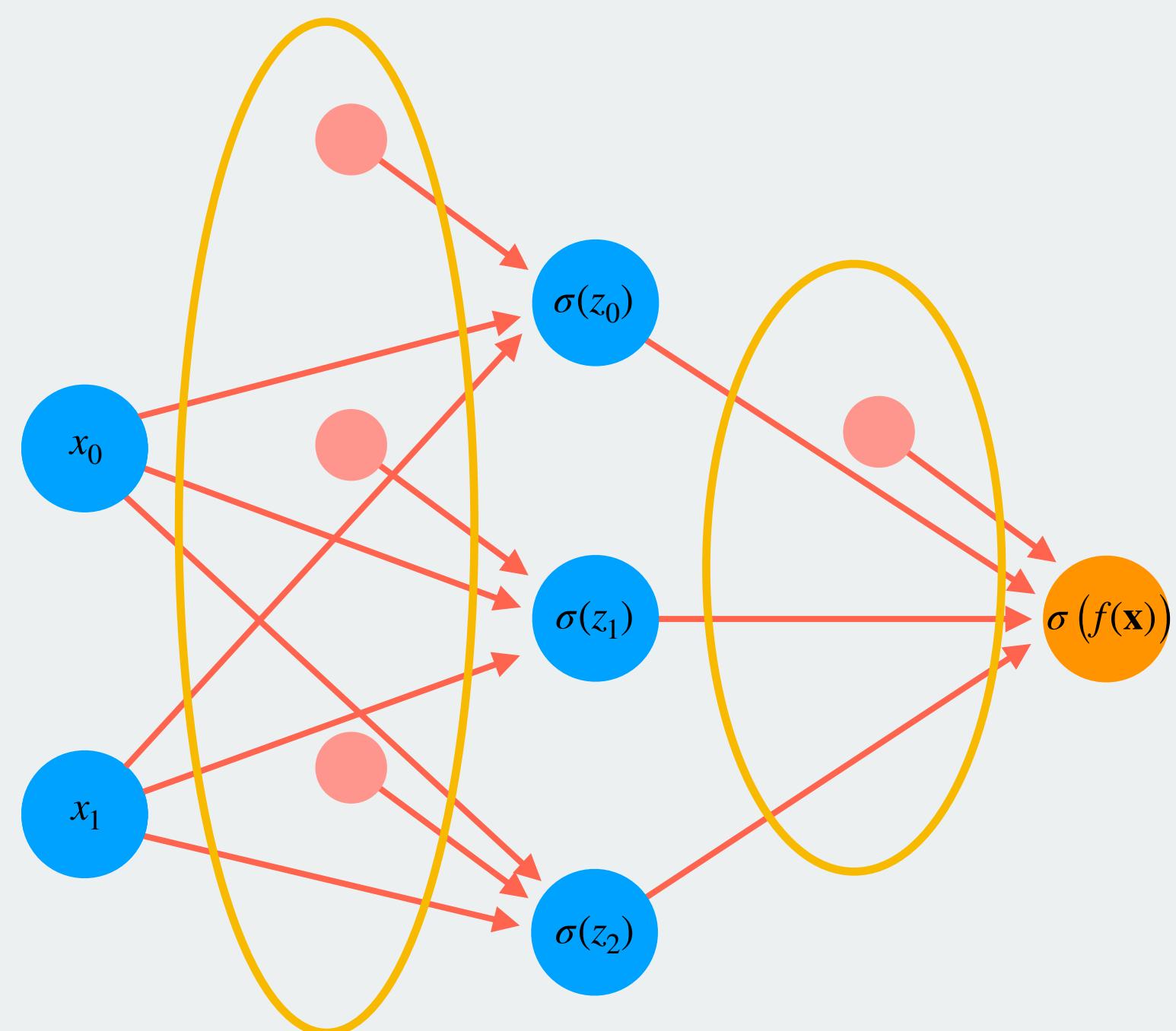
Multilayer perceptron – the structure



$$\sigma(\mathbf{b}_n + \mathbf{W}_n \mathbf{a}_{n-1}) = \mathbf{a}_n$$

TODAY: How neural networks learn

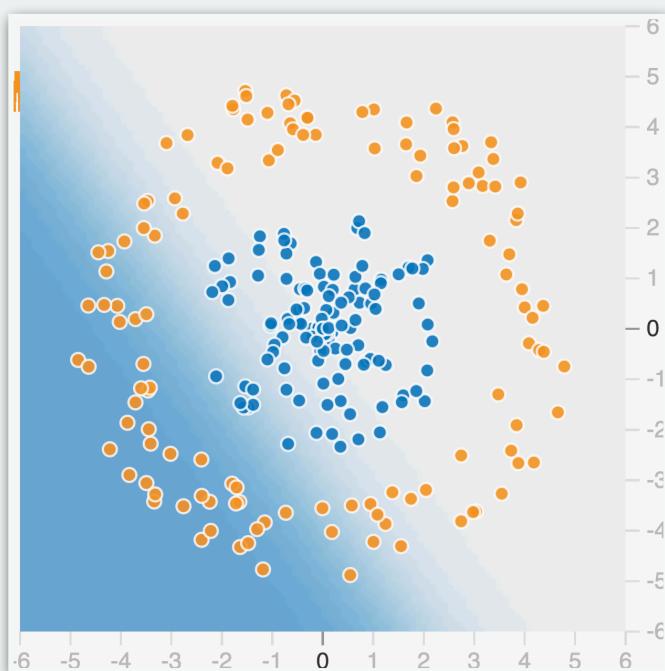
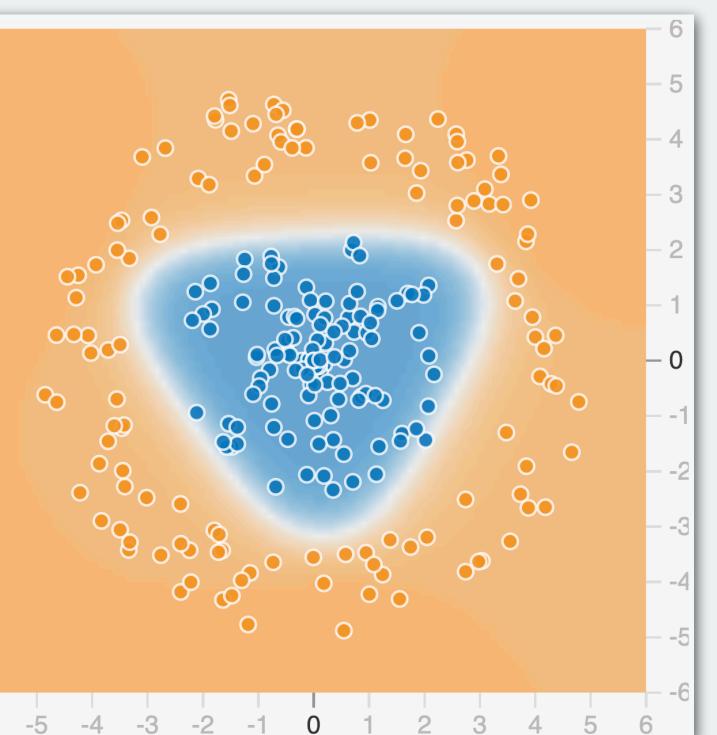
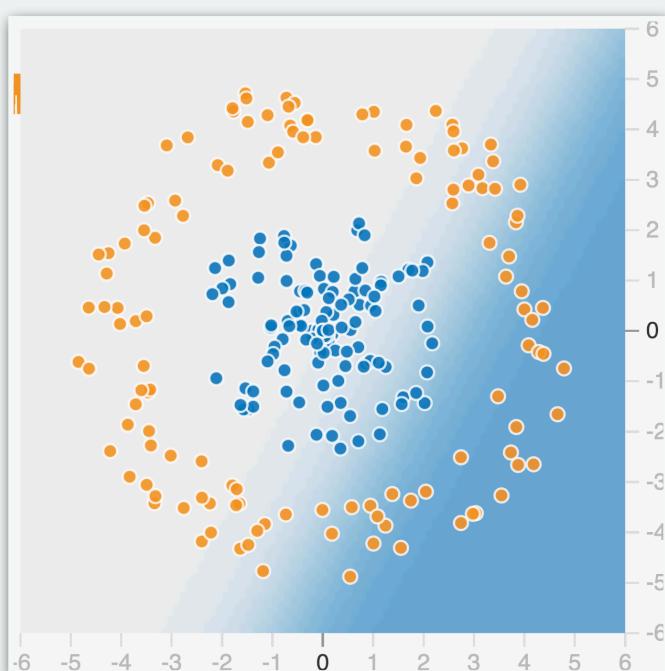
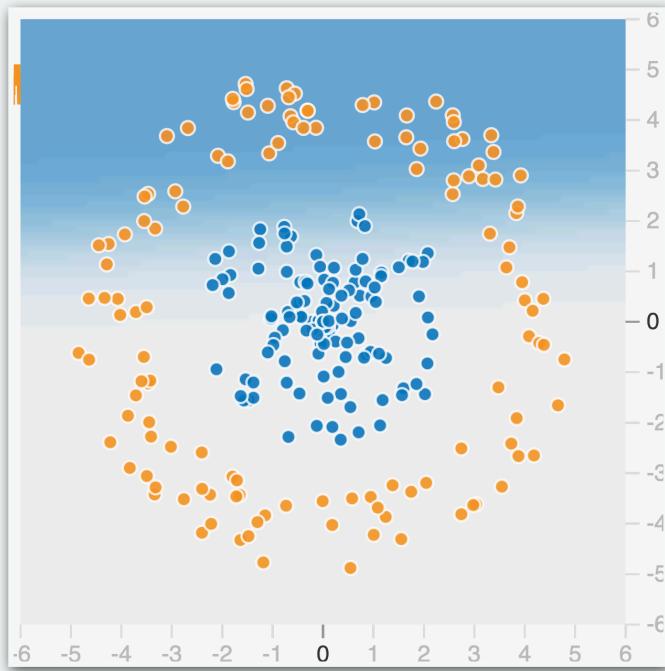
Specifically: how to **set weights** so a network makes **good predictions**



STEPS:

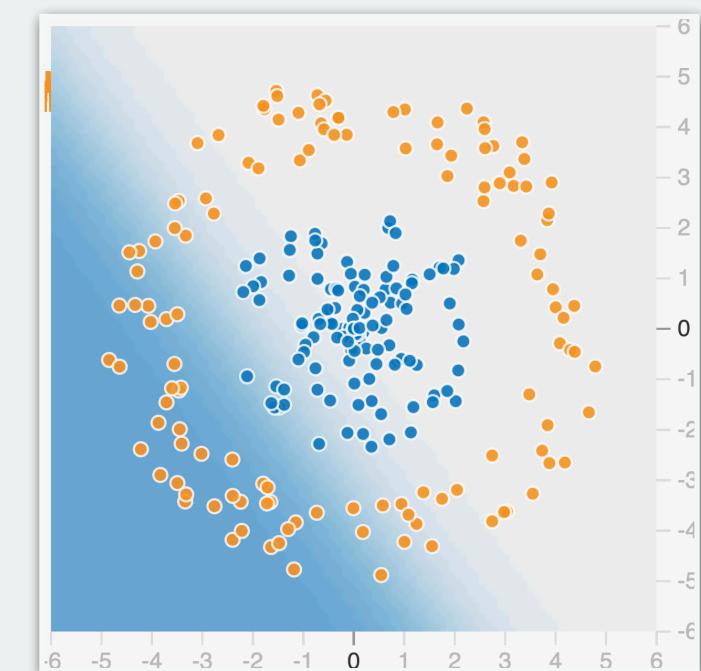
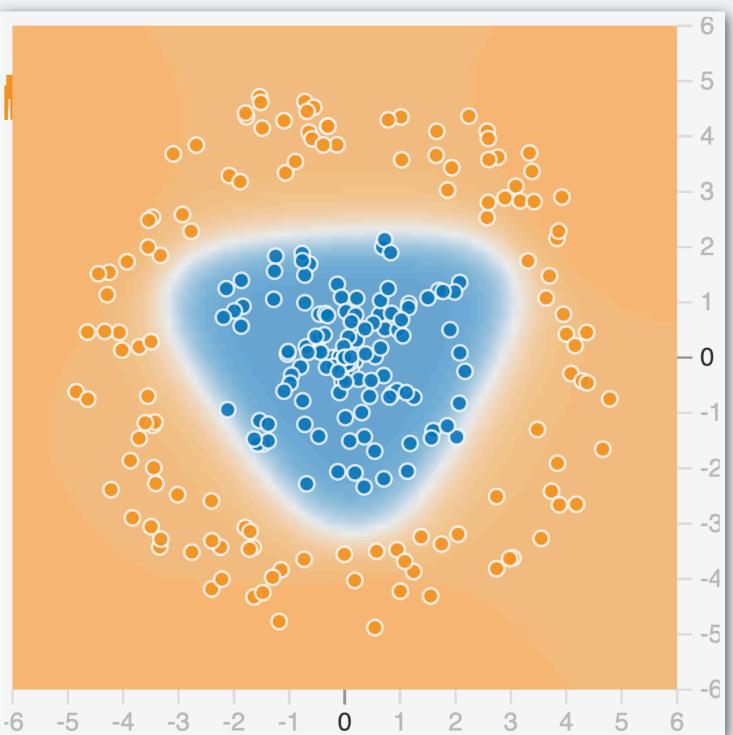
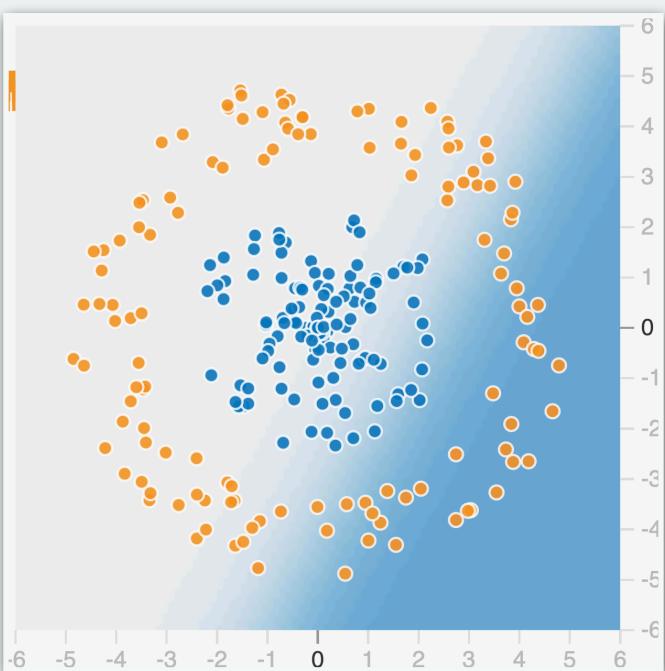
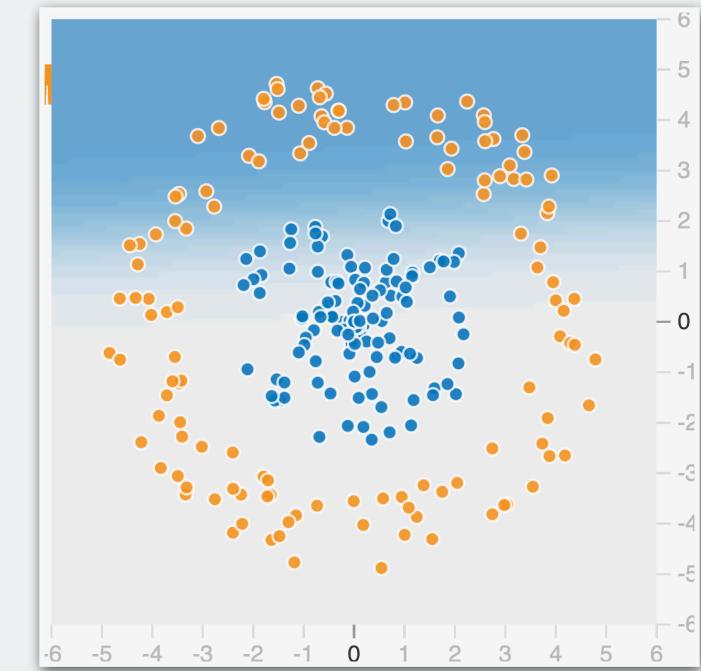
1. Gradient descent
2. Backpropagation

How do we find the weights that give the best classification?



example

How do we find the weights that give the best classification?



predicted from \mathbf{X} given \mathbf{W}

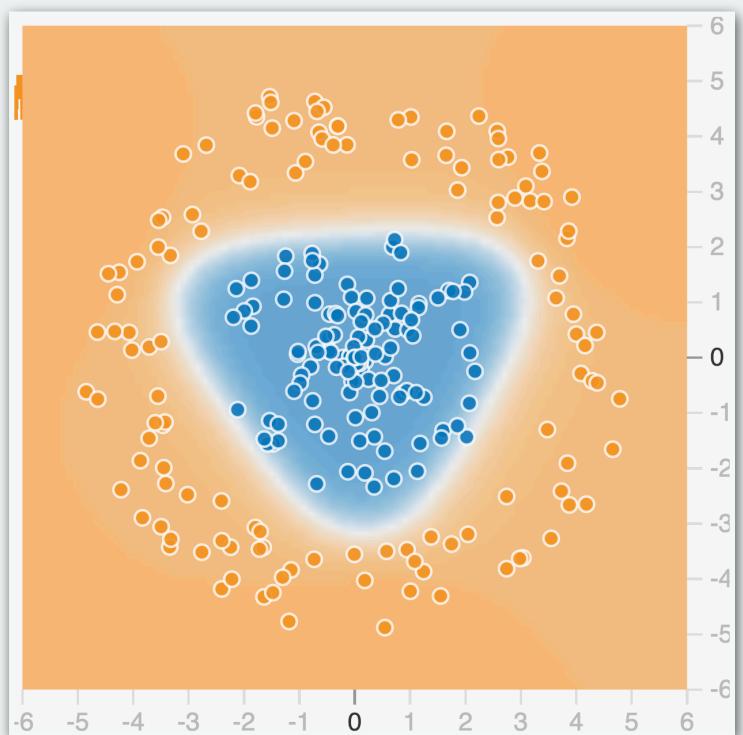
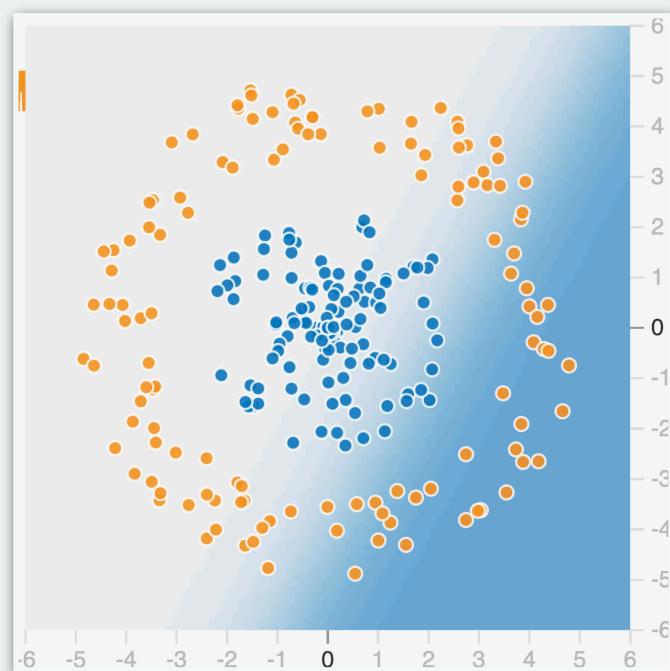
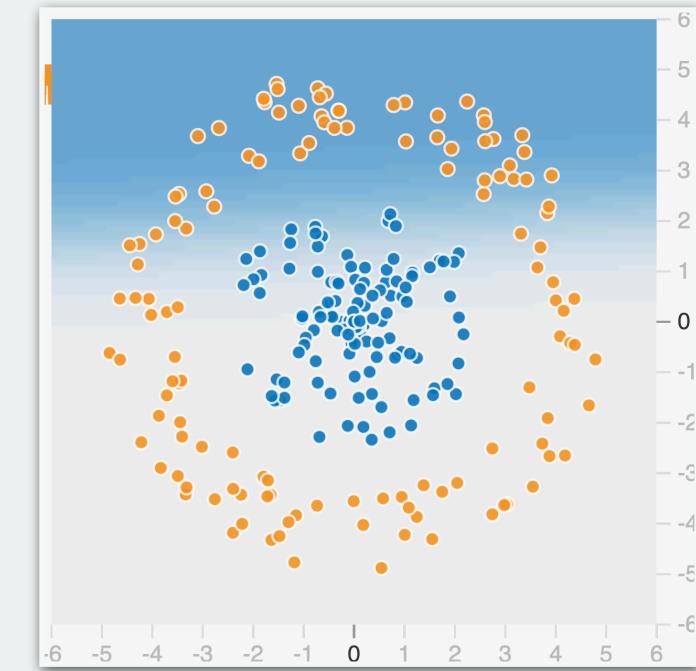
$$\tilde{\mathbf{y}} = \begin{bmatrix} 0.96 \\ 0.10 \\ 0.04 \\ \dots \\ 0.70 \\ 0.02 \\ 0.99 \end{bmatrix}$$

true

$$\mathbf{y} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \dots \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

$$\begin{aligned} C(\mathbf{W}) &= \frac{1}{N} \sum_i (\tilde{y}_i - y_i)^2 \\ &= (0.96 - 1)^2 \\ &\quad + (0.10 - 0)^2 \\ &\quad + (0.04 - 0)^2 \\ &\quad + \dots \\ &\quad + (0.70 - 1)^2 \\ &\quad + (0.02 - 0)^2 \\ &\quad + (0.99 - 1)^2 \end{aligned}$$

How do we find the weights that give the best classification?



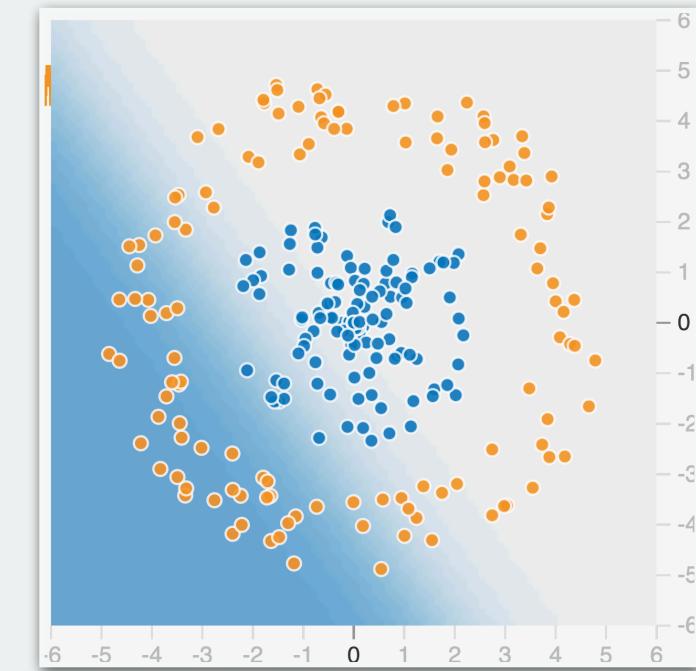
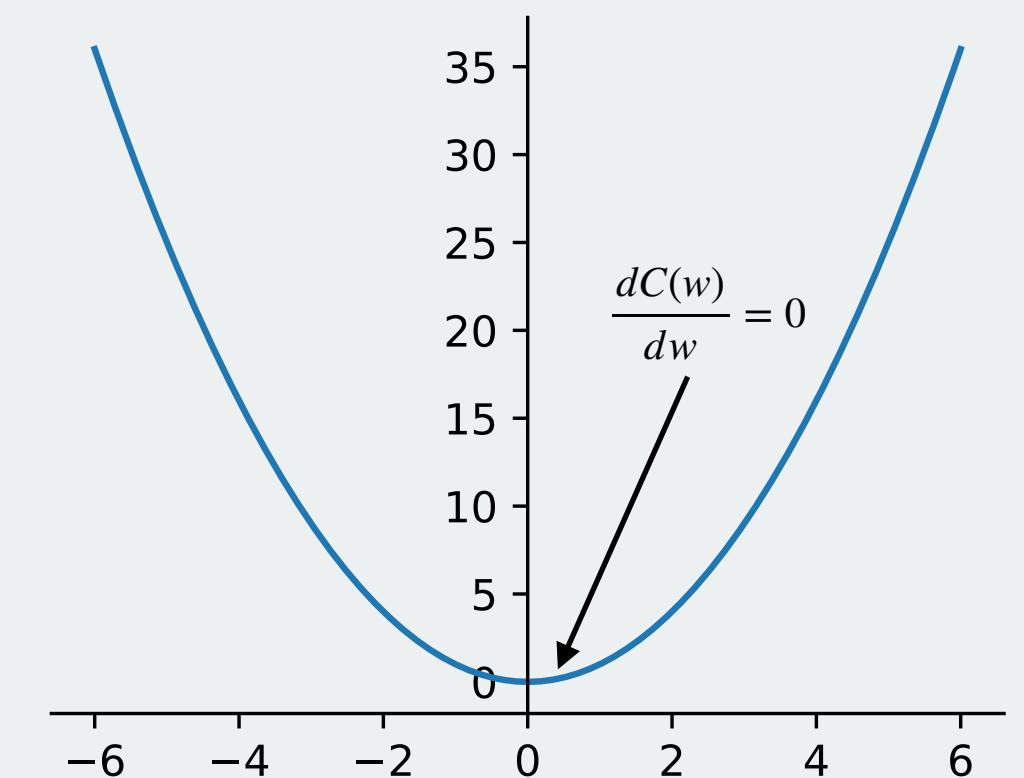
predicted from \mathbf{X} given \mathbf{W}

$$\tilde{\mathbf{y}} = \begin{bmatrix} 0.96 \\ 0.10 \\ 0.04 \\ \dots \\ 0.70 \\ 0.02 \\ 0.99 \end{bmatrix}$$

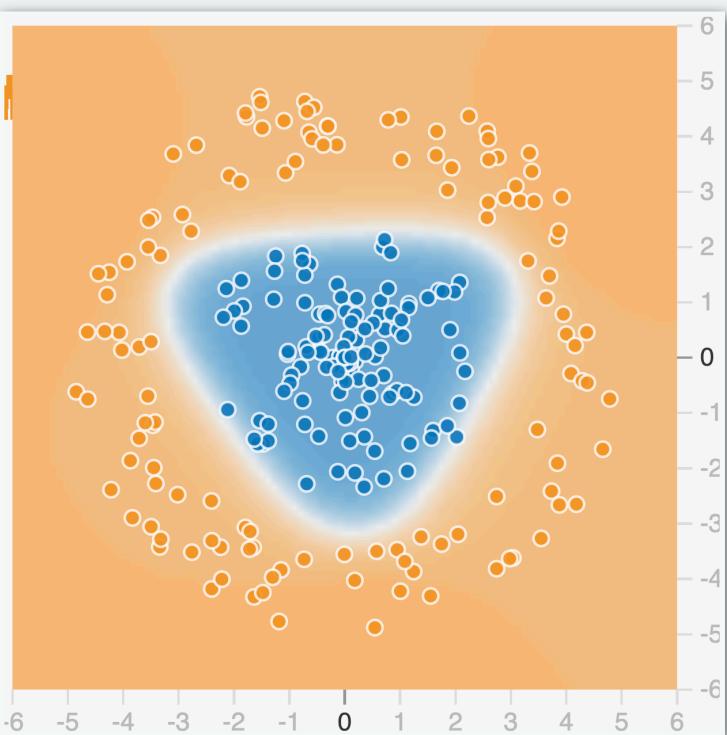
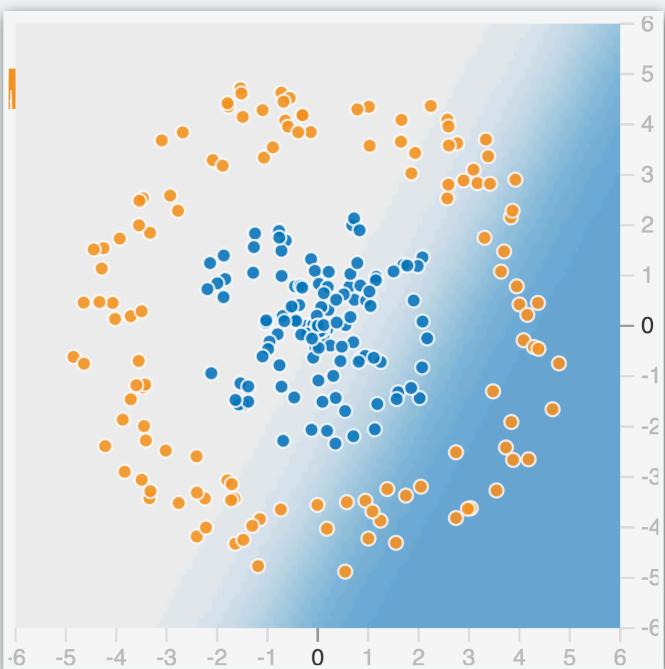
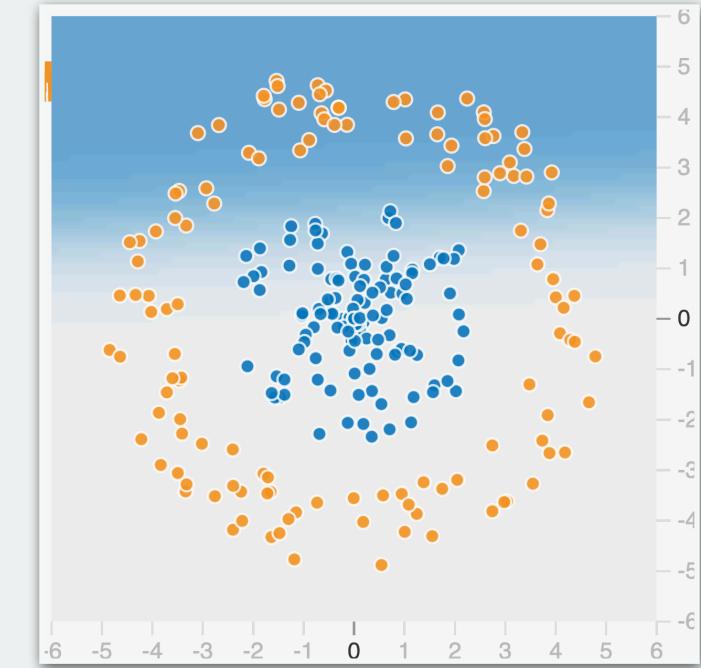
true

$$\mathbf{y} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \dots \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

$$\begin{aligned} C(\mathbf{W}) &= \frac{1}{N} \sum_i (\tilde{y}_i - y_i)^2 \\ &= (0.96 - 1)^2 \\ &\quad + (0.10 - 0)^2 \\ &\quad + (0.04 - 0)^2 \\ &\quad + \dots \\ &\quad + (0.70 - 1)^2 \\ &\quad + (0.02 - 0)^2 \\ &\quad + (0.99 - 1)^2 \end{aligned}$$



How do we find the weights that give the best classification?



example

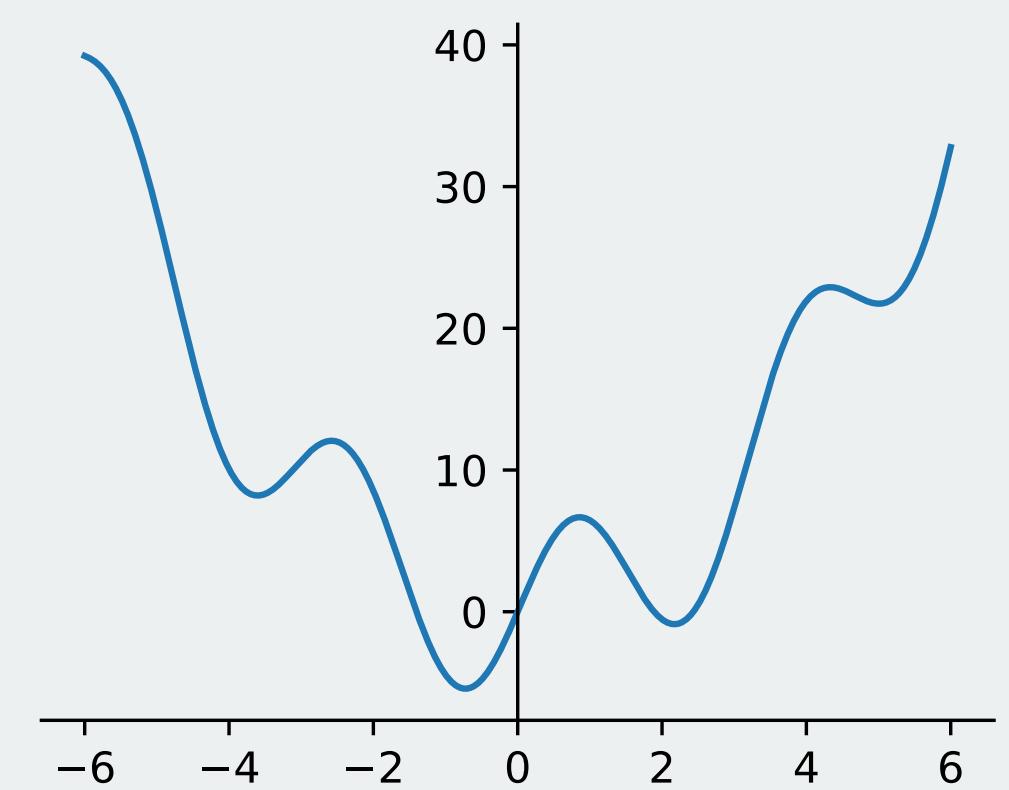
predicted from X given W

$$\tilde{\mathbf{y}} = \begin{bmatrix} 0.96 \\ 0.10 \\ 0.04 \\ \dots \\ 0.70 \\ 0.02 \\ 0.99 \end{bmatrix}$$

true

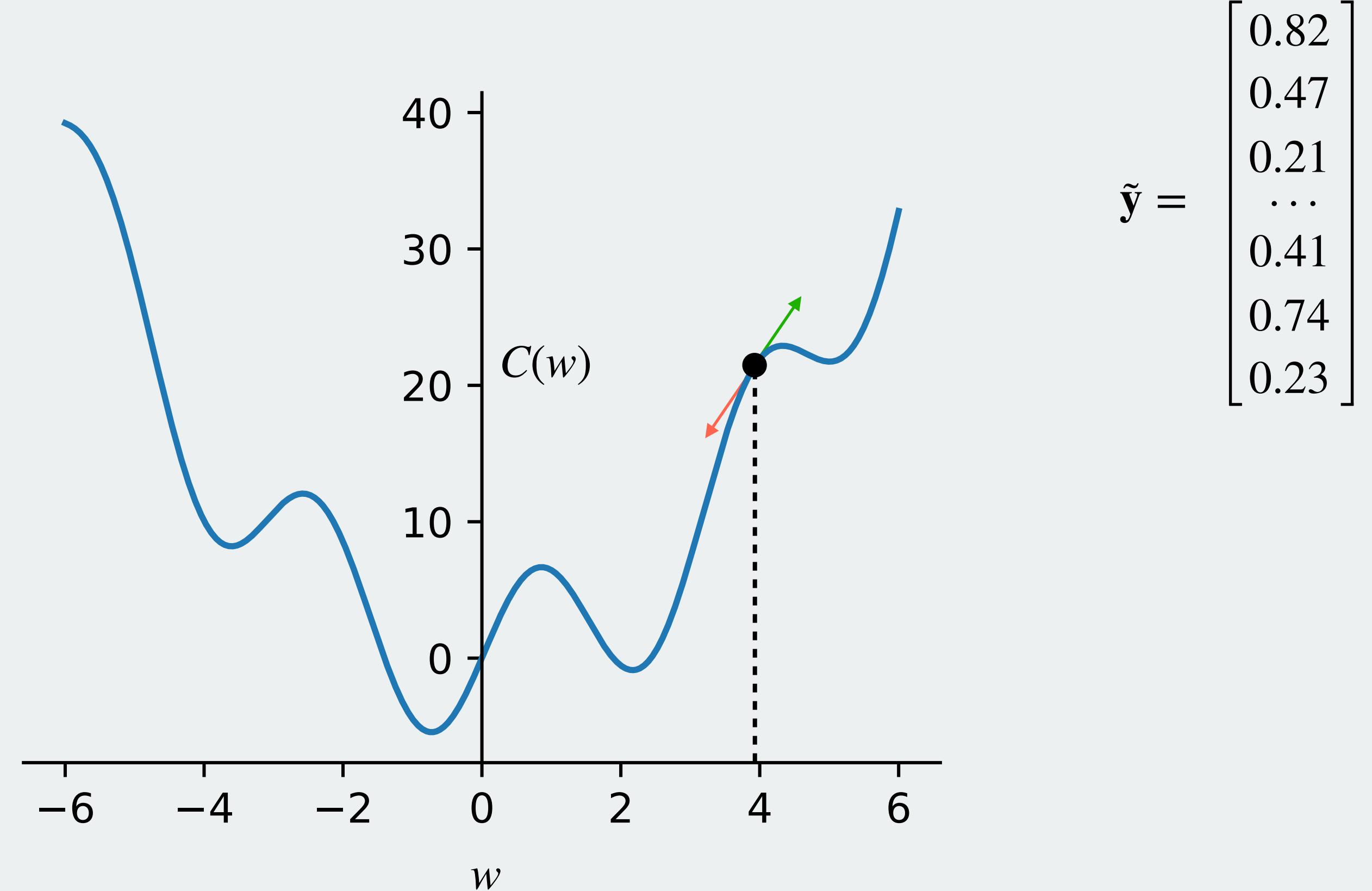
$$\mathbf{y} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \dots \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

$$\begin{aligned} C(\mathbf{W}) &= \frac{1}{N} \sum_i (\tilde{y}_i - y_i)^2 \\ &= (0.96 - 1)^2 \\ &\quad + (0.10 - 0)^2 \\ &\quad + (0.04 - 0)^2 \\ &\quad + \dots \\ &\quad + (0.70 - 1)^2 \\ &\quad + (0.02 - 0)^2 \\ &\quad + (0.99 - 1)^2 \end{aligned}$$



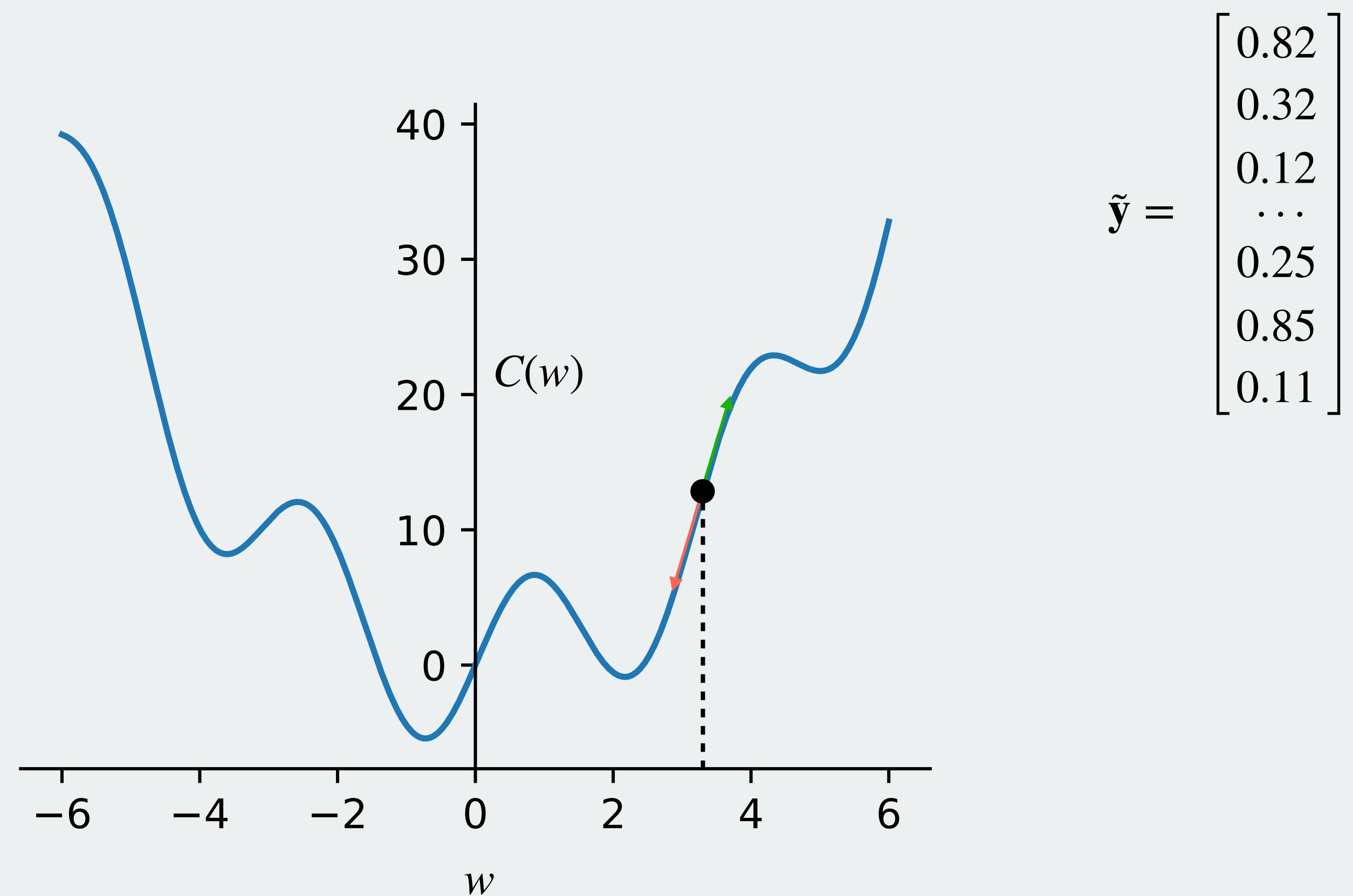
How do we find the weights that give the best classification?

> Gradient Descent



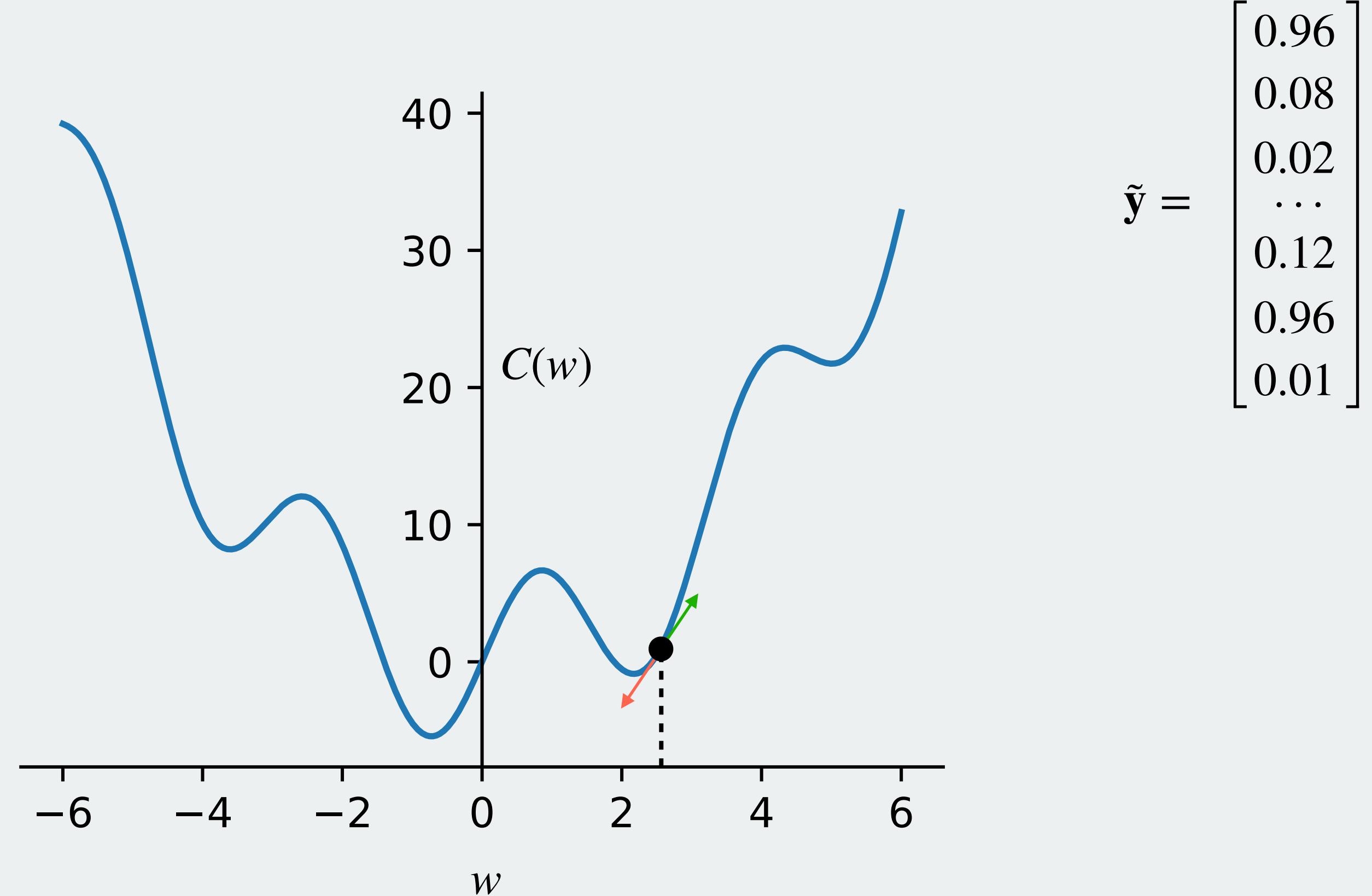
How do we find the weights that give the best classification?

> Gradient Descent



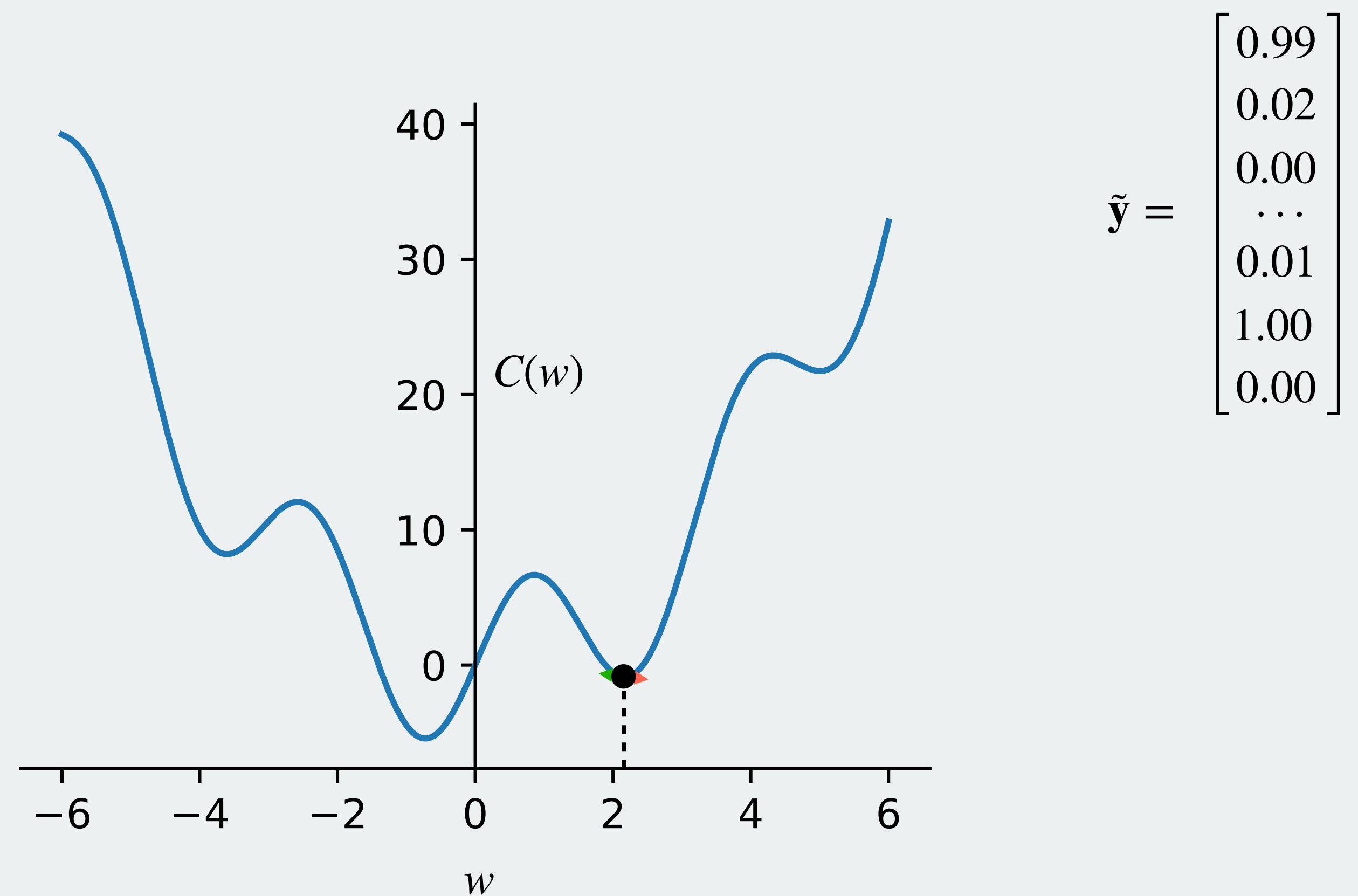
How do we find the weights that give the best classification?

> Gradient Descent



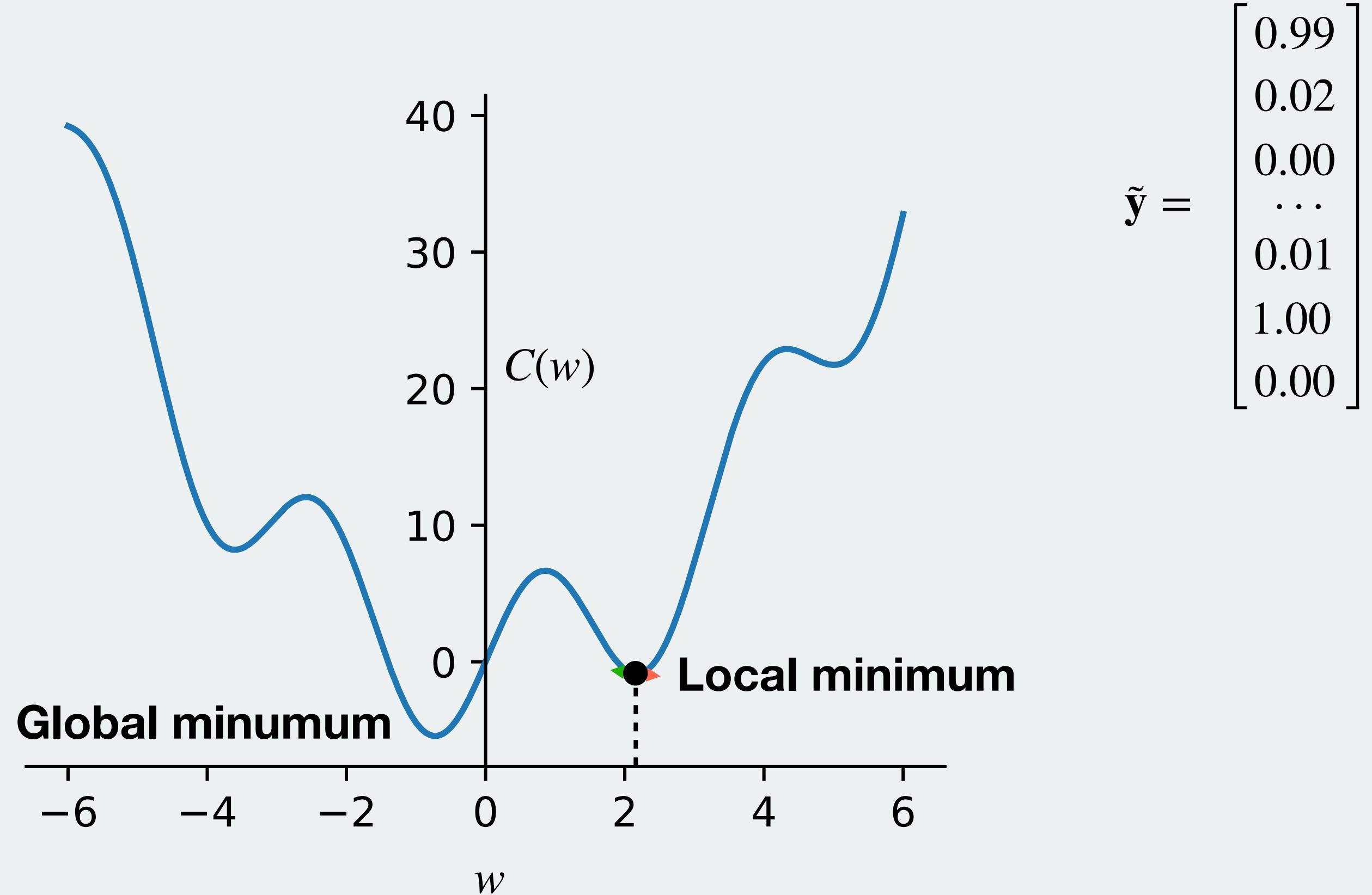
How do we find the weights that give the best classification?

> Gradient Descent



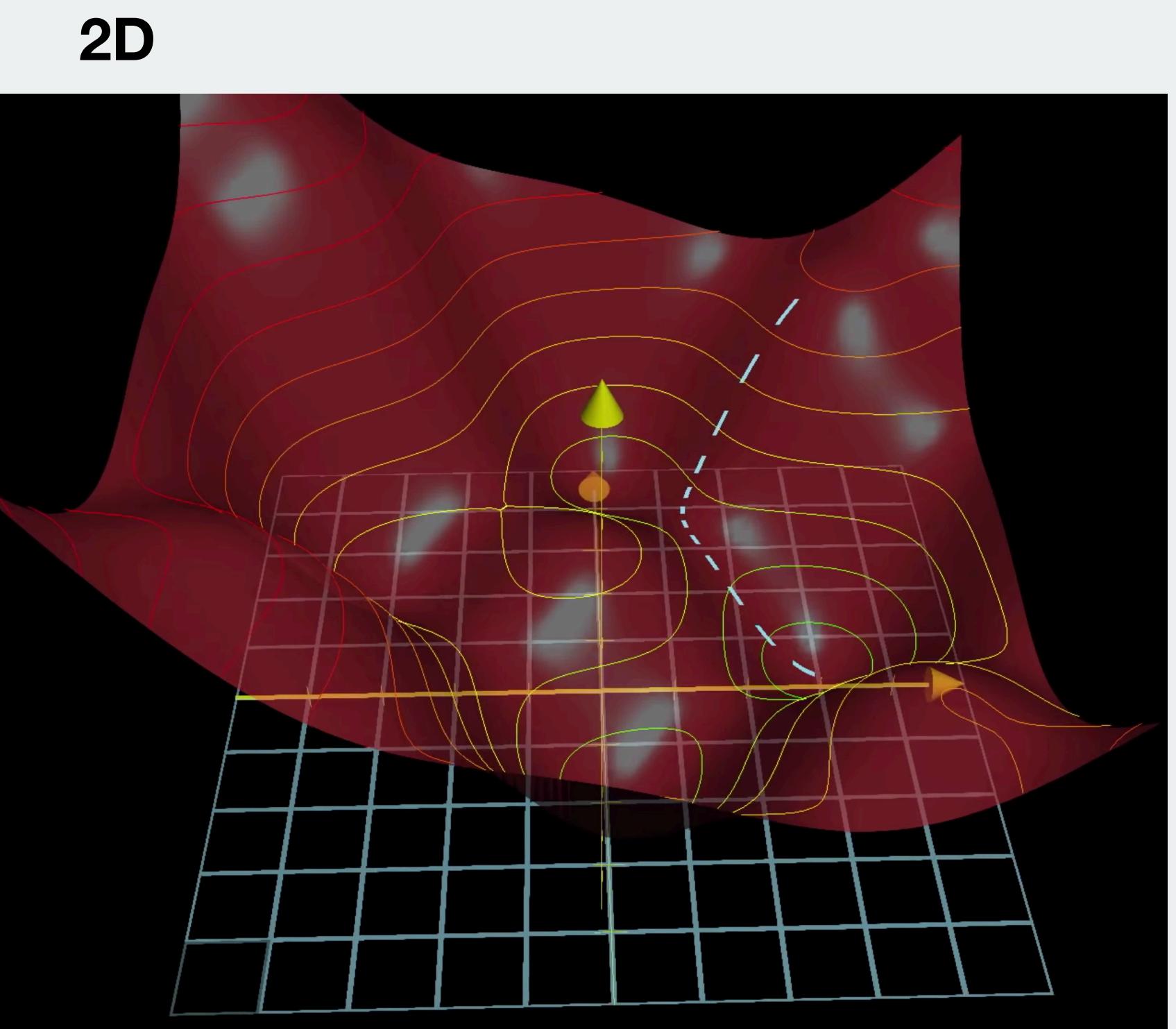
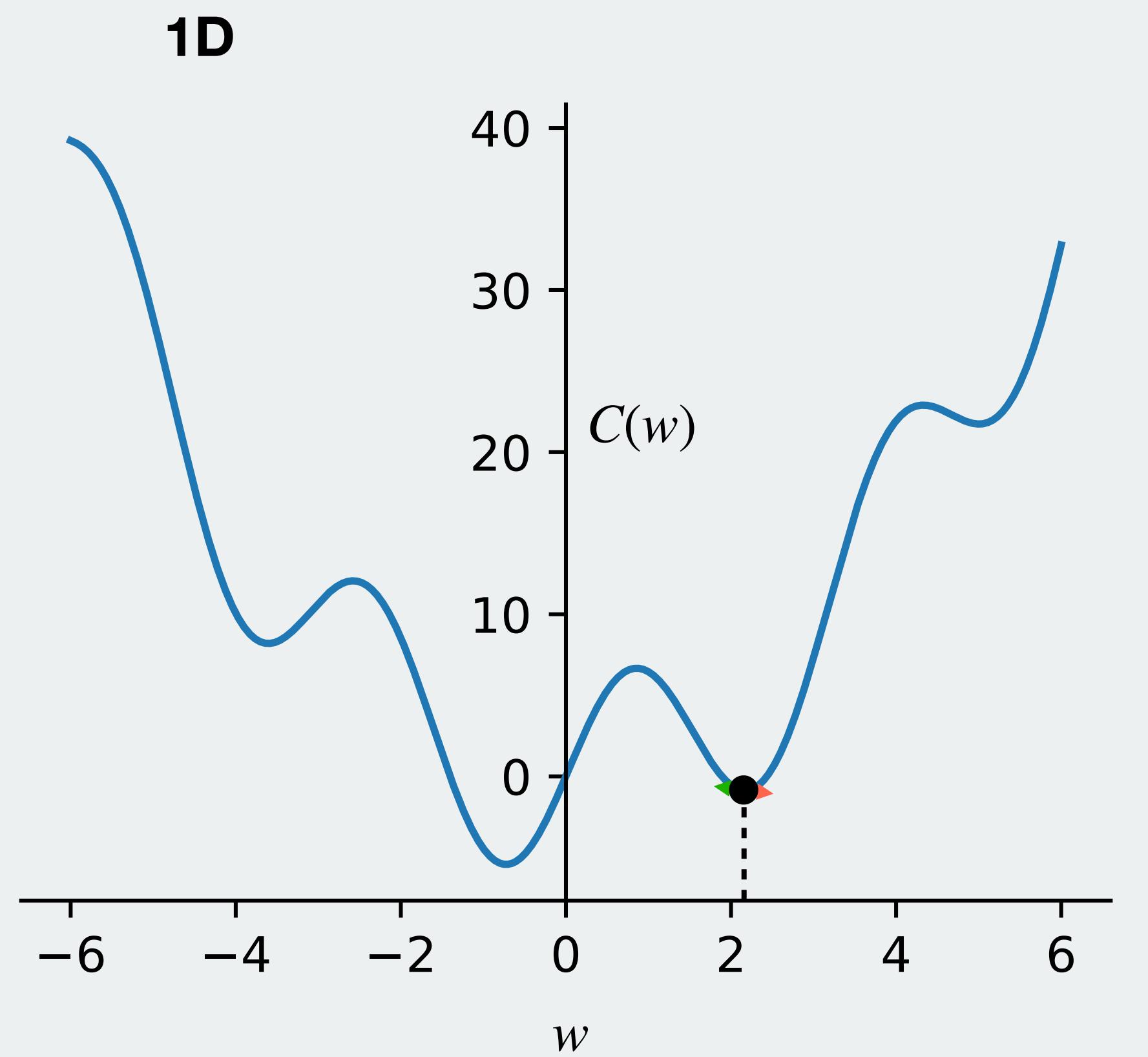
How do we find the weights that give the best classification?

> Gradient Descent



How do we find the weights that give the best classification?

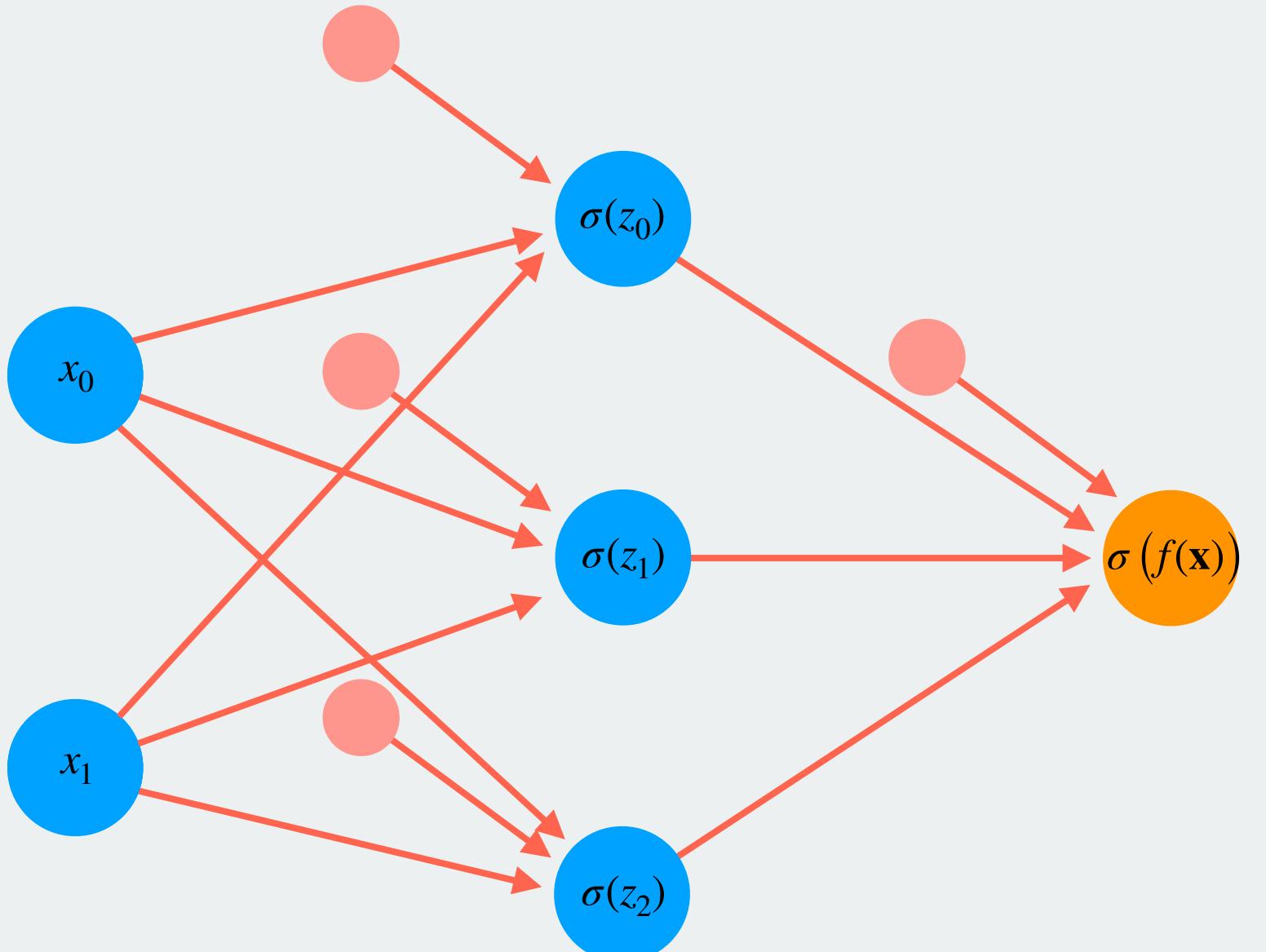
> Gradient Descent



How do we find the weights that give the best classification?

> Gradient Descent

13D? Or higher?

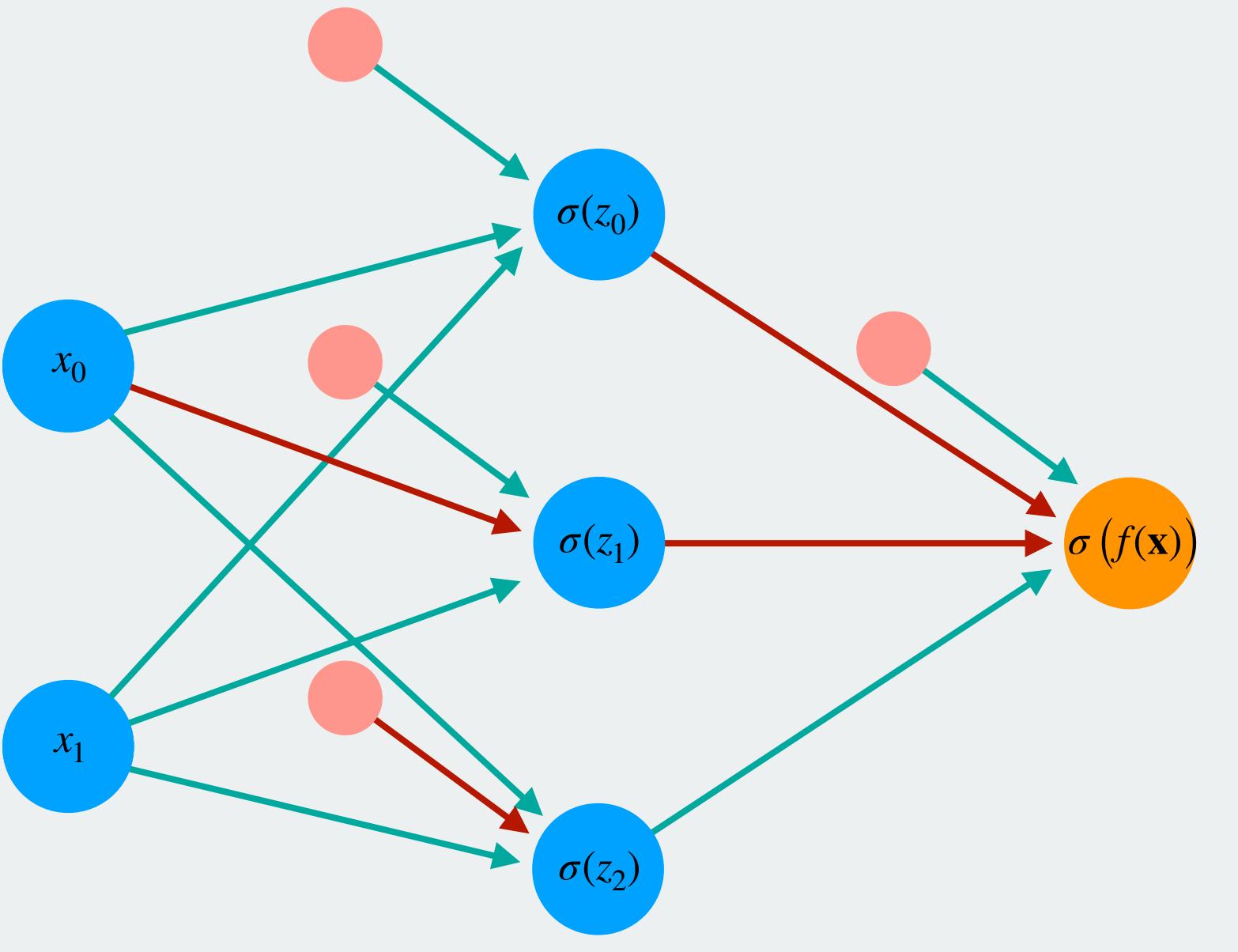


$$\mathbf{W} = \begin{bmatrix} w_{0,0} \\ w_{1,0} \\ w_{2,0} \\ w_{3,0} \\ w_{4,0} \\ w_{5,0} \\ w_{6,0} \\ w_{7,0} \\ w_{8,0} \\ w_{0,1} \\ w_{1,1} \\ w_{2,1} \\ w_{3,1} \end{bmatrix}$$

How do we find the weights that give the best classification?

> Gradient Descent

13D? Or higher?

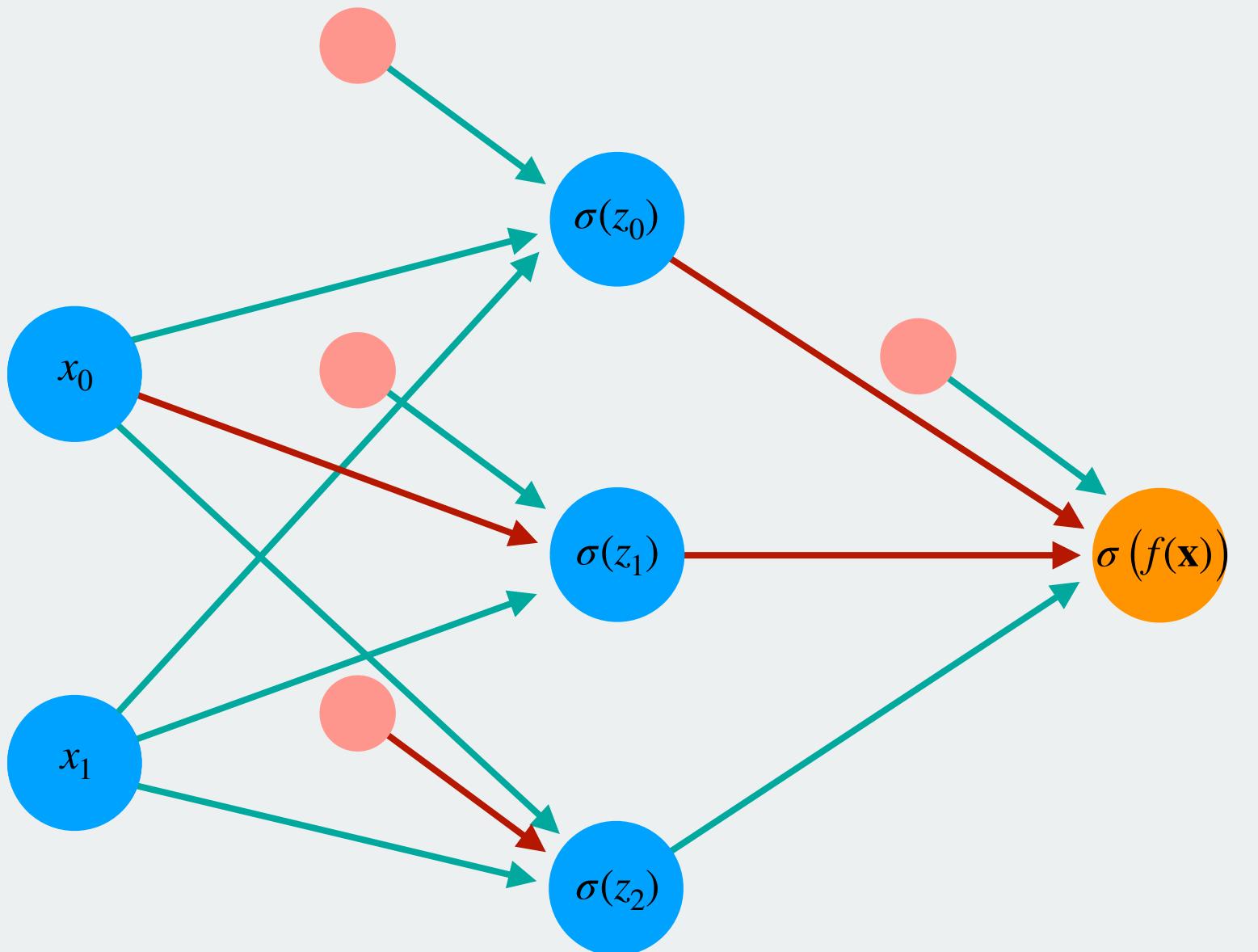


$$\mathbf{W} = \begin{bmatrix} w_{0,0} \\ w_{1,0} \\ w_{2,0} \\ w_{3,0} \\ w_{4,0} \\ w_{5,0} \\ w_{6,0} \\ w_{7,0} \\ w_{8,0} \\ w_{0,1} \\ w_{1,1} \\ w_{2,1} \\ w_{3,1} \end{bmatrix} \quad -\nabla C(\mathbf{W}) = \begin{bmatrix} -0.23 \\ 1.32 \\ 0.20 \\ 0.38 \\ -1.23 \\ 0.01 \\ 1.20 \\ -2.12 \\ 0.73 \\ 2.17 \\ -0.23 \\ -0.93 \\ 0.45 \end{bmatrix}$$

How do we find the weights that give the best classification?

> Gradient Descent

13D? Or higher?

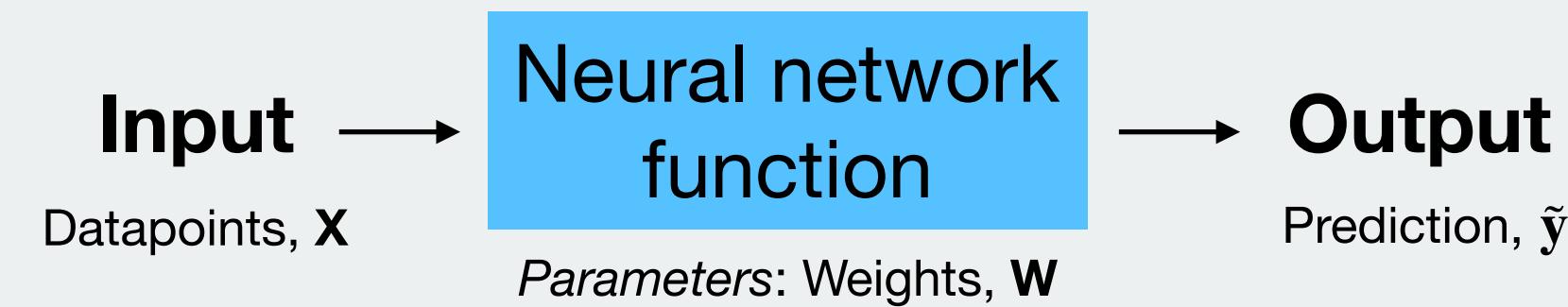


$$\mathbf{W} = \begin{bmatrix} w_{0,0} \\ w_{1,0} \\ w_{2,0} \\ w_{3,0} \\ w_{4,0} \\ w_{5,0} \\ w_{6,0} \\ w_{7,0} \\ w_{8,0} \\ w_{0,1} \\ w_{1,1} \\ w_{2,1} \\ w_{3,1} \end{bmatrix} \quad -\nabla C(\mathbf{W}) = \begin{bmatrix} -0.23 \\ 1.32 \\ 0.20 \\ 0.38 \\ -1.23 \\ 0.01 \\ 1.20 \\ -2.12 \\ 0.73 \\ 2.17 \\ -0.23 \\ -0.93 \\ 0.45 \end{bmatrix}$$

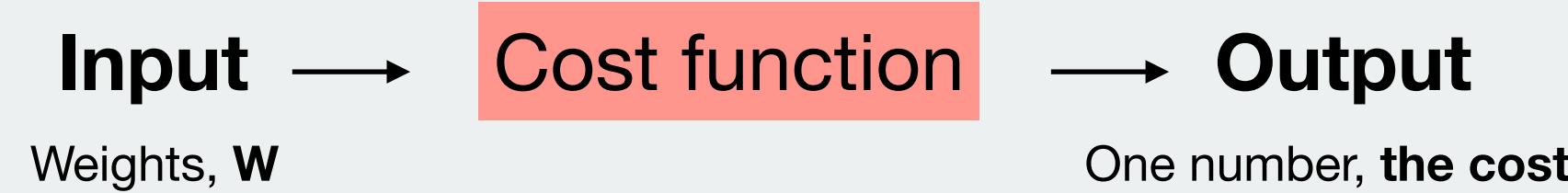
$$\mathbf{W} = \mathbf{W}^{\text{old}} + r(-\nabla C(\mathbf{W}^{\text{old}}))$$

How it all hangs together

(1) The model



(2) Its performance



(3) The cost function gradient in \mathbf{W}

$$-\nabla C(\mathbf{W}) = \begin{bmatrix} -0.23 \\ 1.32 \\ 0.20 \\ 0.38 \\ -1.23 \\ 0.01 \\ 1.20 \\ -2.12 \\ 0.73 \\ 2.17 \\ 0.54 \\ -0.23 \\ -0.93 \\ 0.45 \end{bmatrix}$$

(4) Updating \mathbf{W}

$$\mathbf{W} = \mathbf{W}^{\text{old}} + r (-\nabla C(\mathbf{W}^{\text{old}}))$$

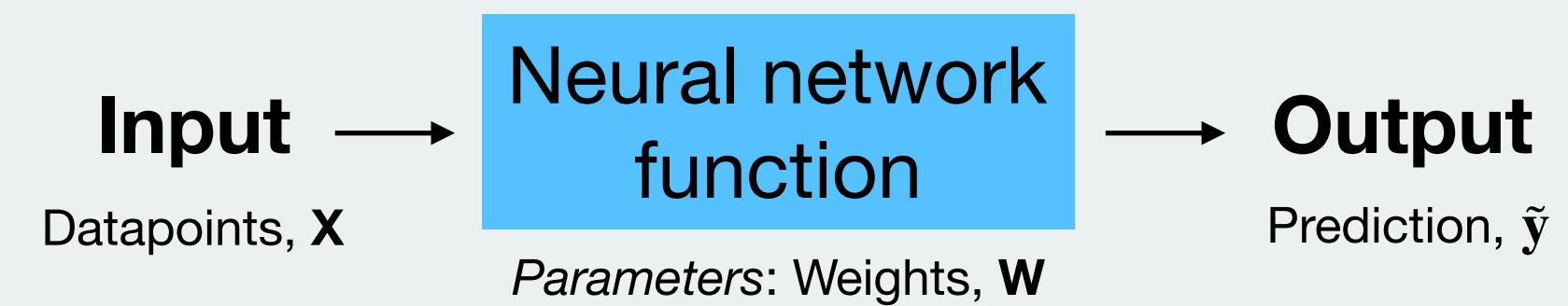
(5) Repeat 3 and 4

r is usually called the *learning rate*

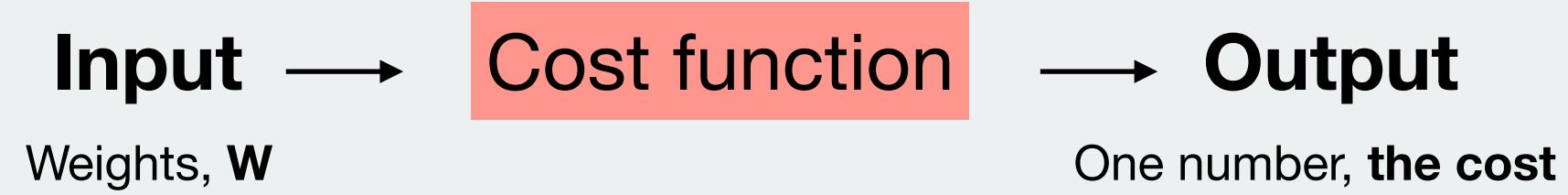
How it all hangs together

Find the gradients with
Backpropagation

(1) The model



(2) Its performance



(3) The cost function gradient in \mathbf{W}

$$-\nabla C(\mathbf{W}) = \begin{bmatrix} -0.23 \\ 1.32 \\ 0.20 \\ 0.38 \\ -1.23 \\ 0.01 \\ 1.20 \\ -2.12 \\ 0.73 \\ 2.17 \\ 0.54 \\ -0.23 \\ -0.93 \\ 0.45 \end{bmatrix}$$

(4) Updating \mathbf{W}

$$\mathbf{W} = \mathbf{W}^{\text{old}} + r (-\nabla C(\mathbf{W}^{\text{old}}))$$

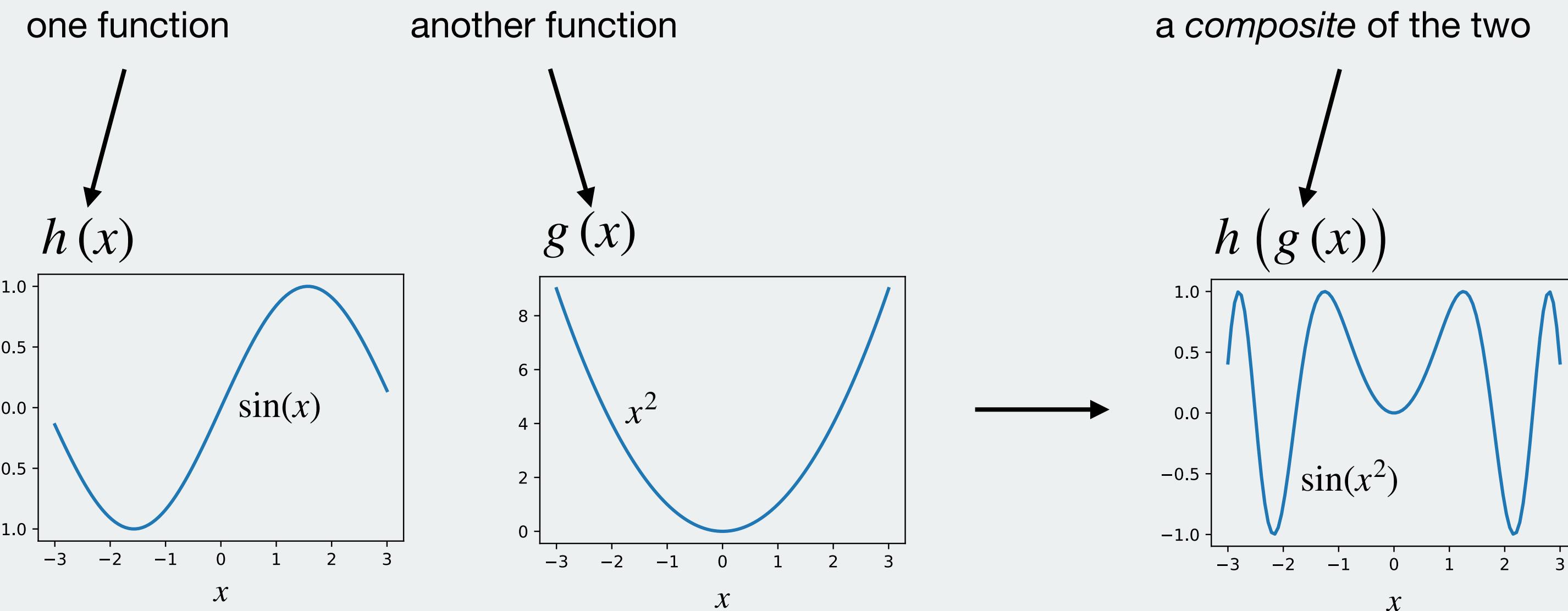
(5) Repeat 3 and 4

r is usually called the *learning rate*

Backpropagation

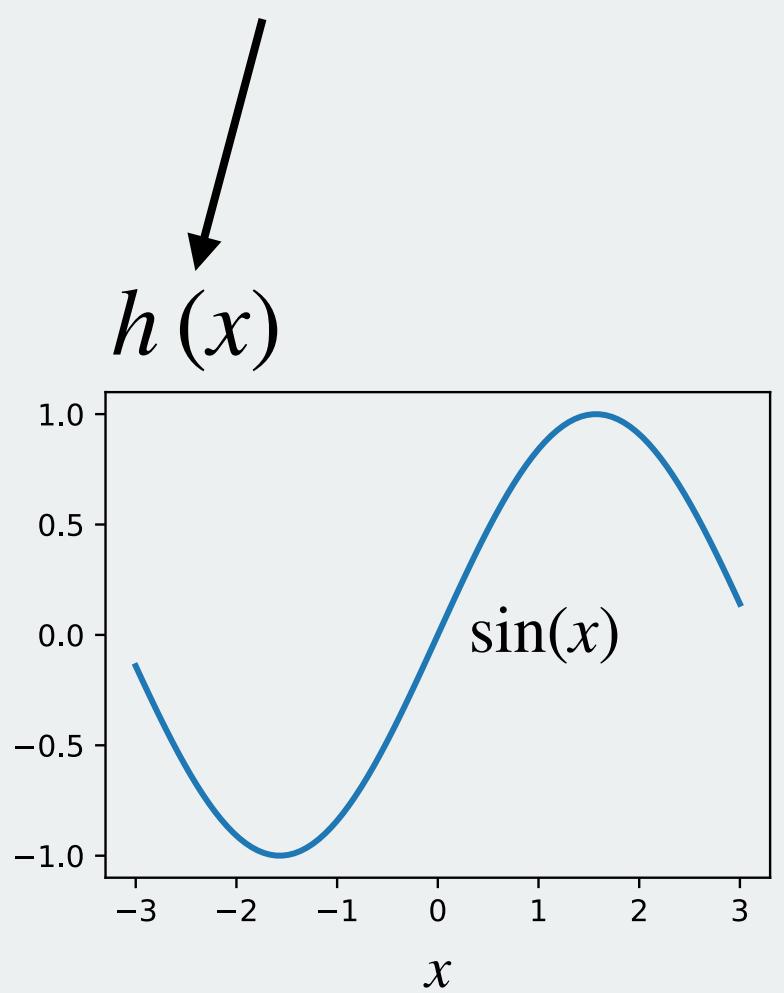
THE algorithm for computing the analytical **gradient** of the cost function

Chain rule – Taking derivatives of composite functions

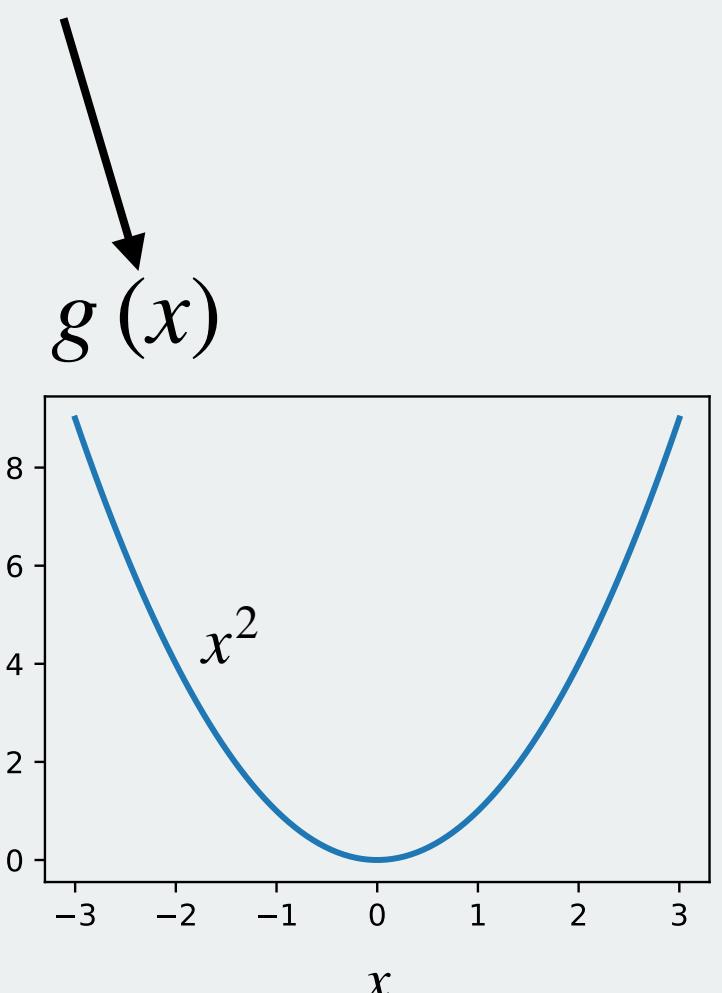


Chain rule – Taking derivatives of composite functions

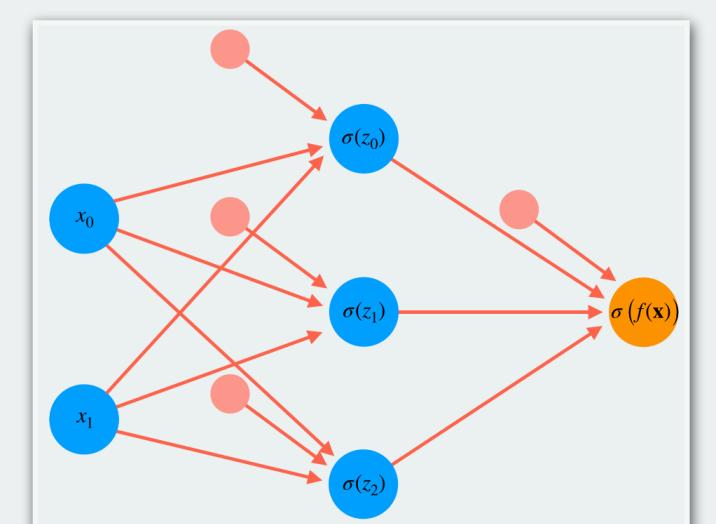
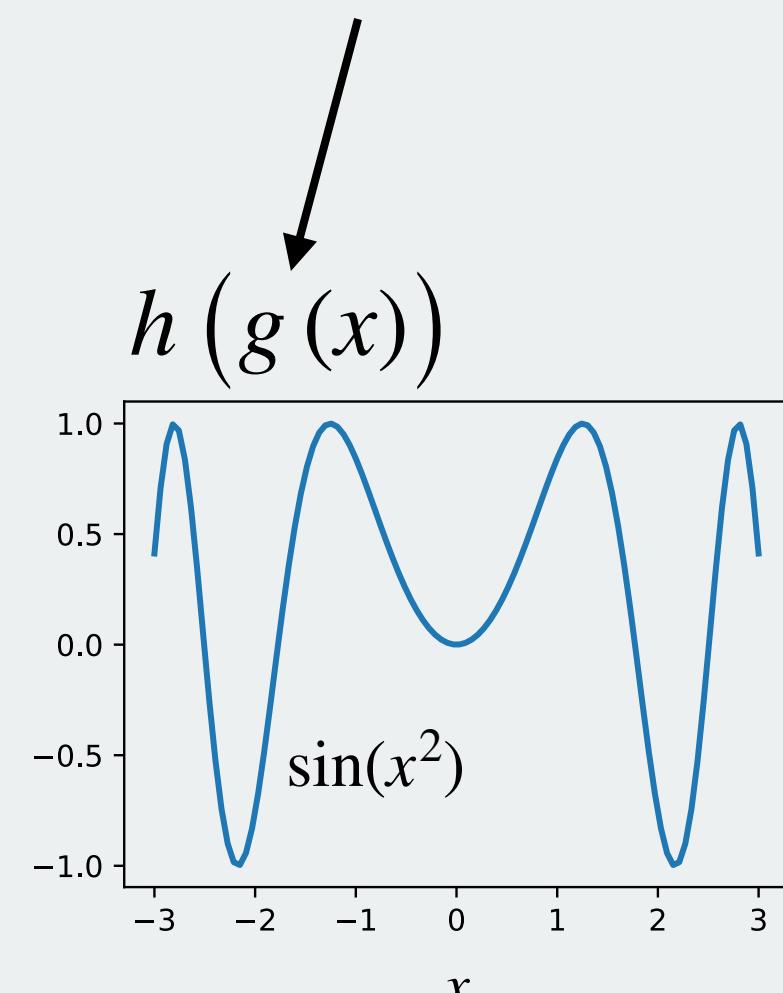
one function



another function



a composite of the two

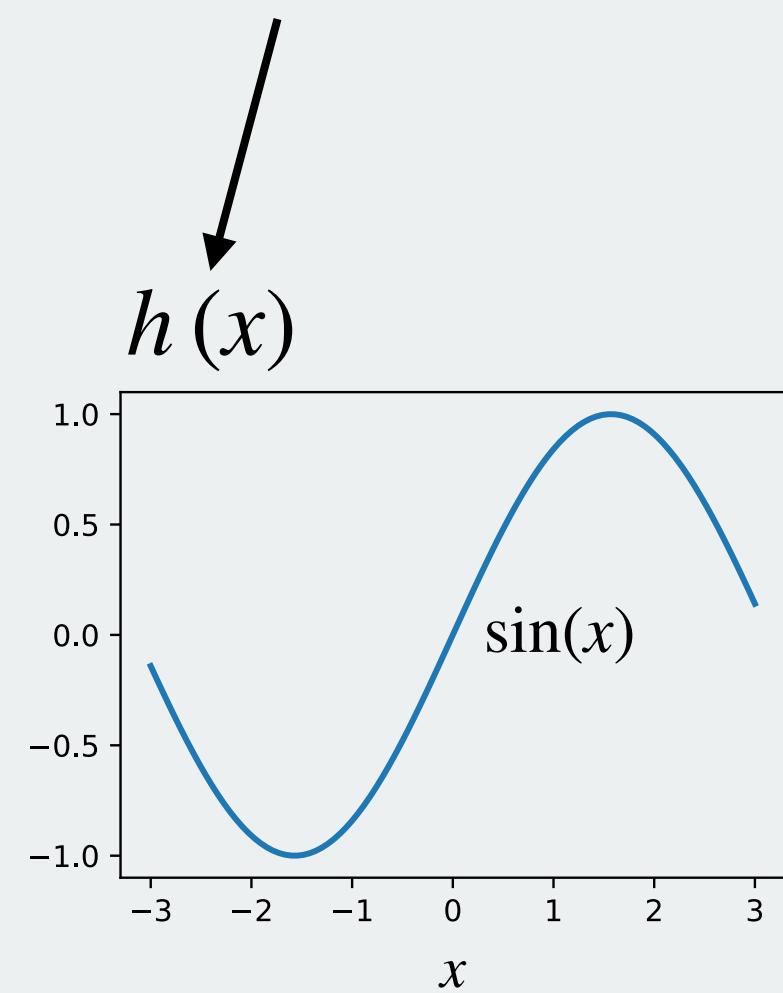


$$\sigma(w_{0,1} + w_{0,0}x_0 + w_{1,0}x_1 + w_{2,0})$$

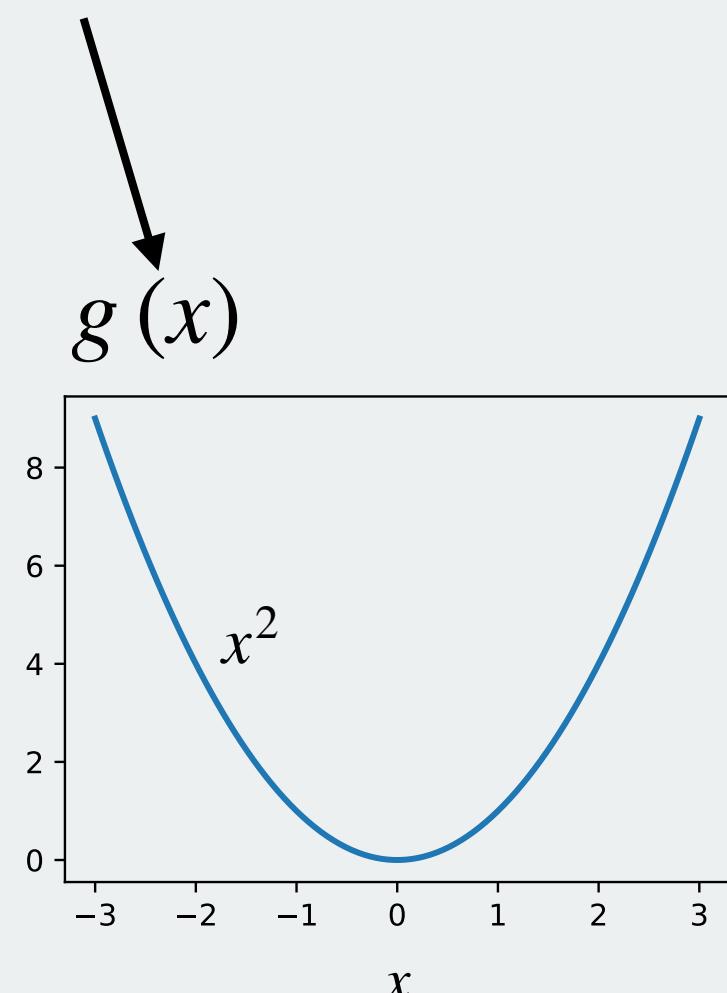
$$\begin{aligned} & \sigma(w_{0,1} + \sigma(w_{0,0} + x_0w_{1,0} + x_1w_{2,0})w_{1,1} + \\ & \sigma(w_{3,0} + x_0w_{4,0} + x_1w_{5,0})w_{2,1} + \\ & \sigma(w_{6,0} + x_0w_{7,0} + x_1w_{8,0})w_{3,1}) = \sigma(f(x)) \end{aligned}$$

Chain rule – Taking derivatives of composite functions

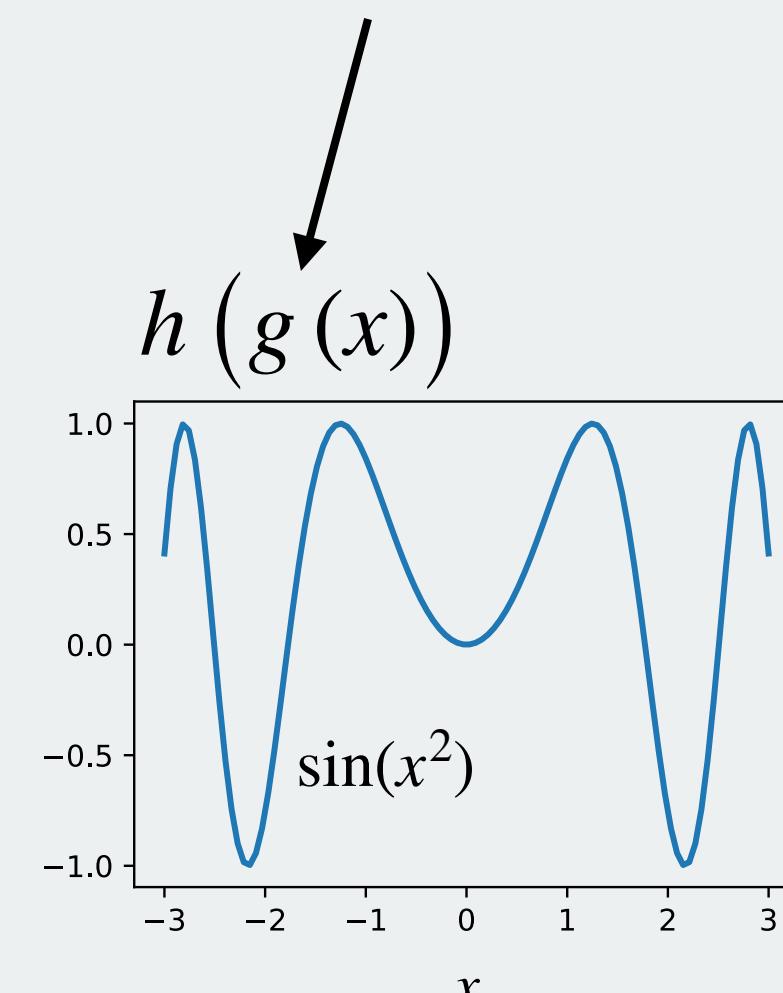
one function



another function



a composite of the two

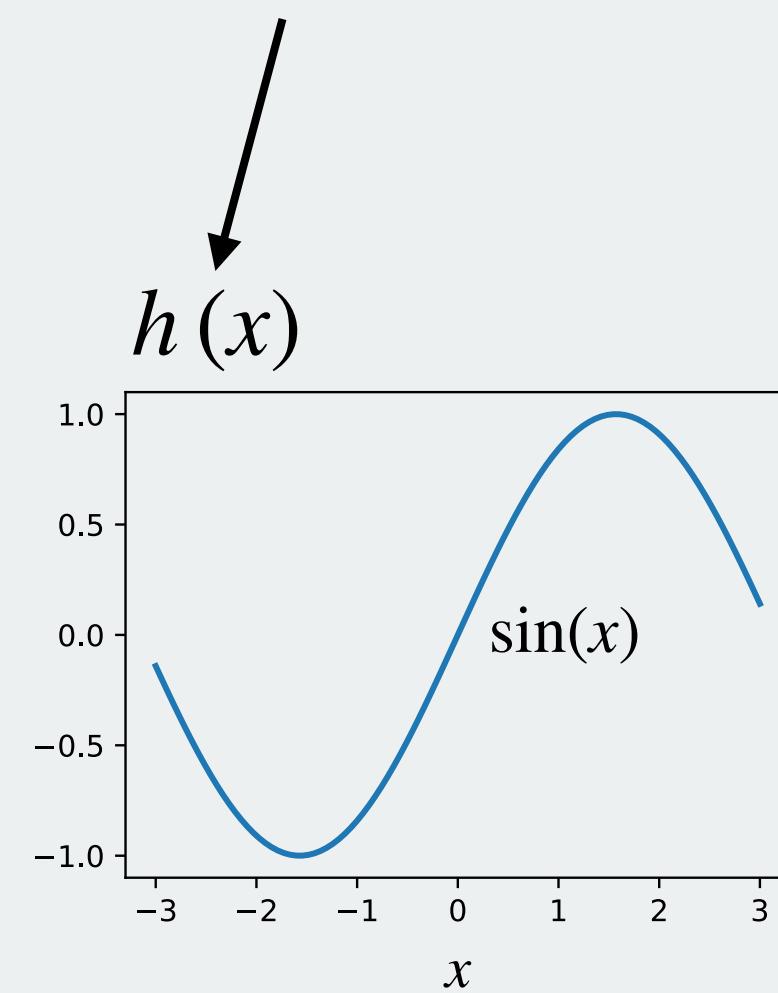


$$h(g(x))$$

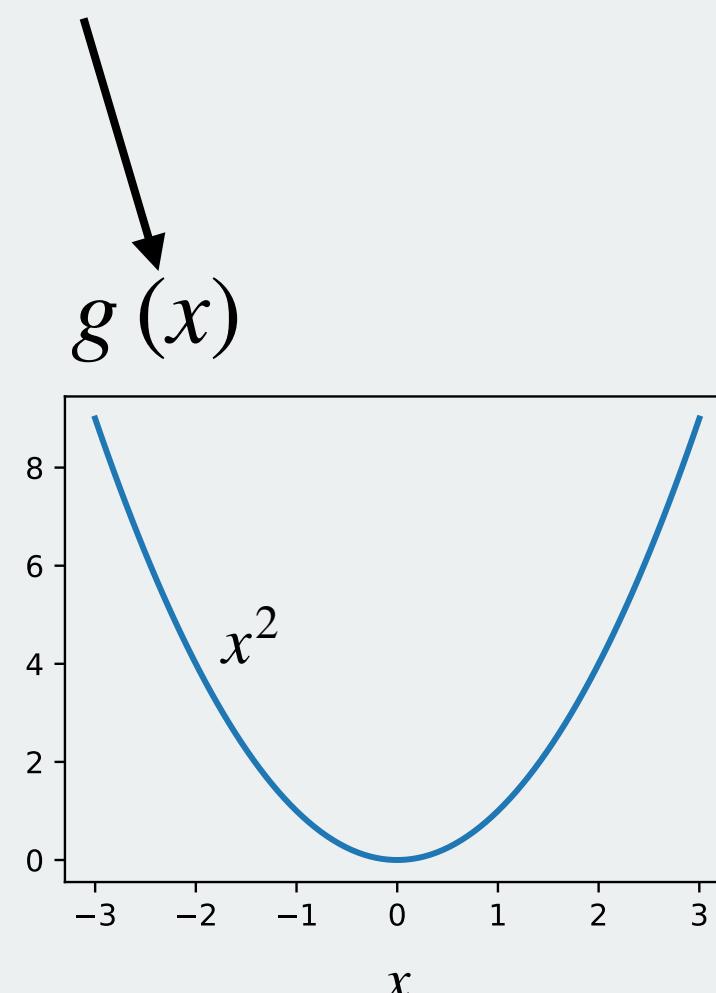


Chain rule – Taking derivatives of composite functions

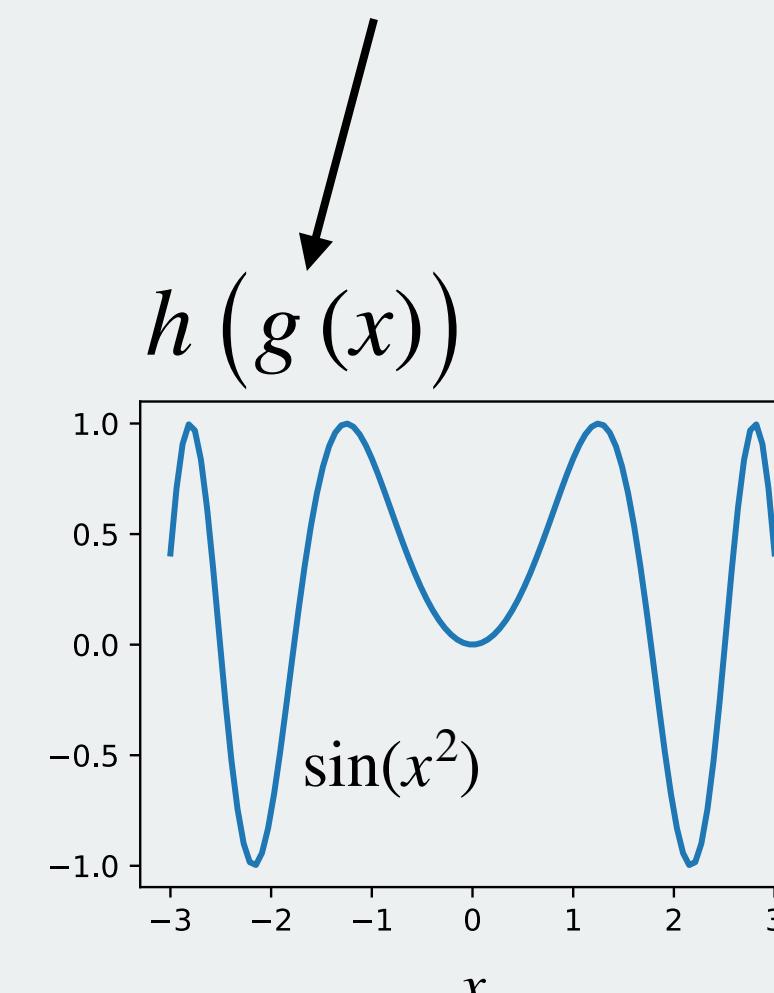
one function



another function



a composite of the two



$$h(g(x))$$

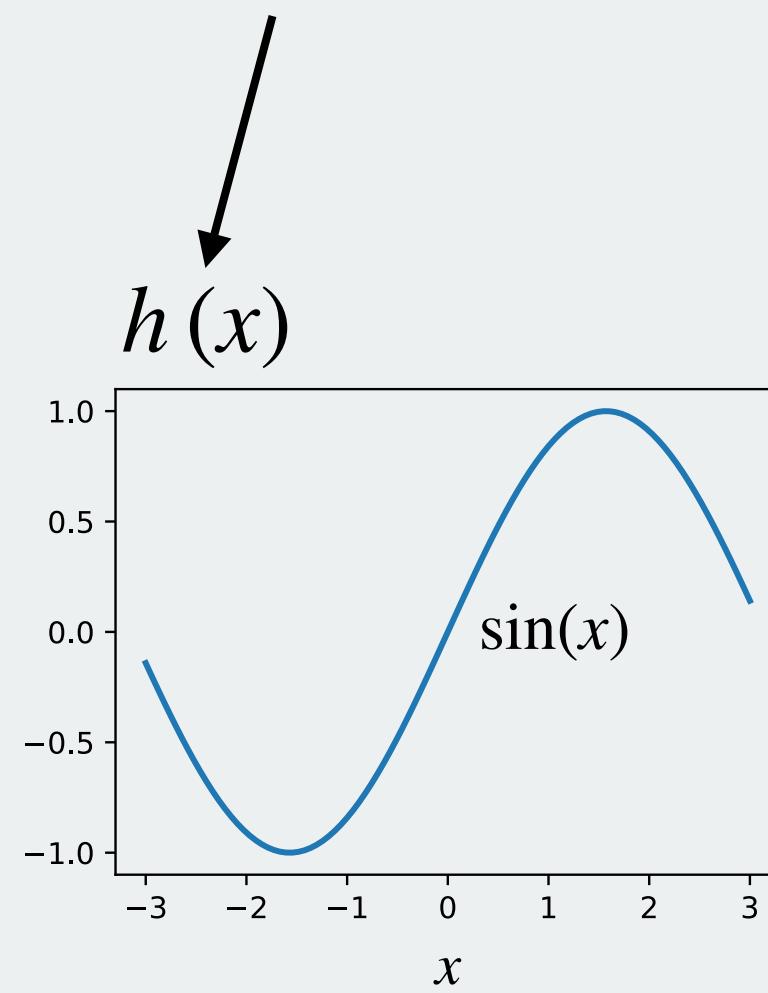


Chain rule says:

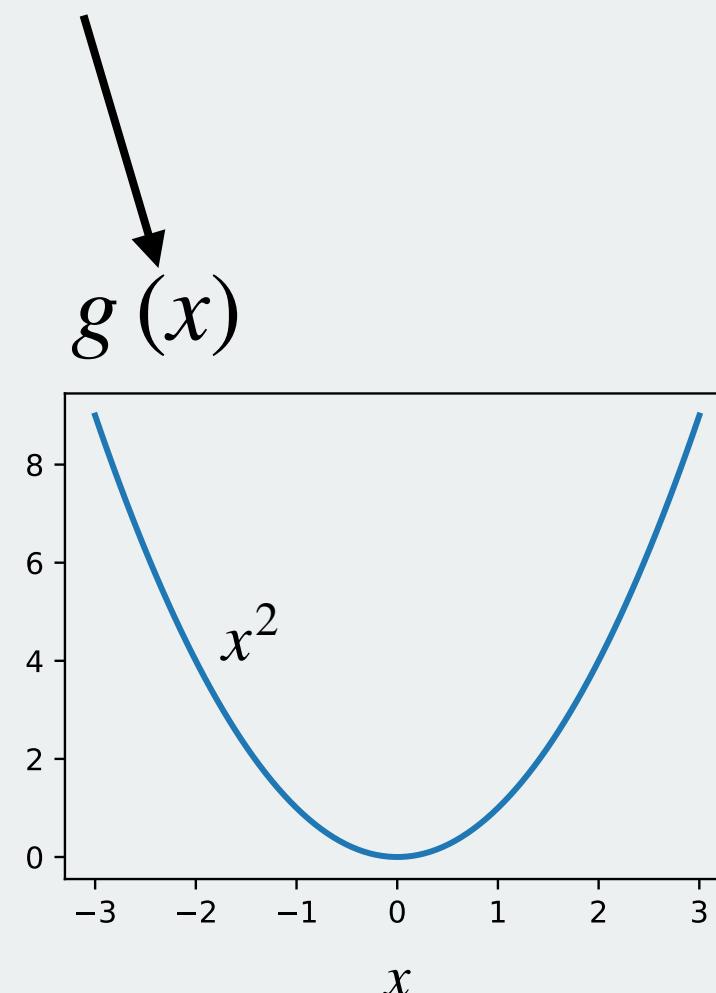
$$\frac{dh}{dx} = \frac{dh}{dg} \frac{dg}{dx}$$

Chain rule – Taking derivatives of composite functions

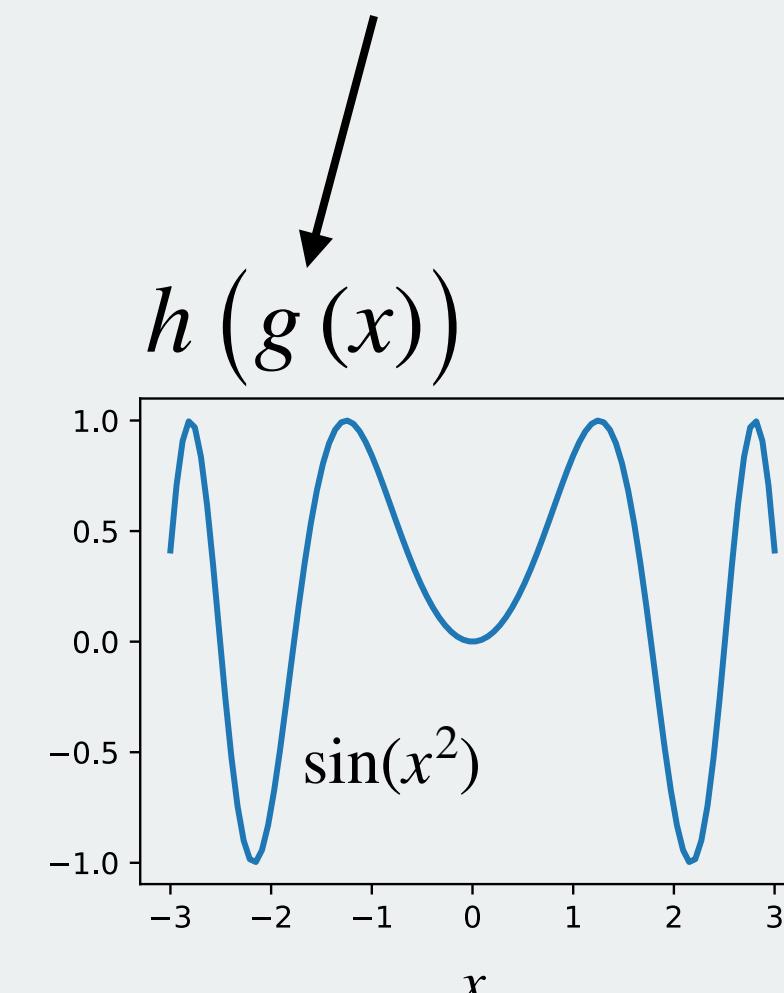
one function



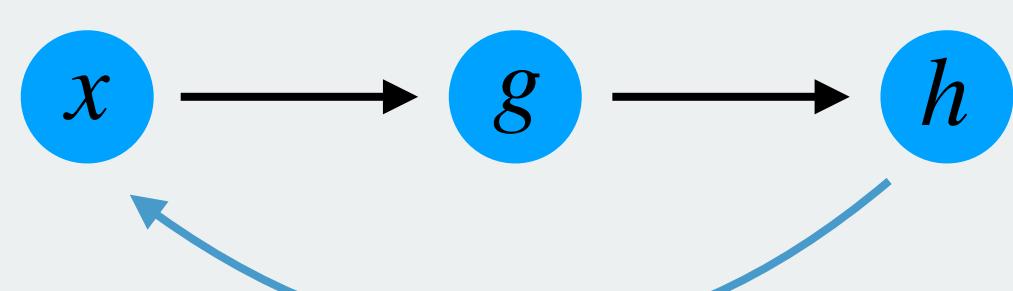
another function



a composite of the two



$$h(g(x))$$

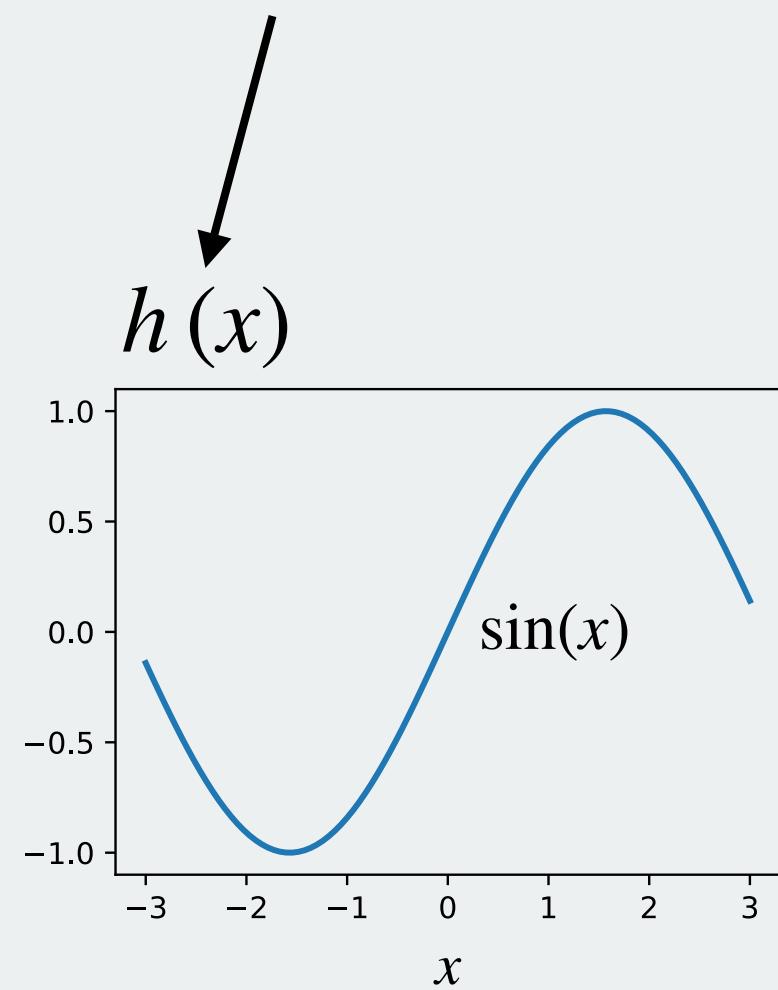


Chain rule says:

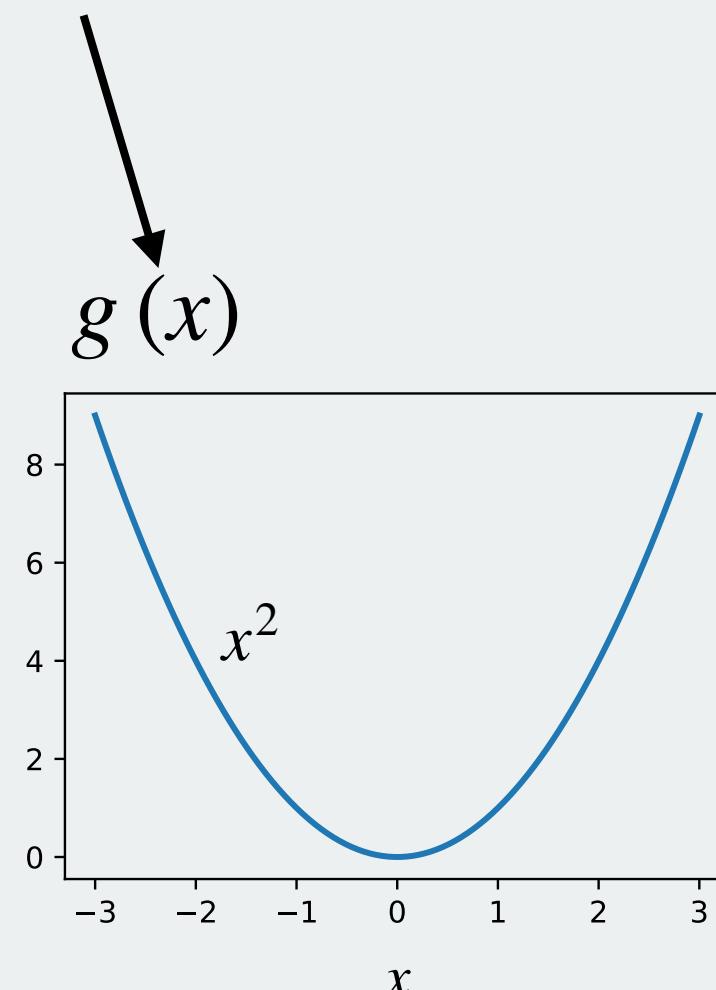
$$\frac{dh}{dx} = \frac{dh}{dg} \frac{dg}{dx}$$

Chain rule – Taking derivatives of composite functions

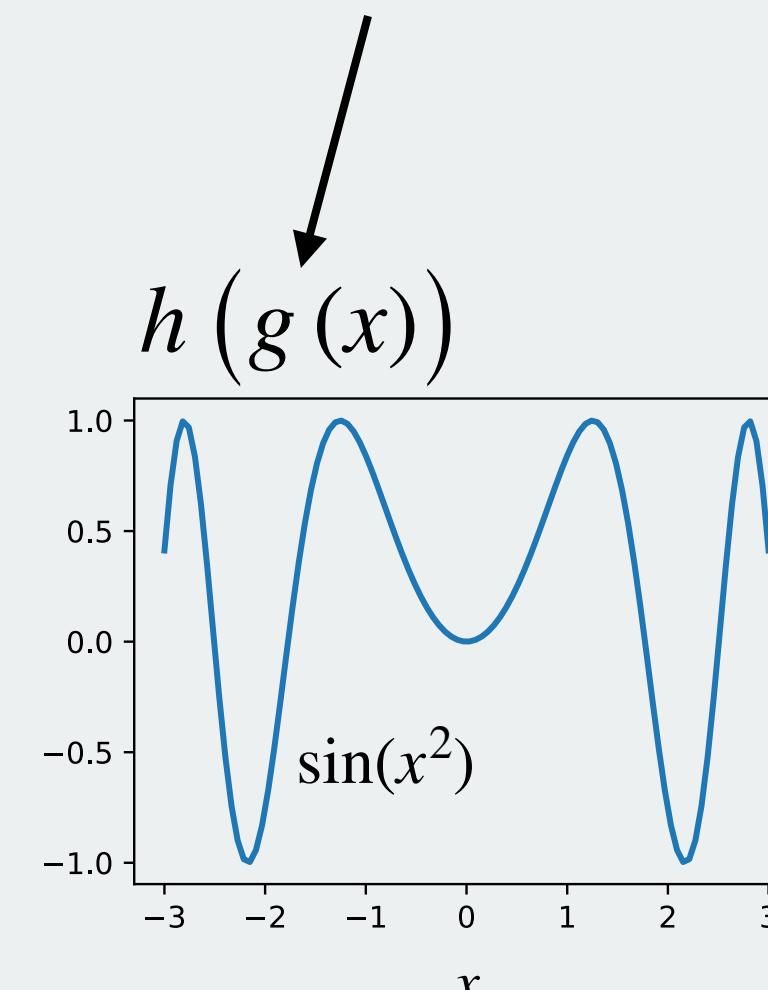
one function



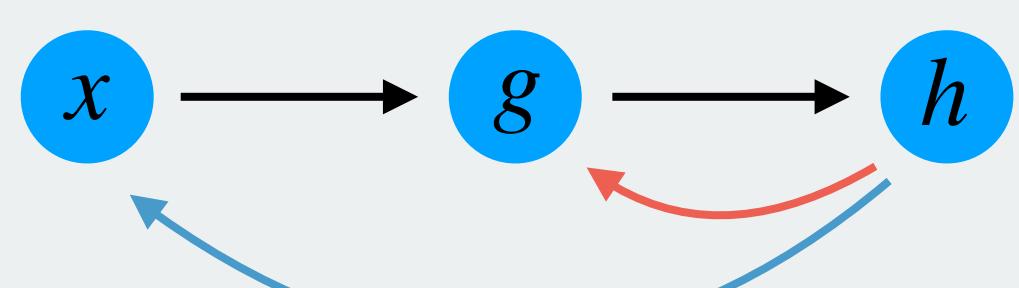
another function



a composite of the two



$h(g(x))$

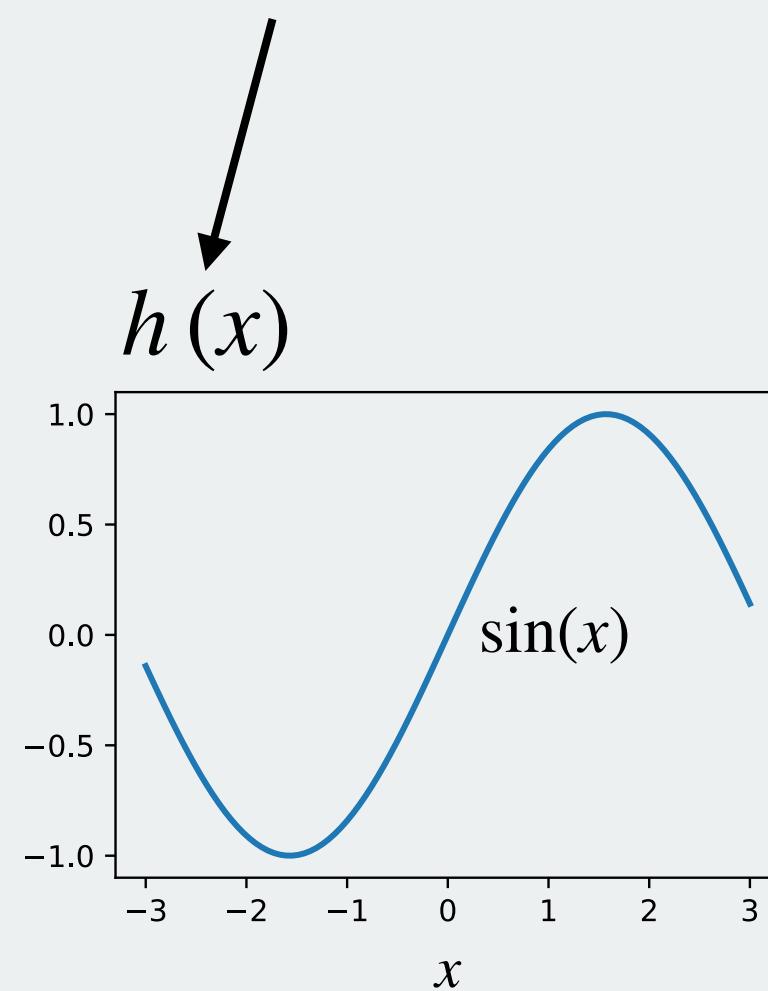


Chain rule says:

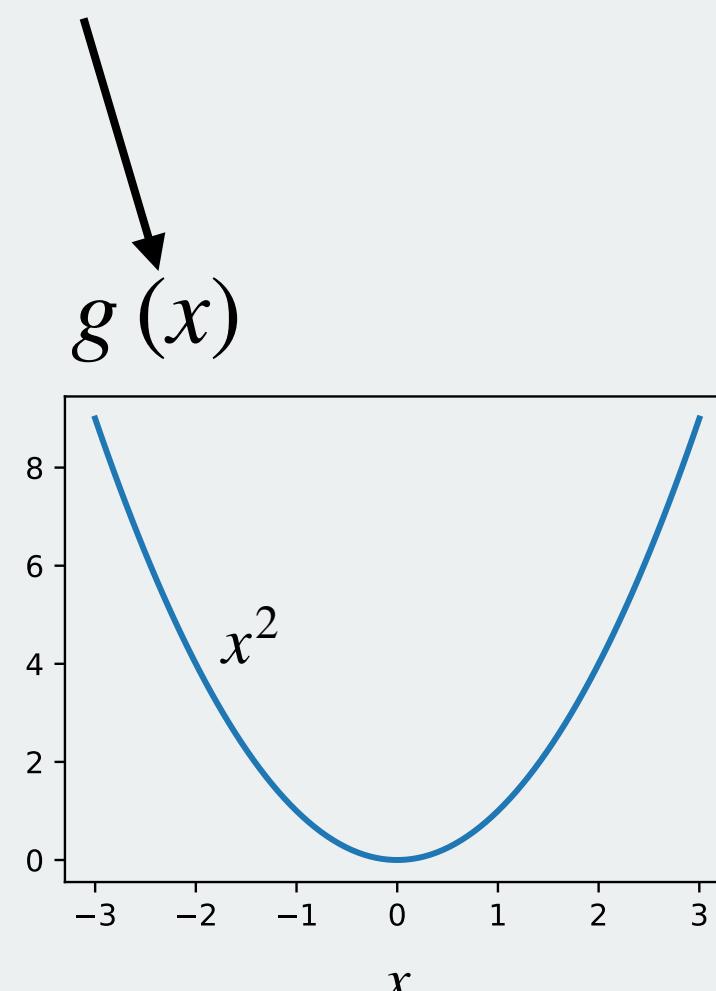
$$\frac{dh}{dx} = \frac{dh}{dg} \frac{dg}{dx}$$

Chain rule – Taking derivatives of composite functions

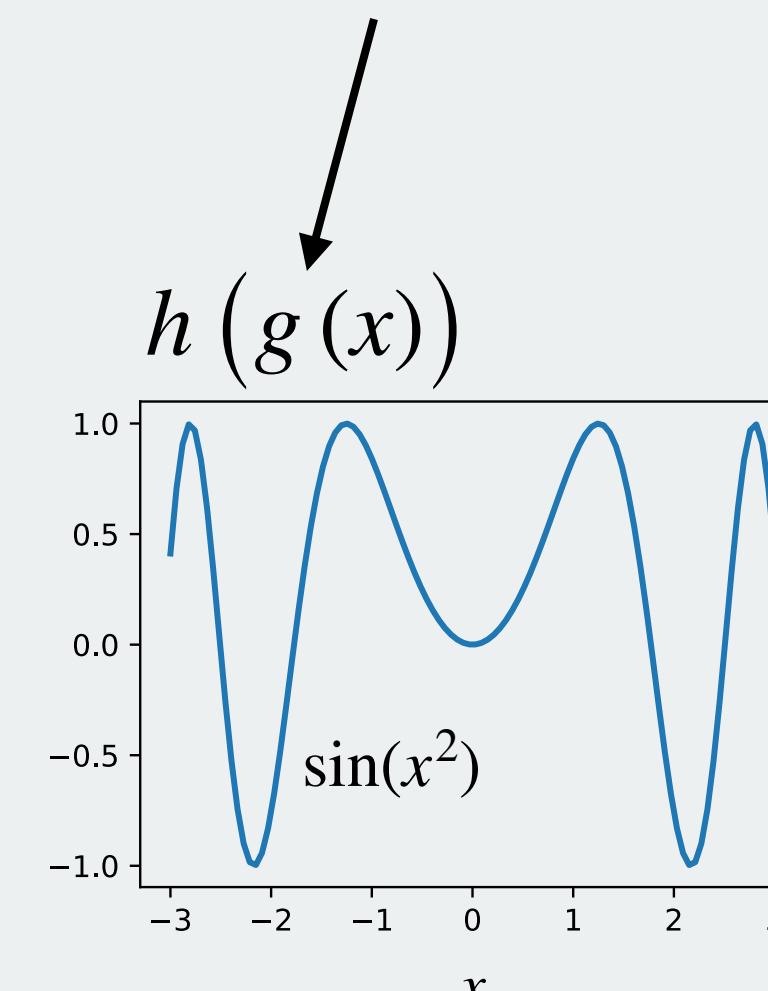
one function



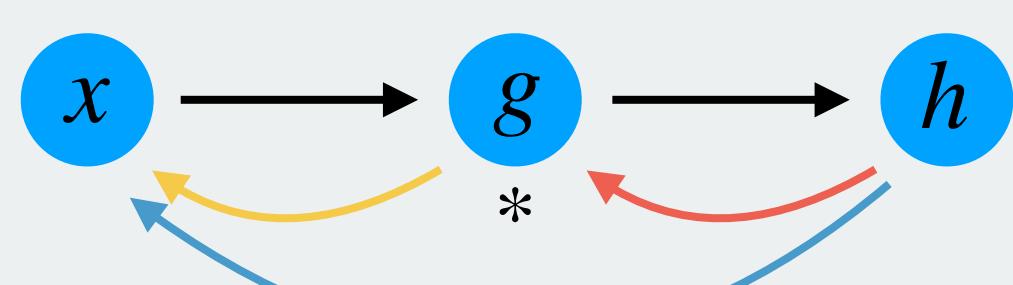
another function



a composite of the two



$$h(g(x))$$

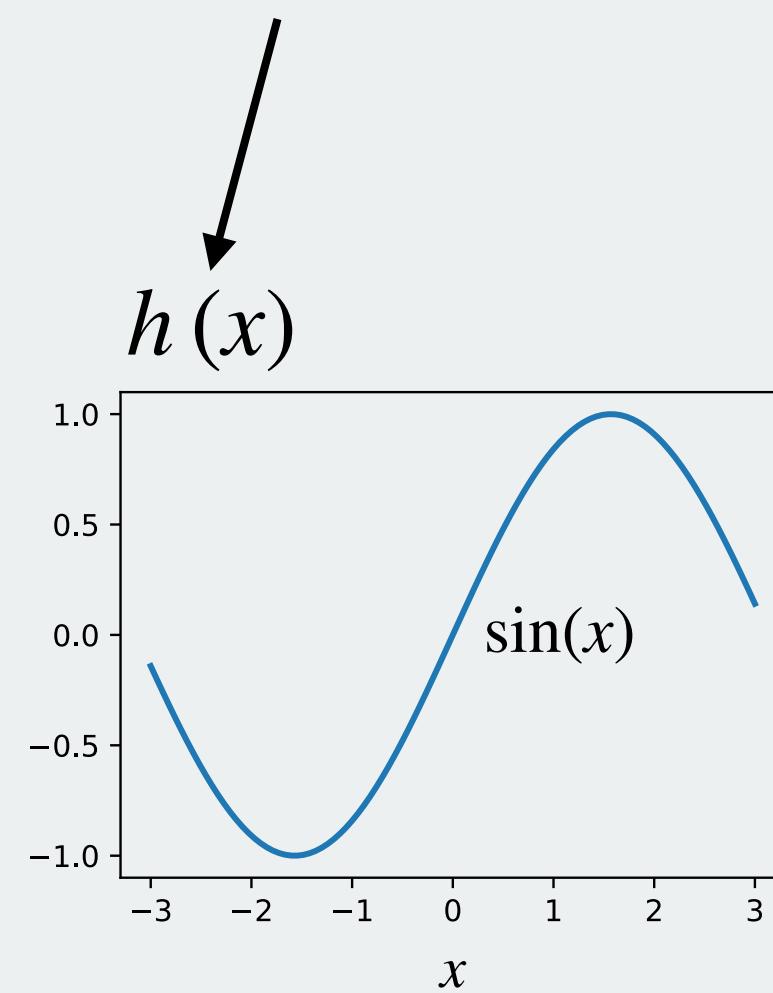


Chain rule says:

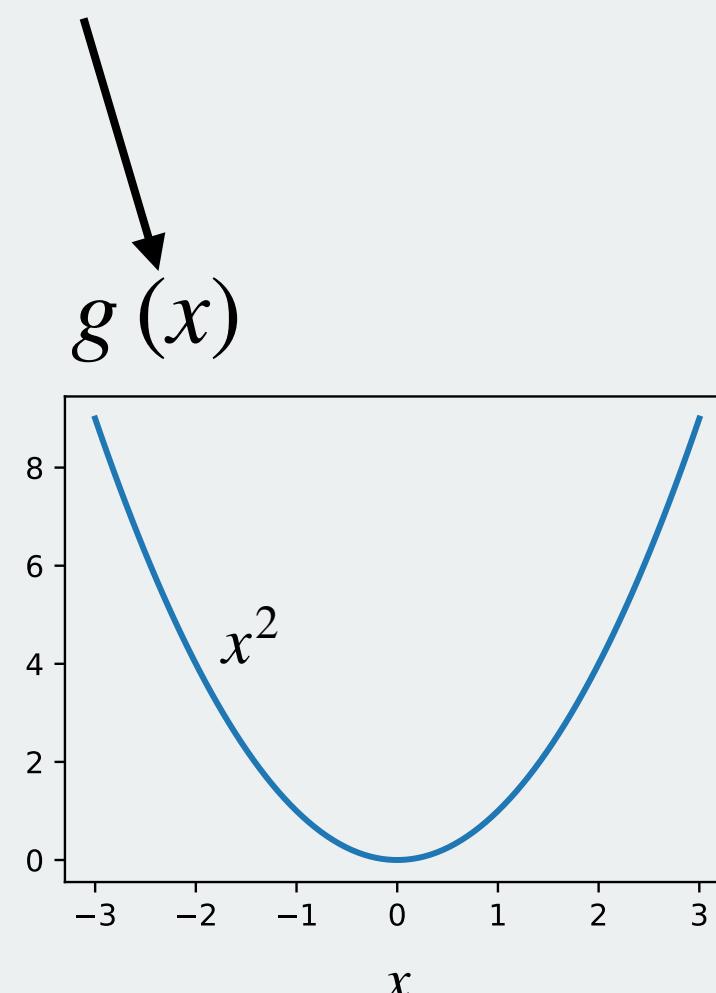
$$\frac{dh}{dx} = \frac{dh}{dg} \frac{dg}{dx}$$

Chain rule – Taking derivatives of composite functions

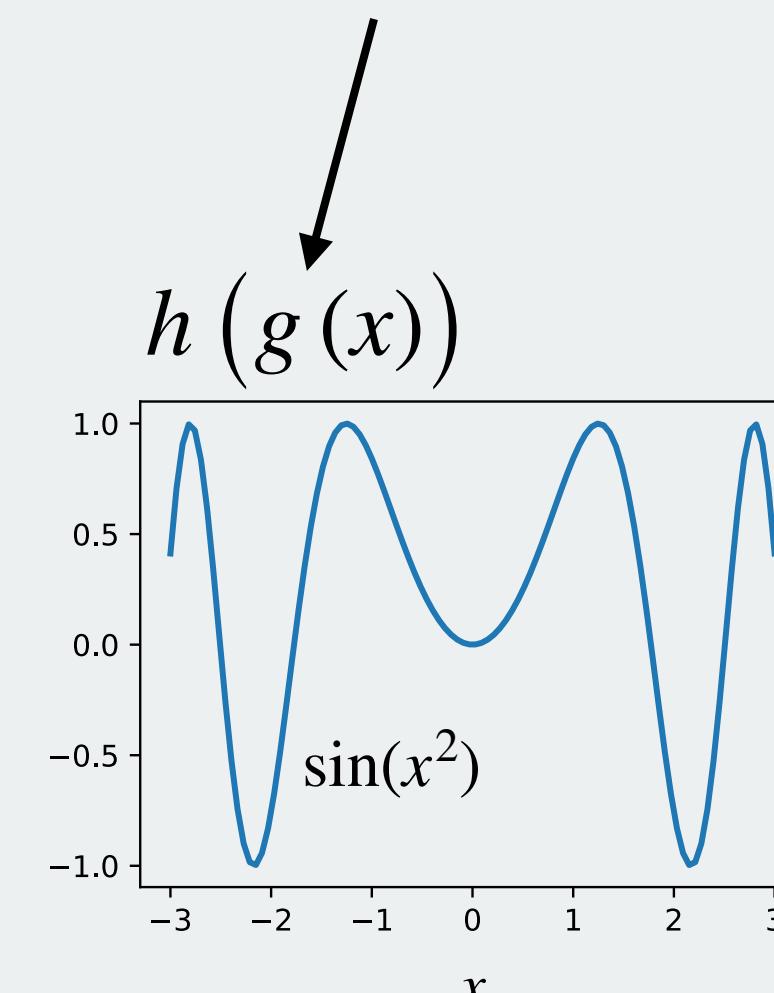
one function



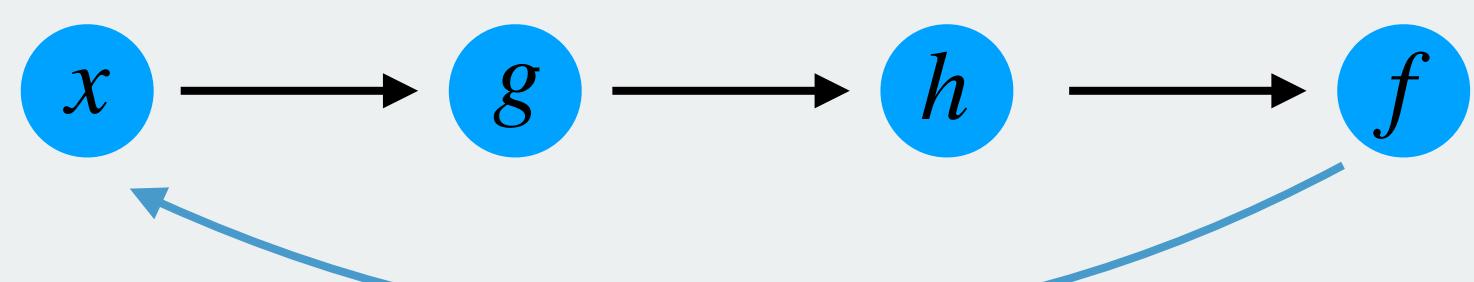
another function



a composite of the two



$$f(h(g(x)))$$

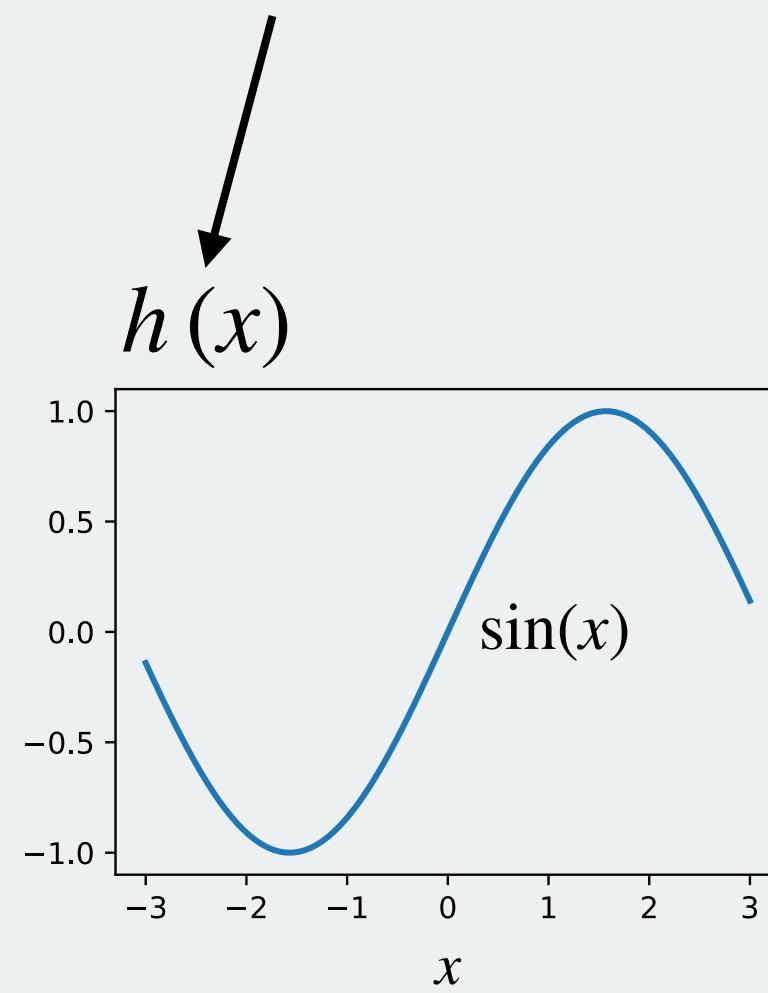


Chain rule says:

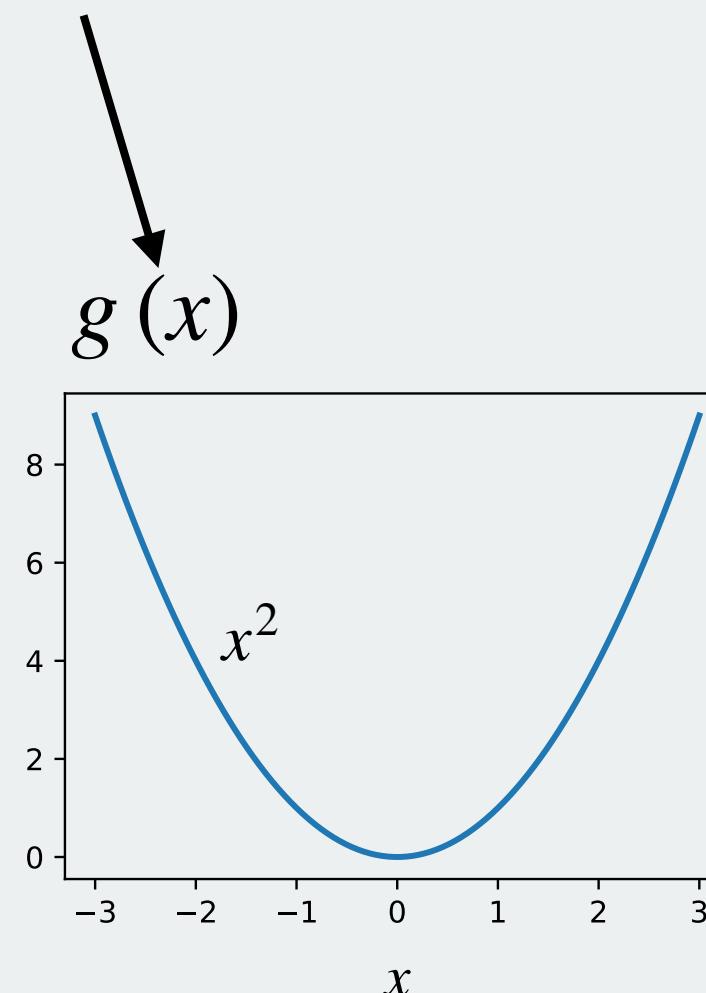
$$\frac{df}{dx} = ?$$

Chain rule – Taking derivatives of composite functions

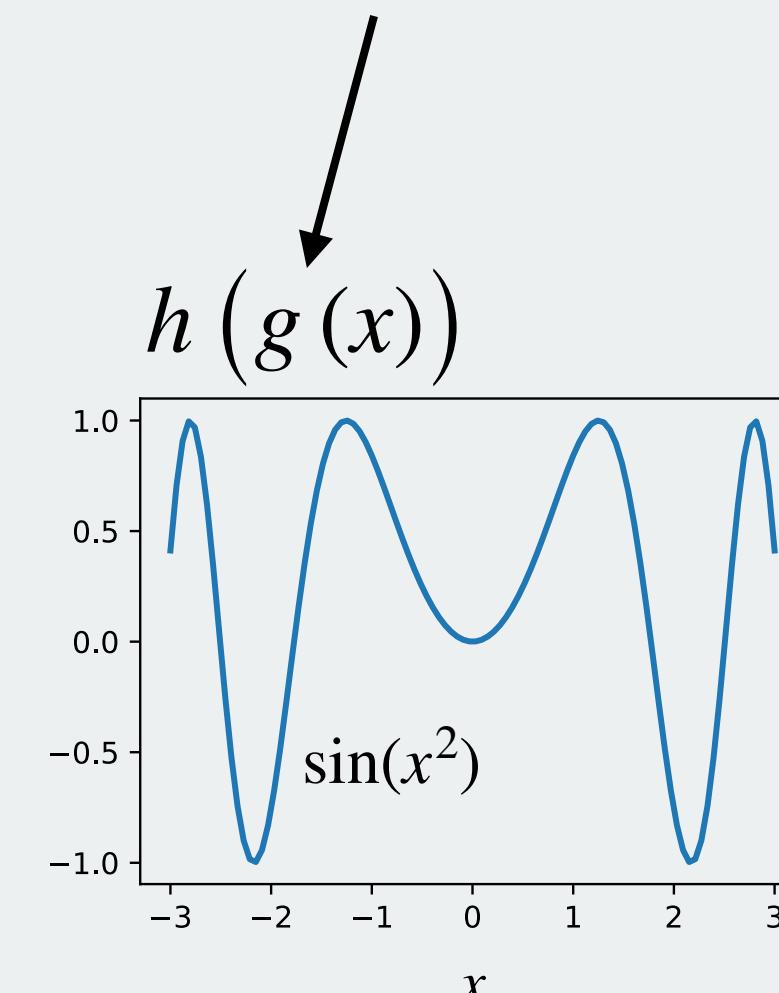
one function



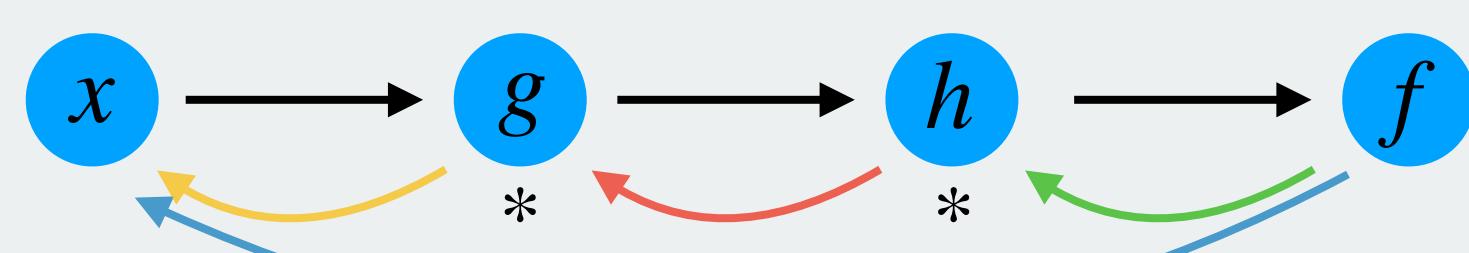
another function



a composite of the two



$$f(h(g(x)))$$



Chain rule says:

$$\frac{df}{dx} = \frac{df}{dh} \frac{dh}{dg} \frac{dg}{dx}$$

[3Blue1Brown Chain Rule video](#)

Backpropagation – simple example

Model:

$$m(z) = -z$$

$$g(z) = \exp(z)$$

$$h(z) = z + 1$$

$$f(z) = \frac{1}{z}$$

Data:

$$x_0 = 1$$

$$x_1 = 1.1$$



Backpropagation – simple example

Model:

$$m(z) = -z$$

$$g(z) = \exp(z)$$

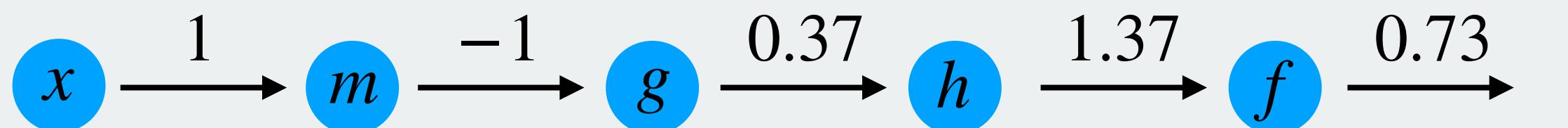
$$h(z) = z + 1$$

$$f(z) = \frac{1}{z}$$

Data:

$$x_0 = 1$$

$$x_1 = 1.1$$



Backpropagation – simple example

Model:

$$m(z) = -z$$

$$g(z) = \exp(z)$$

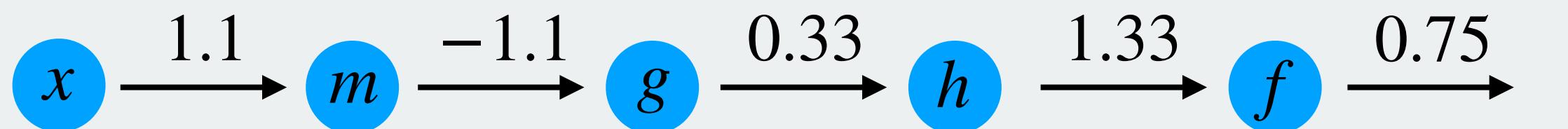
$$h(z) = z + 1$$

$$f(z) = \frac{1}{z}$$

Data:

$$x_0 = 1$$

$$x_1 = 1.1$$



Backpropagation – simple example

Model:

$$m(z) = -z$$

$$g(z) = \exp(z)$$

$$h(z) = z + 1$$

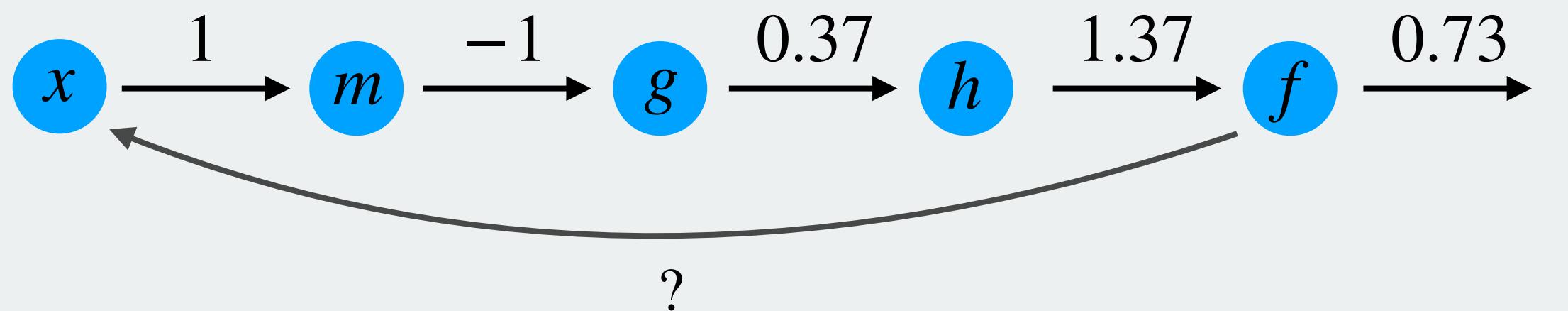
$$f(z) = \frac{1}{z}$$

Data:

$$x_0 = 1$$

$$x_1 = 1.1$$

Q: How does a small nudge in x influence $f(h(g(m(x))))$?



Backpropagation – simple example

Model:

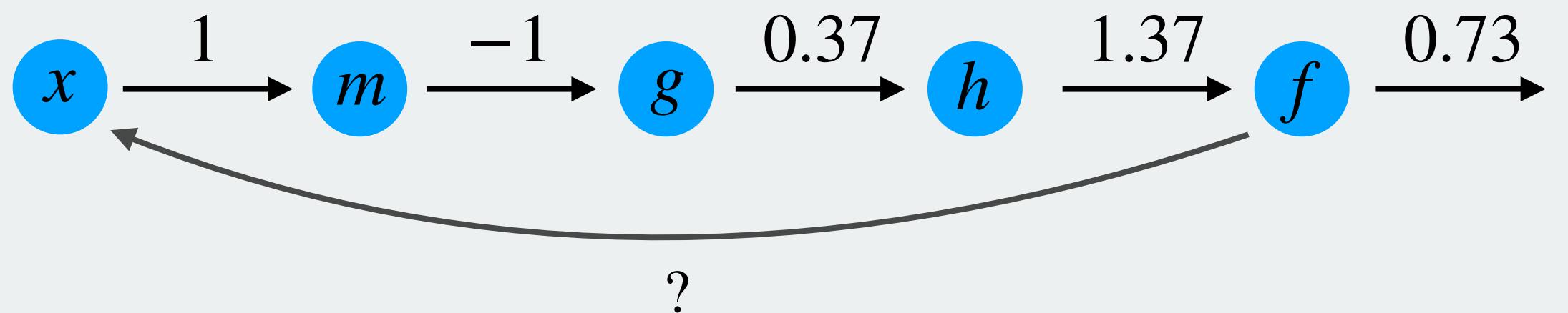
$$m(z) = -z$$
$$g(z) = \exp(z)$$
$$h(z) = z + 1$$
$$f(z) = \frac{1}{z}$$

Data:

$$x_0 = 1$$
$$x_1 = 1.1$$

Q: How does a small nudge in x influence $f(h(g(m(x))))$?

A: Propagate gradients backwards using the chain rule!



Backpropagation – simple example

Model:

$$\begin{aligned}m(z) &= -z \\g(z) &= \exp(z) \\h(z) &= z + 1 \\f(z) &= \frac{1}{z}\end{aligned}$$

Data:

$$\begin{aligned}x_0 &= 1 \\x_1 &= 1.1\end{aligned}$$

Q: How does a small nudge in x influence $f(h(g(m(x))))$?

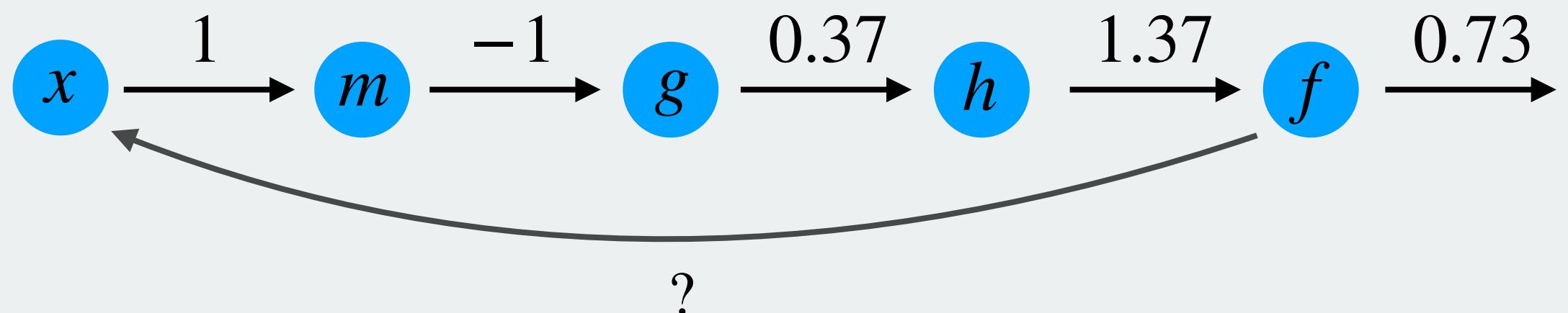
A: Propagate gradients backwards using the chain rule!

Chain rule from f to x :

$$\frac{df}{dx} = \frac{df}{dh} \frac{dh}{dg} \frac{dg}{dm} \frac{dm}{dx}$$

Model derivatives:

$$\begin{aligned}m'(z) &= -1 \\g'(z) &= \exp(z) \\h'(z) &= 1 \\f'(z) &= -\frac{1}{z^2}\end{aligned}$$



Backpropagation – simple example

Model:

$$\begin{aligned}m(z) &= -z \\g(z) &= \exp(z) \\h(z) &= z + 1 \\f(z) &= \frac{1}{z}\end{aligned}$$

Data:

$$\begin{aligned}x_0 &= 1 \\x_1 &= 1.1\end{aligned}$$

Q: How does a small nudge in x influence $f(h(g(m(x))))$?

A: Propagate gradients backwards using the chain rule!

Chain rule from f to h :

$$\frac{df}{dh}$$

Model derivatives:

$$\begin{aligned}m'(z) &= -1 \\g'(z) &= \exp(z) \\h'(z) &= 1 \\f'(z) &= -\frac{1}{z^2}\end{aligned}$$



Backpropagation – simple example

Model:

$$\begin{aligned}m(z) &= -z \\g(z) &= \exp(z) \\h(z) &= z + 1 \\f(z) &= \frac{1}{z}\end{aligned}$$

Data:

$$\begin{aligned}x_0 &= 1 \\x_1 &= 1.1\end{aligned}$$

Q: How does a small nudge in x influence $f(h(g(m(x))))$?

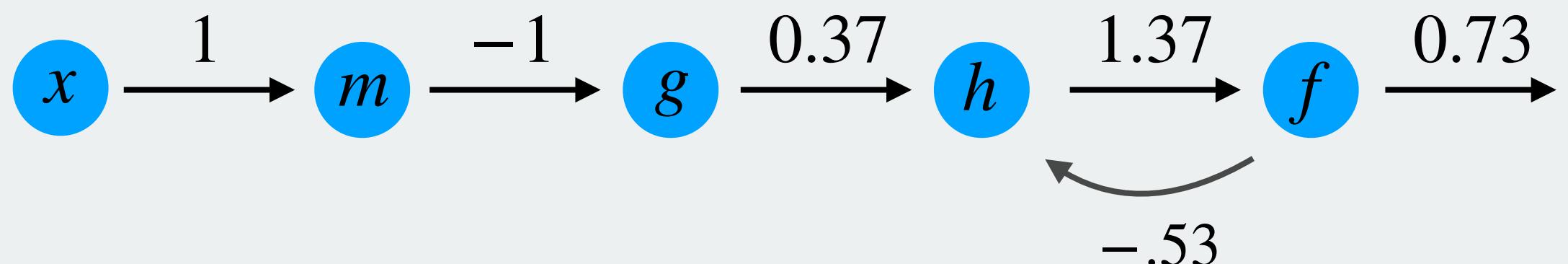
A: Propagate gradients backwards using the chain rule!

Chain rule from f to h :

$$\frac{df}{dh}$$

Model derivatives:

$$\begin{aligned}m'(z) &= -1 \\g'(z) &= \exp(z) \\h'(z) &= 1 \\f'(z) &= -\frac{1}{z^2}\end{aligned}$$



$$\frac{df}{dh} = -\frac{1}{1.37^2} = -0.53$$

Backpropagation – simple example

Model:

$$\begin{aligned}m(z) &= -z \\g(z) &= \exp(z) \\h(z) &= z + 1 \\f(z) &= \frac{1}{z}\end{aligned}$$

Data:

$$\begin{aligned}x_0 &= 1 \\x_1 &= 1.1\end{aligned}$$

Q: How does a small nudge in x influence $f(h(g(m(x))))$?

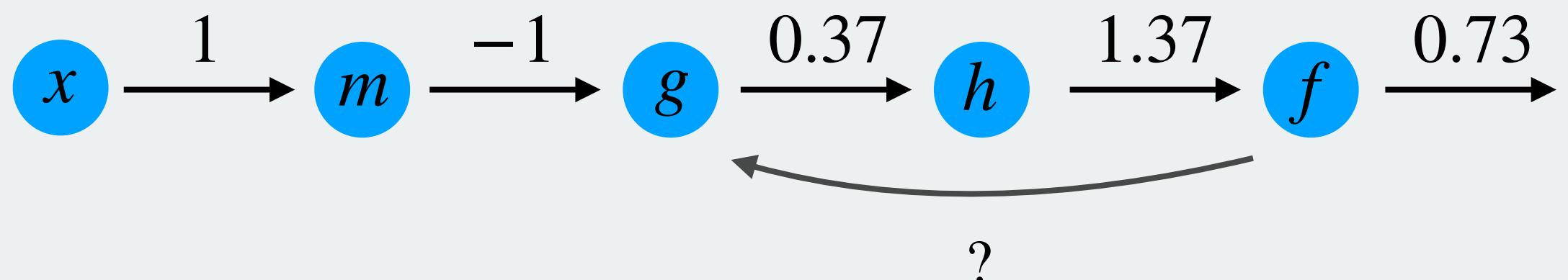
A: Propagate gradients backwards using the chain rule!

Chain rule from f to g :

$$\frac{df}{dg} = \frac{df}{dh} \frac{dh}{dg}$$

Model derivatives:

$$\begin{aligned}m'(z) &= -1 \\g'(z) &= \exp(z) \\h'(z) &= 1 \\f'(z) &= -\frac{1}{z^2}\end{aligned}$$



$$\frac{df}{dh} = -\frac{1}{1.37^2} = -.53$$

Backpropagation – simple example

Model:

$$\begin{aligned}m(z) &= -z \\g(z) &= \exp(z) \\h(z) &= z + 1 \\f(z) &= \frac{1}{z}\end{aligned}$$

Data:

$$\begin{aligned}x_0 &= 1 \\x_1 &= 1.1\end{aligned}$$

Q: How does a small nudge in x influence $f(h(g(m(x))))$?

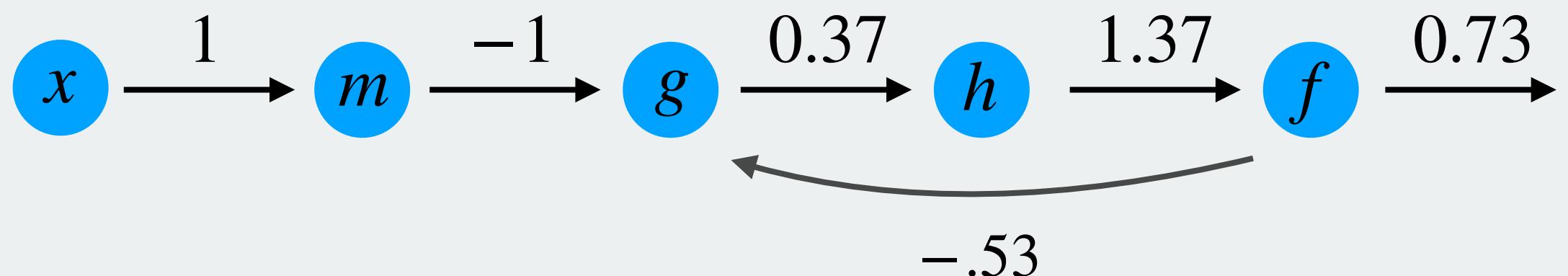
A: Propagate gradients backwards using the chain rule!

Chain rule from f to g :

$$\frac{df}{dg} = \frac{df}{dh} \frac{dh}{dg}$$

Model derivatives:

$$\begin{aligned}m'(z) &= -1 \\g'(z) &= \exp(z) \\h'(z) &= 1 \\f'(z) &= -\frac{1}{z^2}\end{aligned}$$



$$\frac{df}{dg} = -\frac{1}{1.37^2} \cdot 1 = -0.53$$

Backpropagation – simple example

Model:

$$m(z) = -z$$

$$g(z) = \exp(z)$$

$$h(z) = z + 1$$

$$f(z) = \frac{1}{z}$$

Data:

$$x_0 = 1$$

$$x_1 = 1.1$$

Q: How does a small nudge in x influence $f(h(g(m(x))))$?

A: Propagate gradients backwards using the chain rule!

Chain rule from f to m :

$$\frac{df}{dm} = \frac{df}{dh} \frac{dh}{dg} \frac{dg}{dm}$$

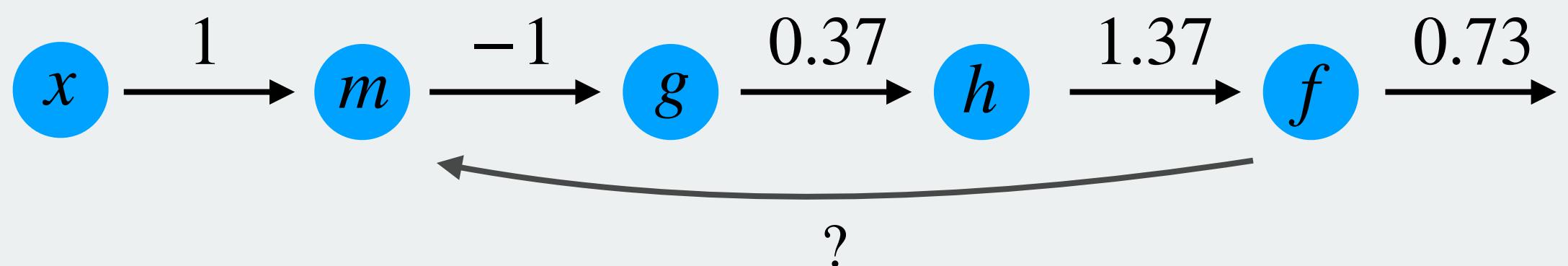
Model derivatives:

$$m'(z) = -1$$

$$g'(z) = \exp(z)$$

$$h'(z) = 1$$

$$f'(z) = -\frac{1}{z^2}$$



$$\frac{df}{dg} = -\frac{1}{1.37^2} \cdot 1 = -0.53$$

Backpropagation – simple example

Model:

$$m(z) = -z$$

$$g(z) = \exp(z)$$

$$h(z) = z + 1$$

$$f(z) = \frac{1}{z}$$

Data:

$$x_0 = 1$$

$$x_1 = 1.1$$

Q: How does a small nudge in x influence $f(h(g(m(x))))$?

A: Propagate gradients backwards using the chain rule!

Chain rule from f to m :

$$\frac{df}{dm} = \frac{df}{dh} \frac{dh}{dg} \frac{dg}{dm}$$

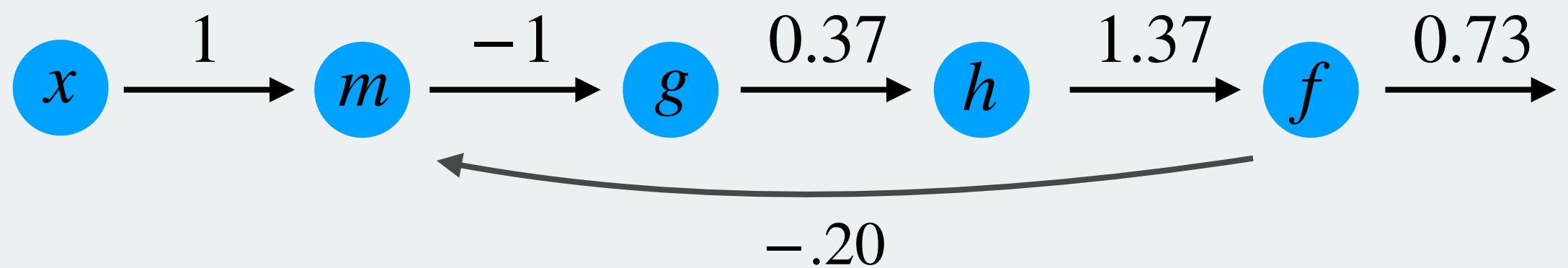
Model derivatives:

$$m'(z) = -1$$

$$g'(z) = \exp(z)$$

$$h'(z) = 1$$

$$f'(z) = -\frac{1}{z^2}$$



$$\frac{df}{dm} = -\frac{1}{1.37^2} \cdot 1 \cdot e^{-1} = -0.20$$

Backpropagation – simple example

Model:

$$\begin{aligned}m(z) &= -z \\g(z) &= \exp(z) \\h(z) &= z + 1 \\f(z) &= \frac{1}{z}\end{aligned}$$

Data:

$$\begin{aligned}x_0 &= 1 \\x_1 &= 1.1\end{aligned}$$

Q: How does a small nudge in x influence $f(h(g(m(x))))$?

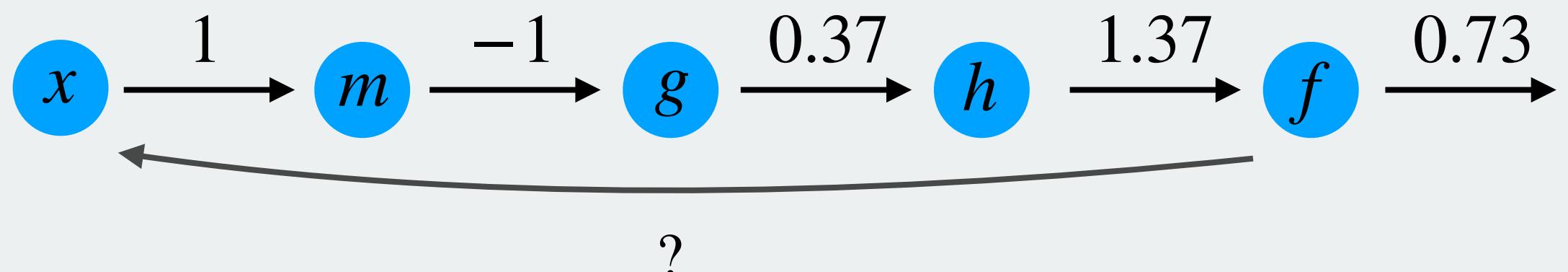
A: Propagate gradients backwards using the chain rule!

Chain rule from f to x :

$$\frac{df}{dx} = \frac{df}{dh} \frac{dh}{dg} \frac{dg}{dm} \frac{dm}{dx}$$

Model derivatives:

$$\begin{aligned}m'(z) &= -1 \\g'(z) &= \exp(z) \\h'(z) &= 1 \\f'(z) &= -\frac{1}{z^2}\end{aligned}$$



$$\frac{df}{dm} = -\frac{1}{1.37^2} \cdot 1 \cdot e^{-1} = -.20$$

Backpropagation – simple example

Model:

$$\begin{aligned}m(z) &= -z \\g(z) &= \exp(z) \\h(z) &= z + 1 \\f(z) &= \frac{1}{z}\end{aligned}$$

Data:

$$\begin{aligned}x_0 &= 1 \\x_1 &= 1.1\end{aligned}$$

Q: How does a small nudge in x influence $f(h(g(m(x))))$?

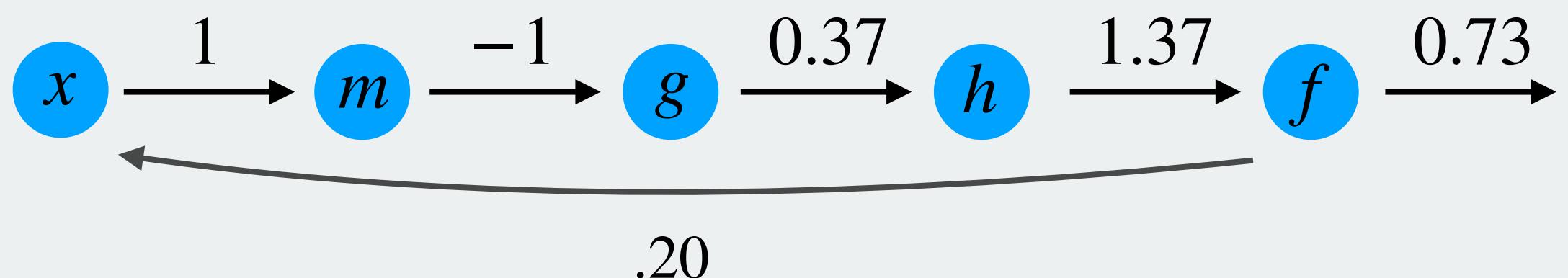
A: Propagate gradients backwards using the chain rule!

Chain rule from f to x :

$$\frac{df}{dx} = \frac{df}{dh} \frac{dh}{dg} \frac{dg}{dm} \frac{dm}{dx}$$

Model derivatives:

$$\begin{aligned}m'(z) &= -1 \\g'(z) &= \exp(z) \\h'(z) &= 1 \\f'(z) &= -\frac{1}{z^2}\end{aligned}$$



$$\frac{df}{dx} = -\frac{1}{1.37^2} \cdot 1 \cdot e^{-1} \cdot (-1) = .20$$

Backpropagation – simple example

Model:

$$\begin{aligned}m(z) &= -z \\g(z) &= \exp(z) \\h(z) &= z + 1 \\f(z) &= \frac{1}{z}\end{aligned}$$

Data:

$$\begin{aligned}x_0 &= 1 \\x_1 &= 1.1\end{aligned}$$

Q: How does a small nudge in x influence $f(h(g(m(x))))$?

A: Propagate gradients backwards using the chain rule!

Chain rule:

$$\frac{df}{dx} = \frac{df}{dh} \frac{dh}{dg} \frac{dg}{dm} \frac{dm}{dx}$$

Model derivatives:

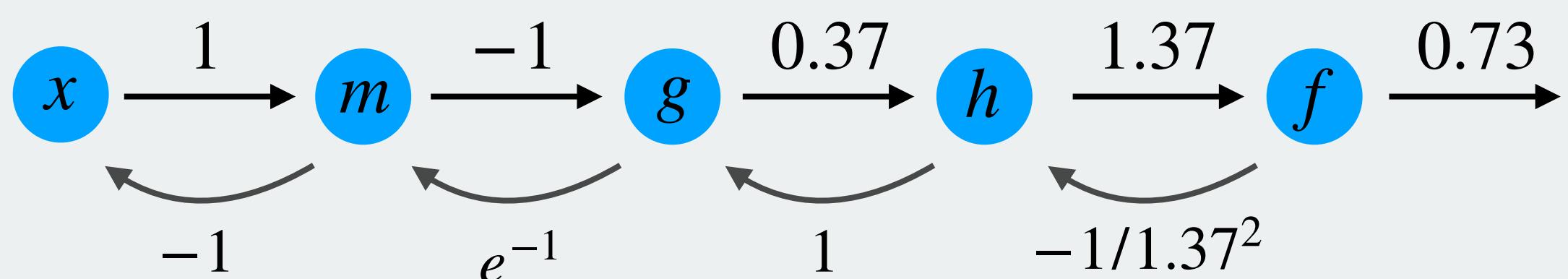
$$\begin{aligned}m'(z) &= -1 \\g'(z) &= \exp(z) \\h'(z) &= 1 \\f'(z) &= -\frac{1}{z^2}\end{aligned}$$

Sigmoid function:

$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$

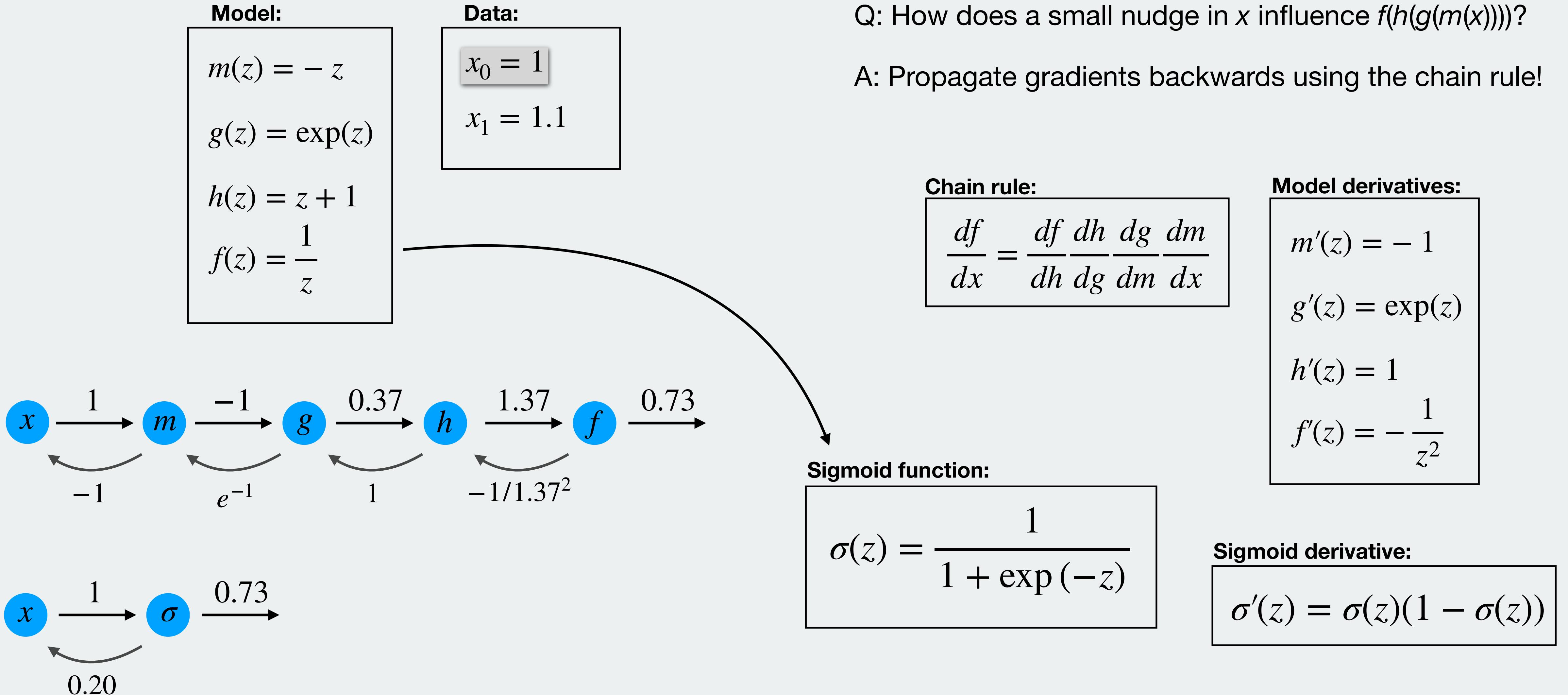
Sigmoid derivative:

$$\sigma'(z) = \sigma(z)(1 - \sigma(z))$$



$$\frac{df}{dx} = -\frac{1}{1.37^2} \cdot 1 \cdot e^{-1} \cdot (-1) = .20$$

Backpropagation – simple example



Backpropagation – simple example

Model:

$$m(z) = -z$$

$$g(z) = \exp(z)$$

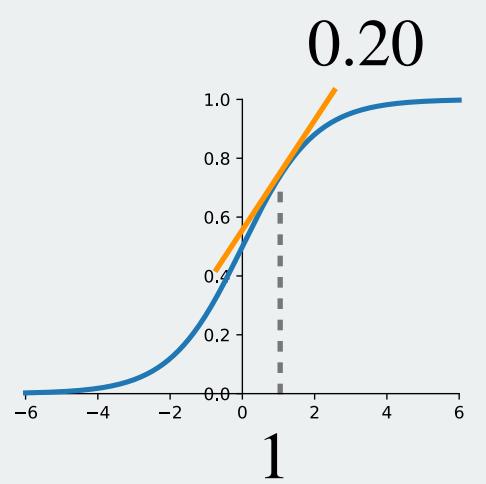
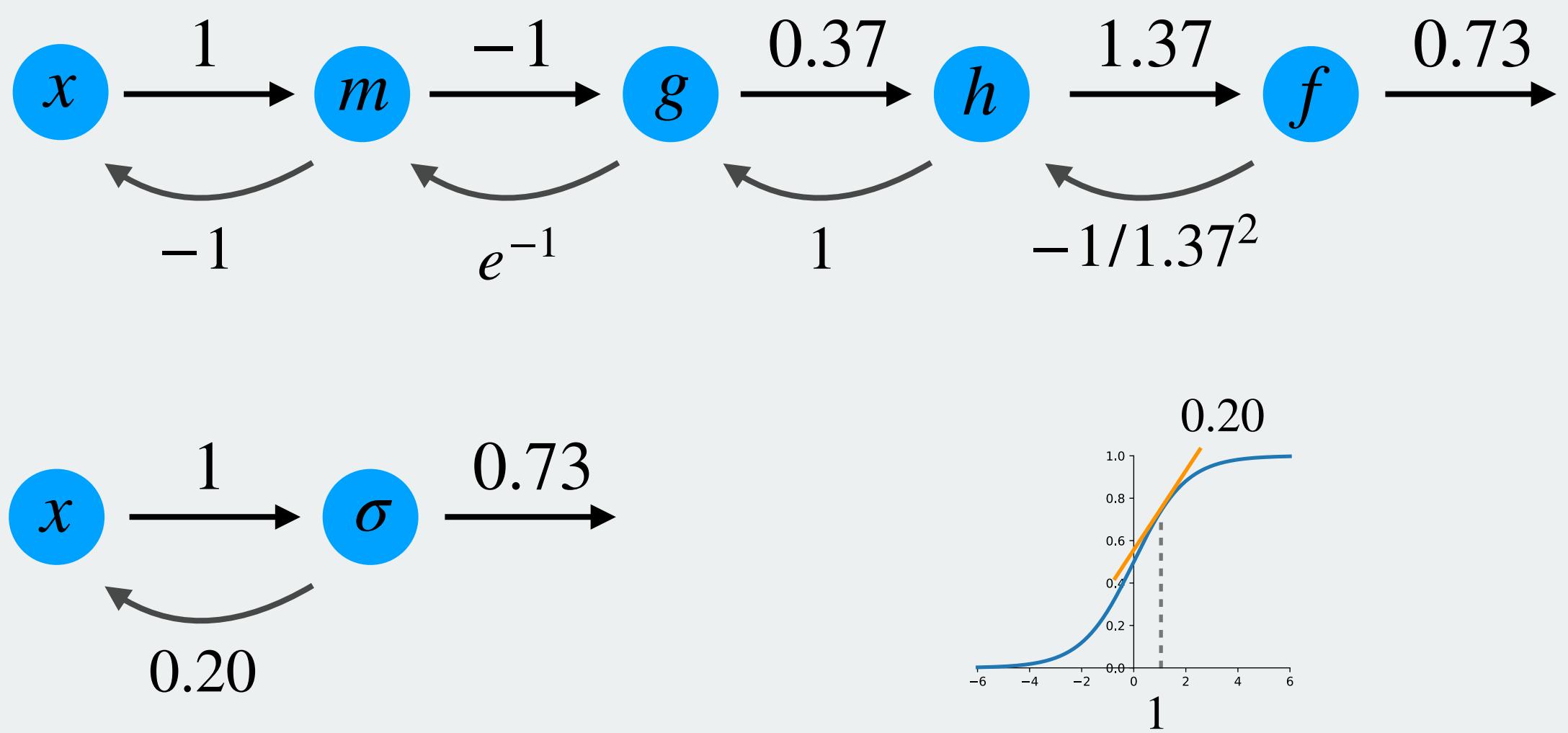
$$h(z) = z + 1$$

$$f(z) = \frac{1}{z}$$

Data:

$$x_0 = 1$$

$$x_1 = 1.1$$



Q: How does a small nudge in x influence $f(h(g(m(x))))$?

A: Propagate gradients backwards using the chain rule!

Chain rule:

$$\frac{df}{dx} = \frac{df}{dh} \frac{dh}{dg} \frac{dg}{dm} \frac{dm}{dx}$$

Model derivatives:

$$m'(z) = -1$$

$$g'(z) = \exp(z)$$

$$h'(z) = 1$$

$$f'(z) = -\frac{1}{z^2}$$

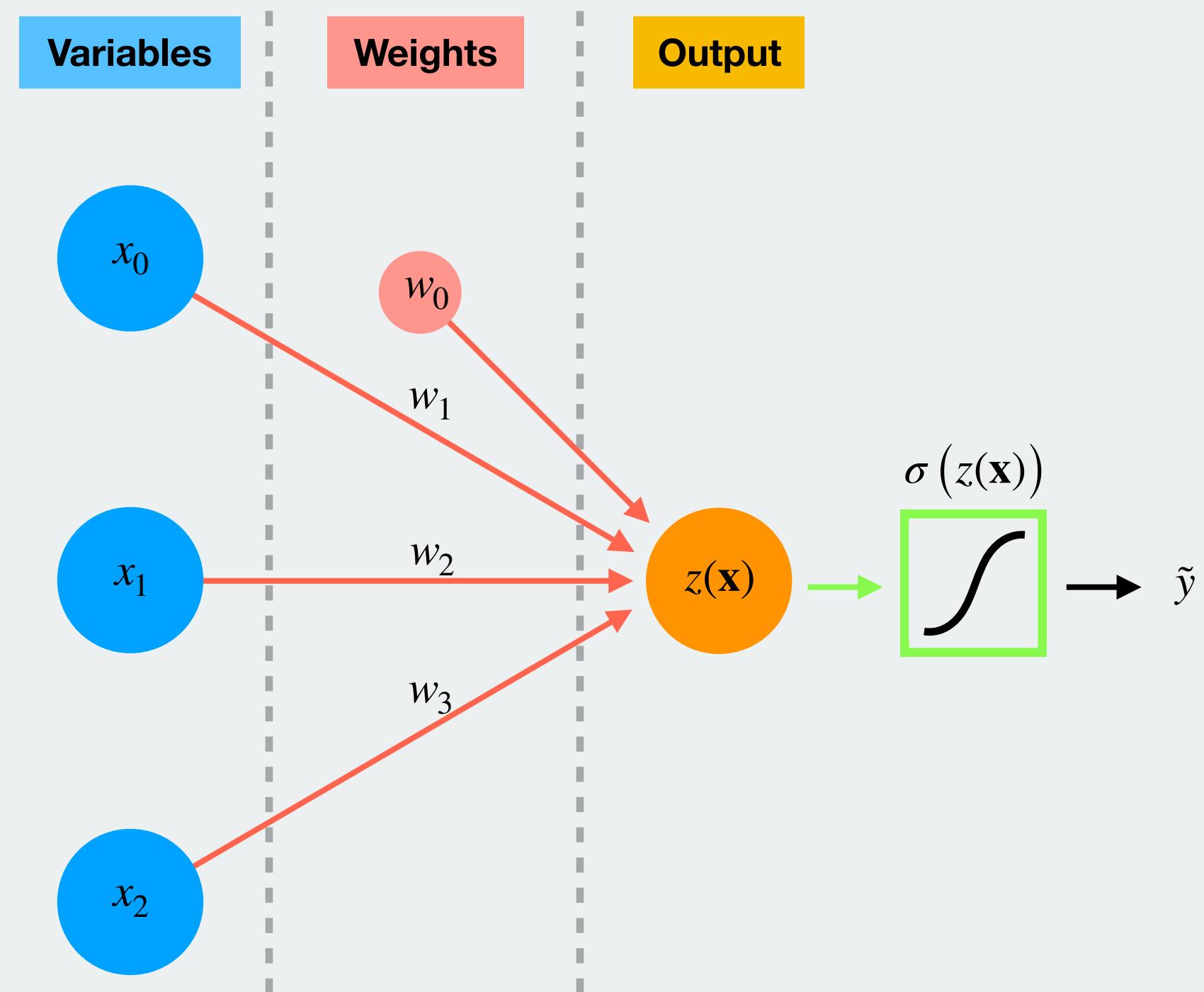
Sigmoid function:

$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$

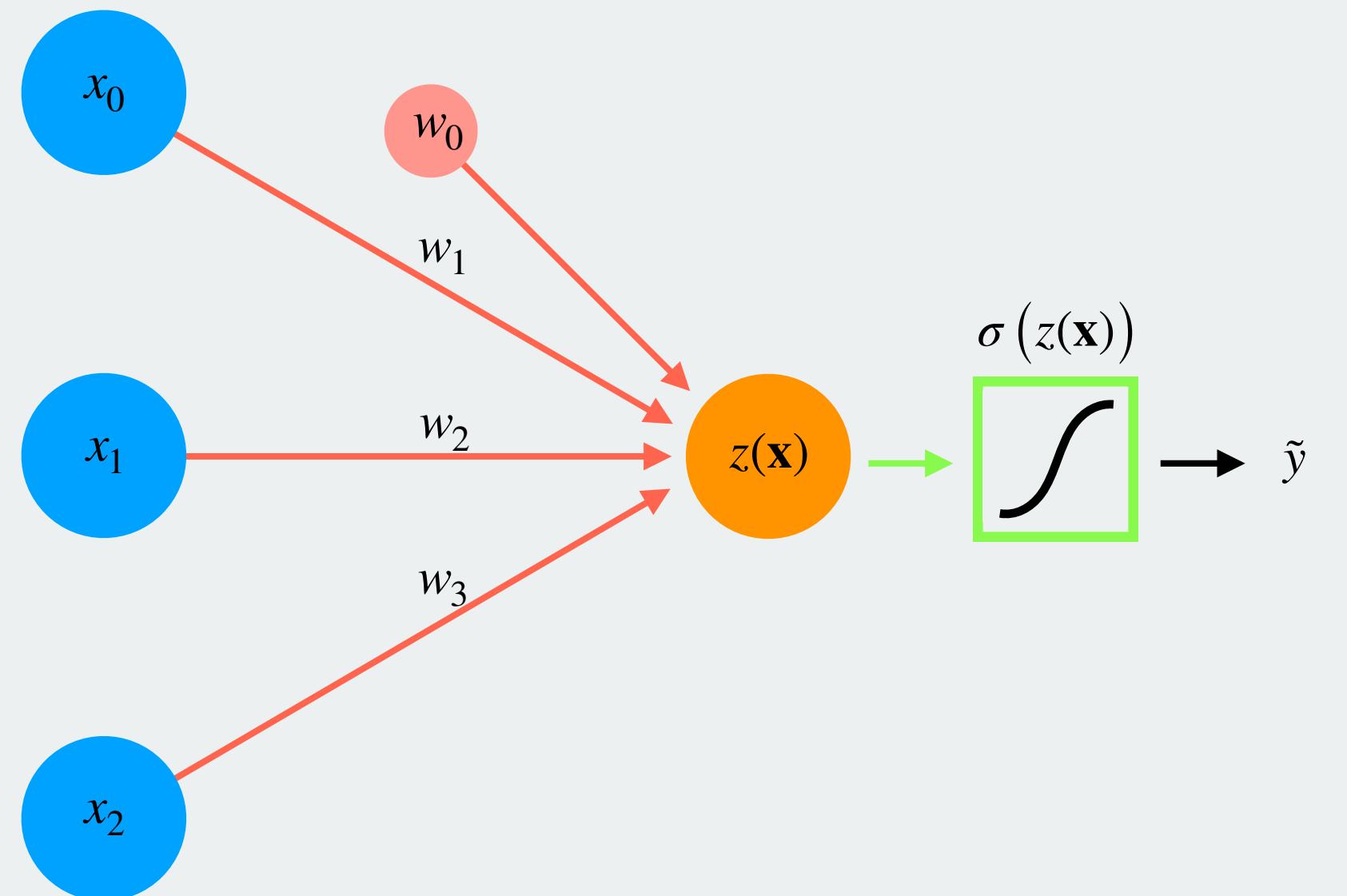
Sigmoid derivative:

$$\sigma'(z) = \sigma(z)(1 - \sigma(z))$$

Backpropagation – on a neural network



Backpropagation – on a neural network



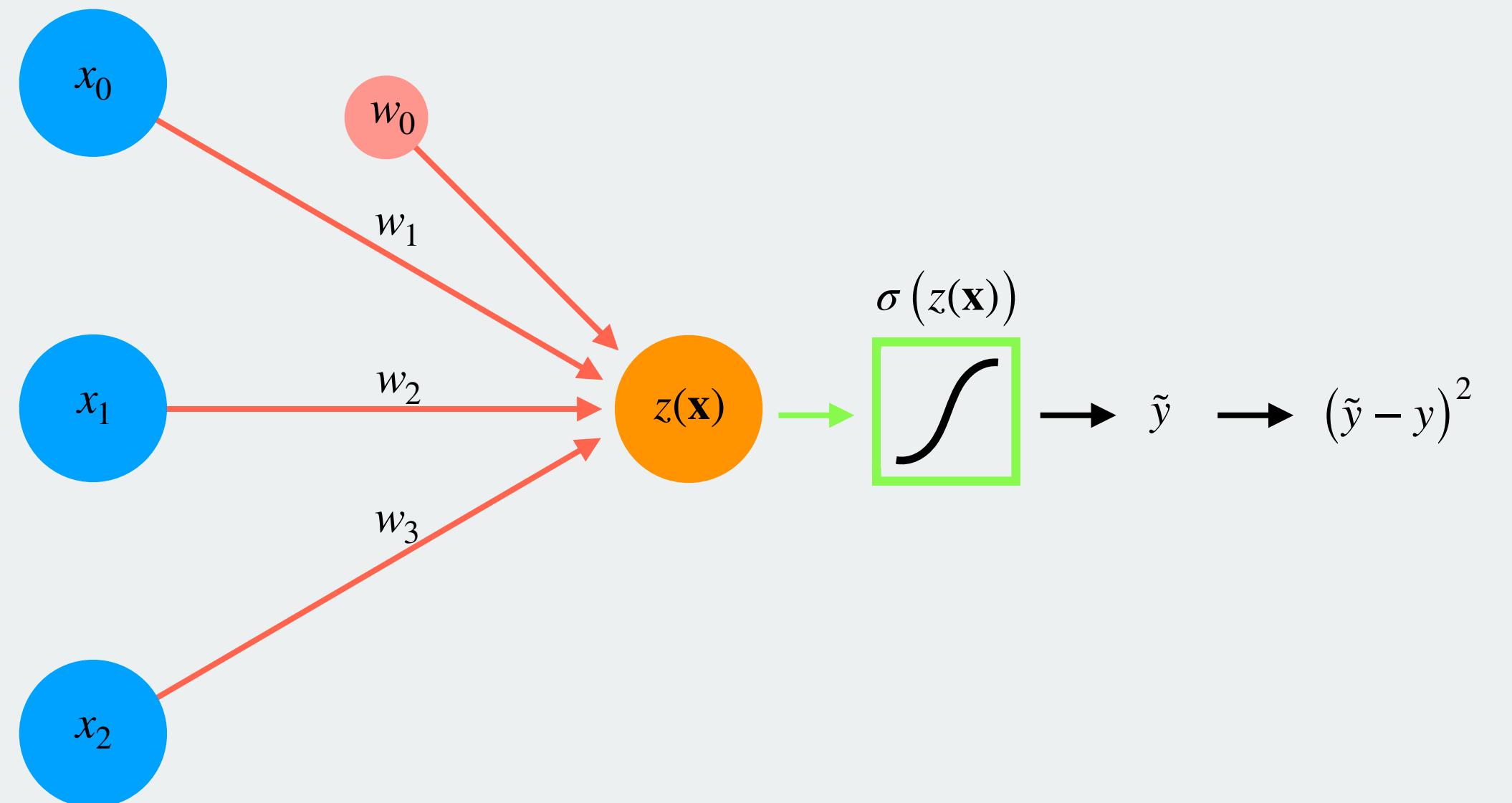
Model:

$$w_0 + x_0 w_1 + x_1 w_2 + x_2 w_3 = z(\mathbf{x})$$

$$\sigma(x) = \frac{1}{1 + \exp(-x)}$$

$$C(\tilde{y}, y) = (\tilde{y} - y)^2$$

Backpropagation – on a neural network



Model:

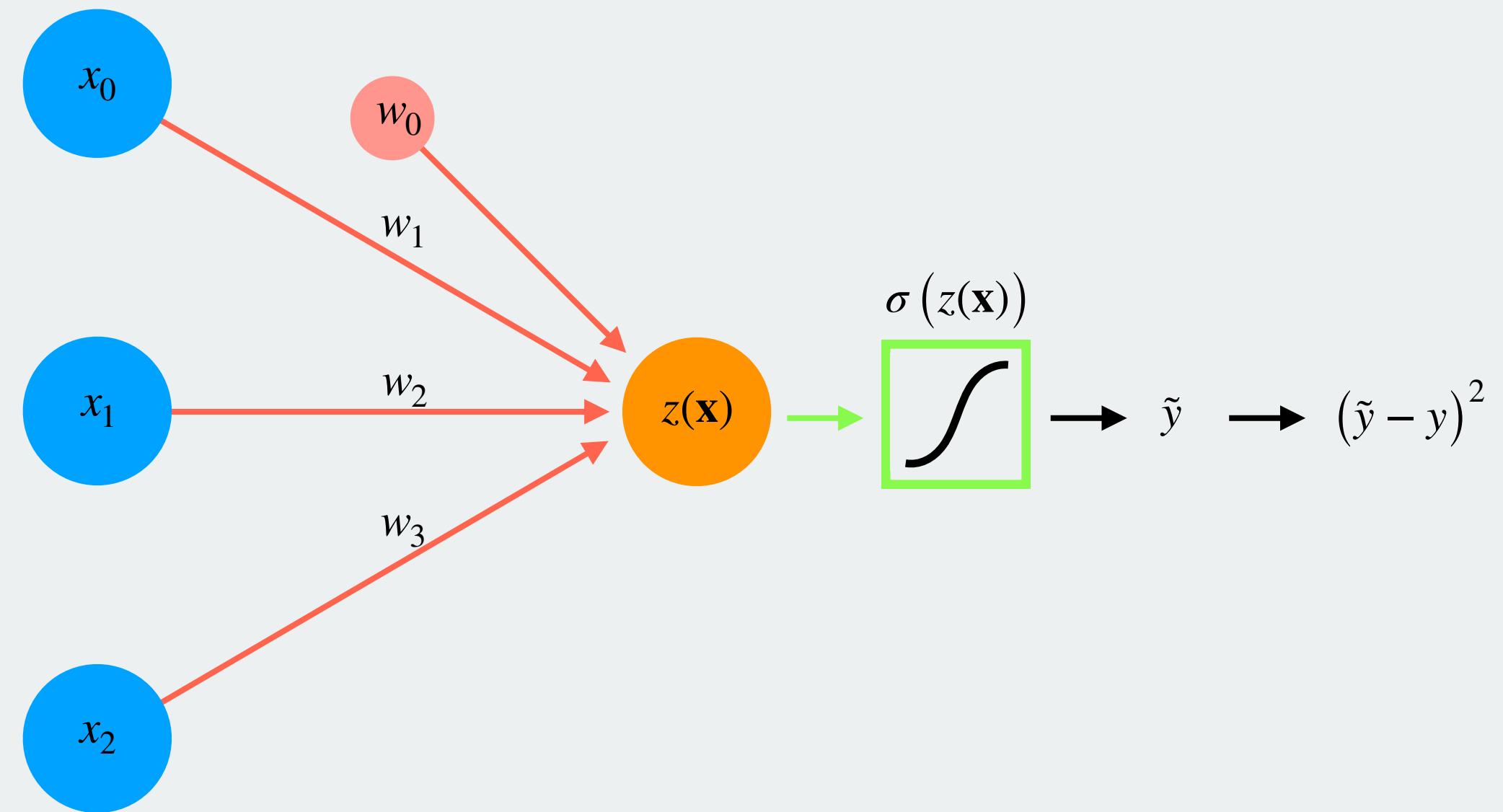
$$w_0 + x_0 w_1 + x_1 w_2 + x_2 w_3 = z(\mathbf{x})$$

$$\sigma(x) = \frac{1}{1 + \exp(-x)}$$

$$C(\tilde{y}, y) = (\tilde{y} - y)^2$$

Backpropagation – on a neural network

As a computational graph:

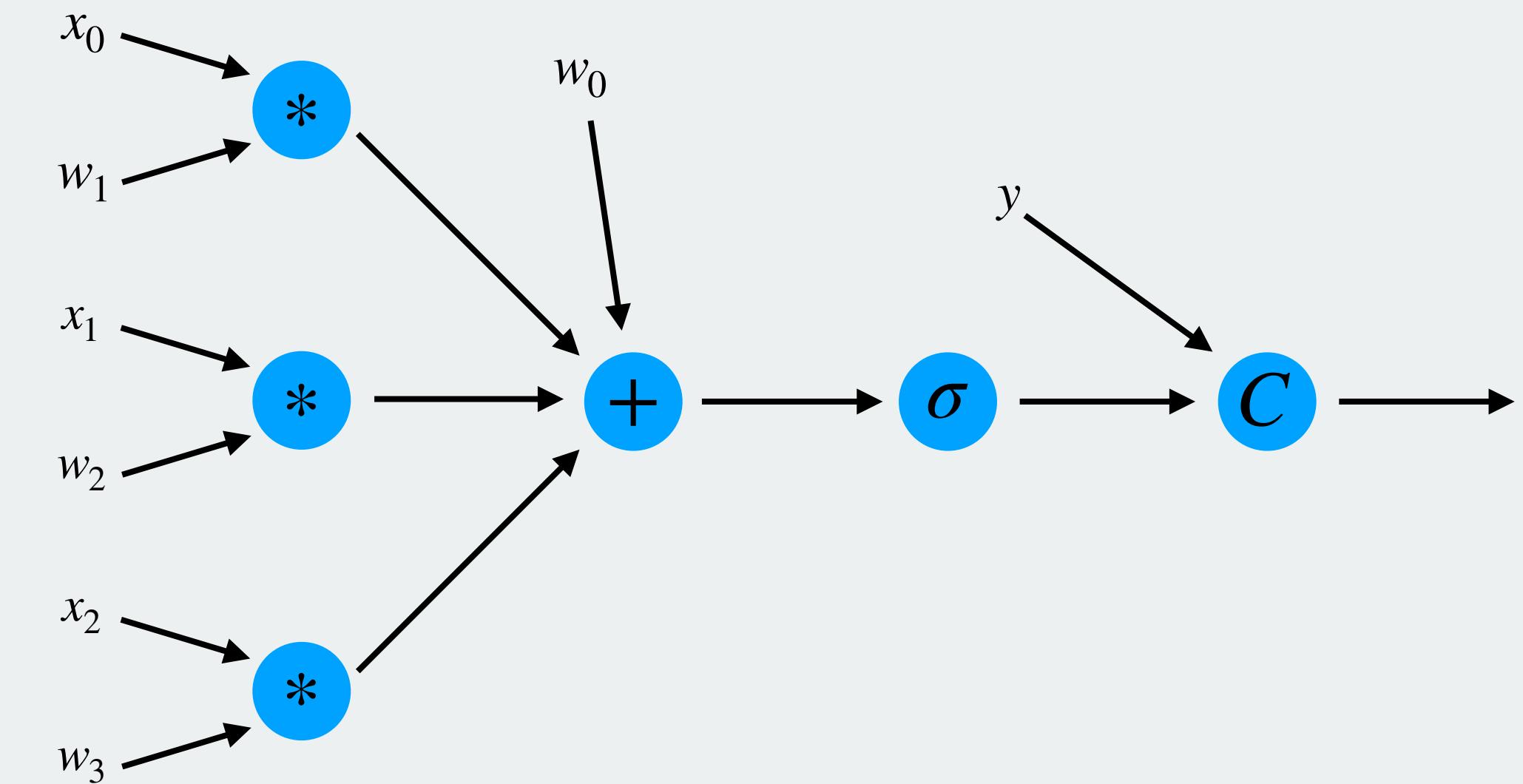


Model:

$$w_0 + x_0 w_1 + x_1 w_2 + x_2 w_3 = z(\mathbf{x})$$

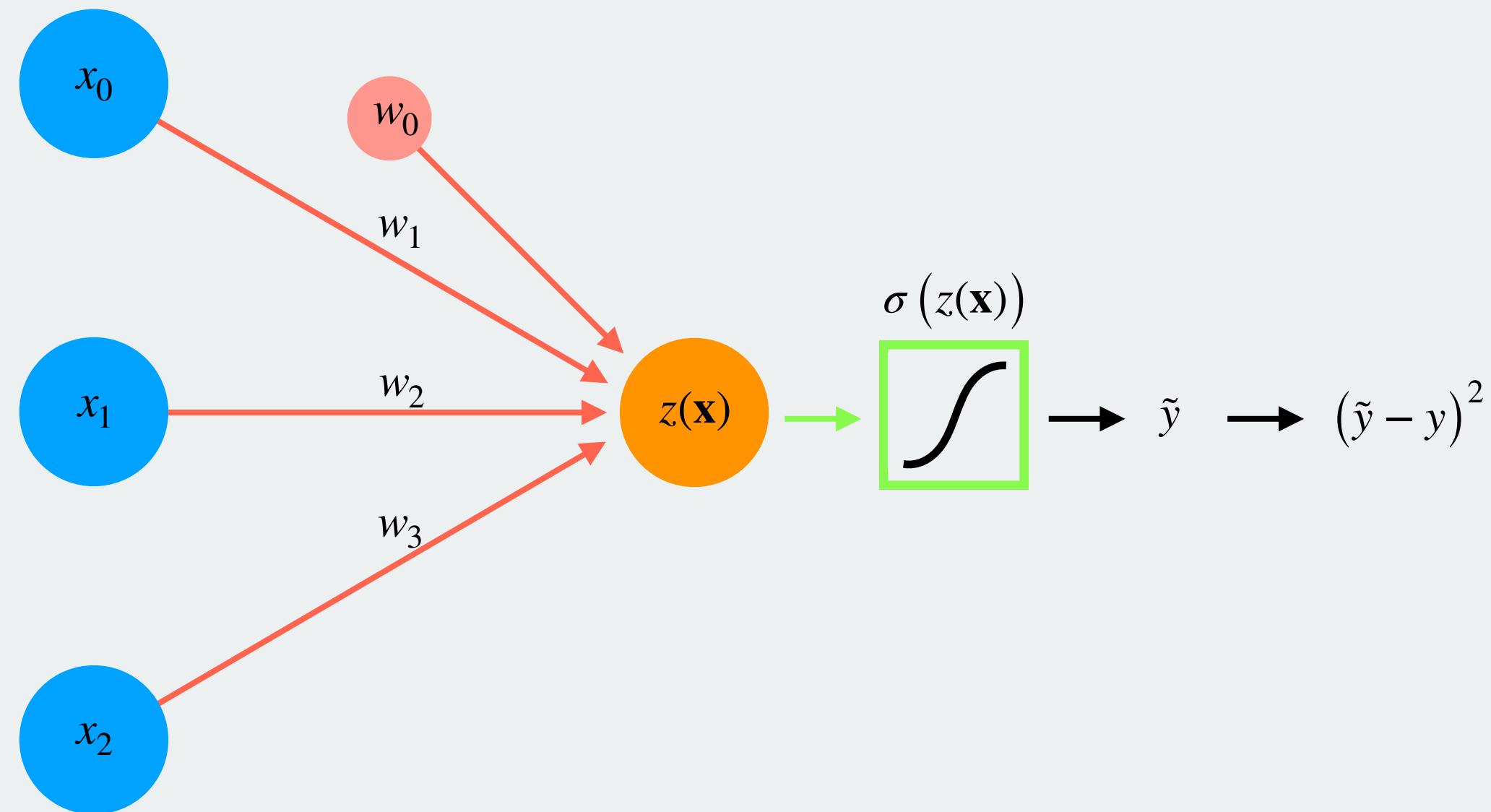
$$\sigma(x) = \frac{1}{1 + \exp(-x)}$$

$$C(\tilde{y}, y) = (\tilde{y} - y)^2$$



Backpropagation – on a neural network

As a computational graph:

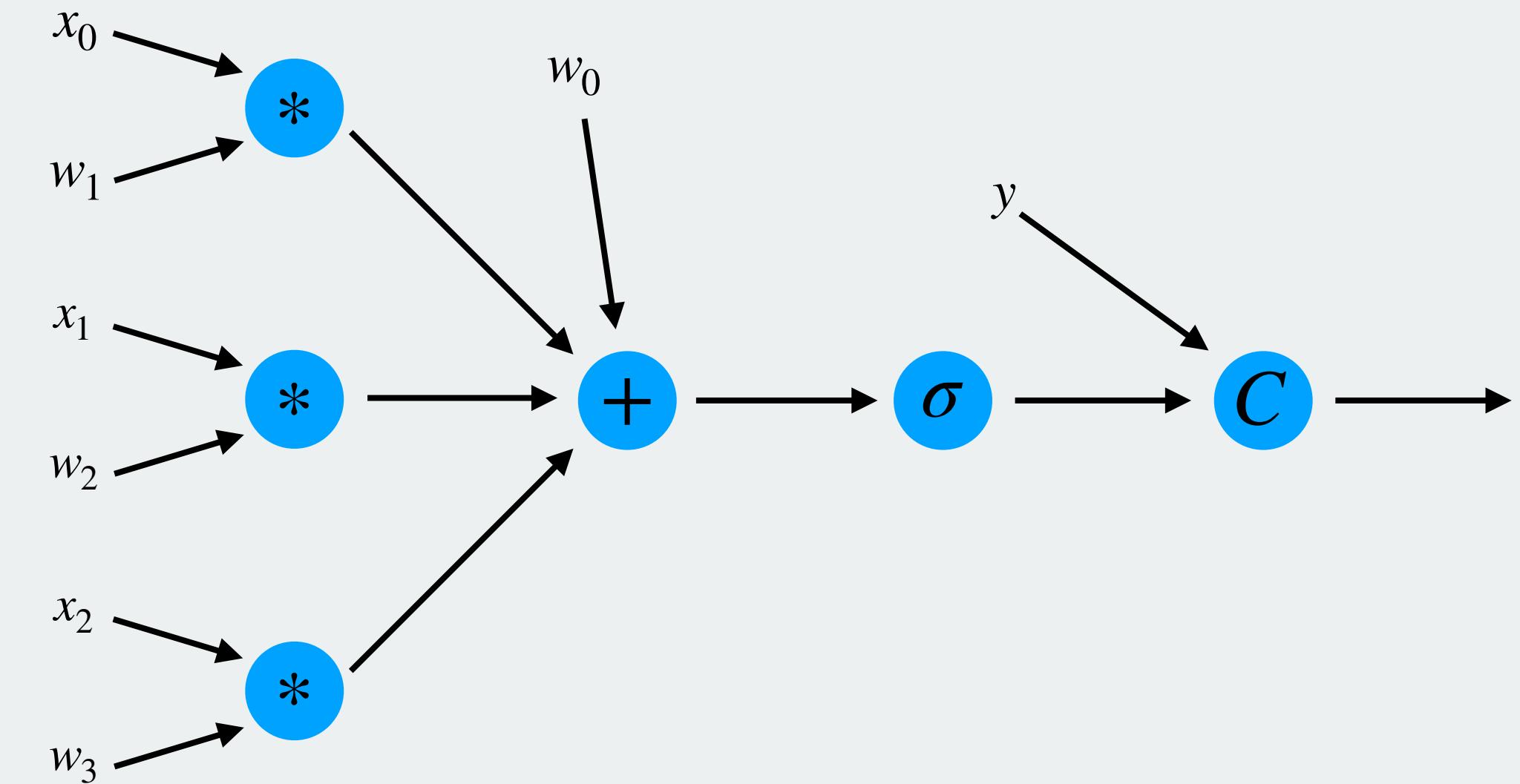


Model:

$$w_0 + x_0w_1 + x_1w_2 + x_2w_3 = z(\mathbf{x})$$

$$\sigma(x) = \frac{1}{1 + \exp(-x)}$$

$$C(\tilde{y}, y) = (\tilde{y} - y)^2$$



Weights:

$$\mathbf{b} = \begin{bmatrix} -2 \end{bmatrix}$$

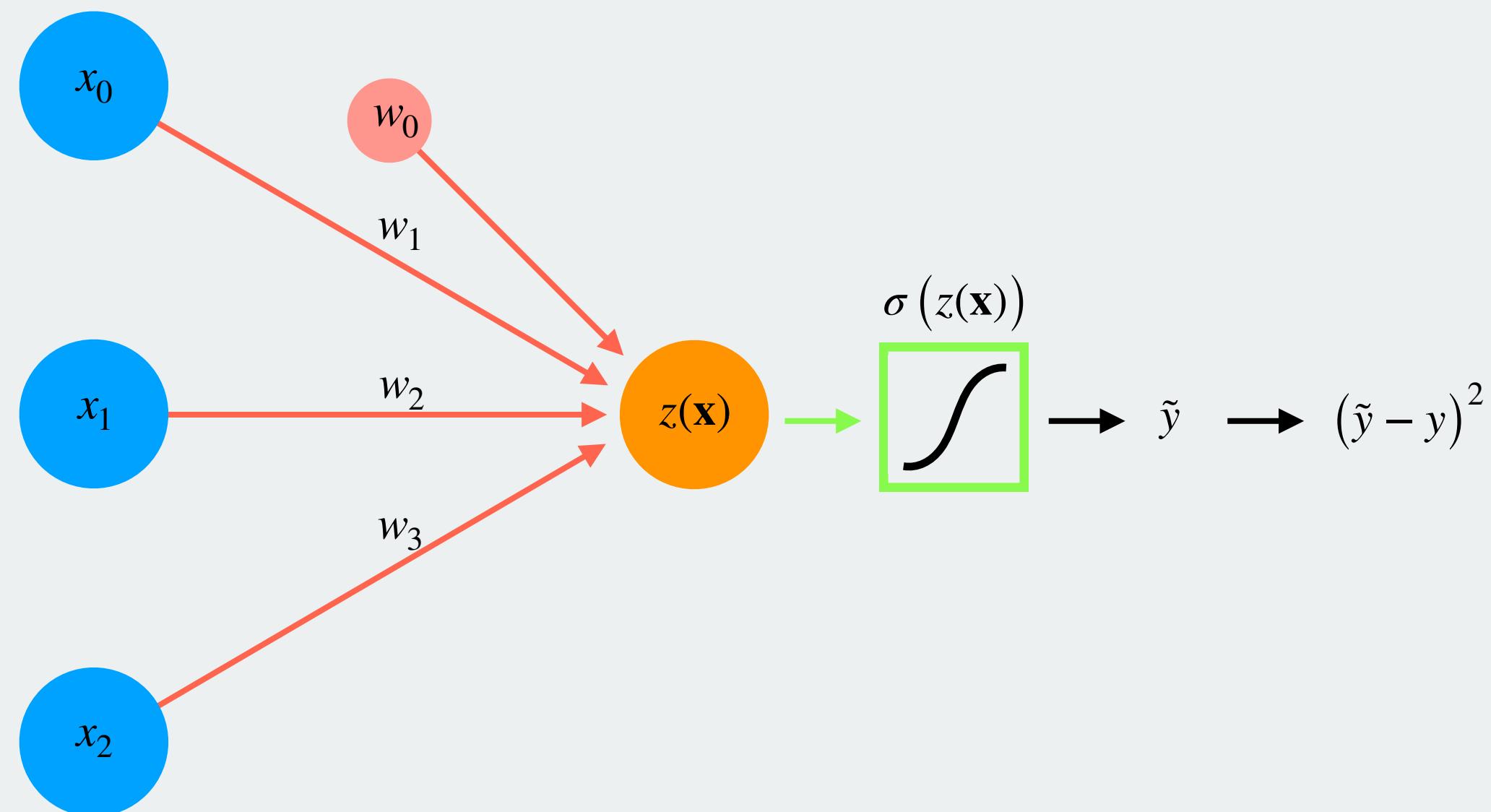
$$\mathbf{W} = \begin{bmatrix} -1 & 0.5 & 10 \end{bmatrix}$$

Data:

$$\mathbf{x} = \begin{bmatrix} 4 \\ -2 \\ 1 \end{bmatrix}$$

$$\mathbf{y} = \begin{bmatrix} 1 \end{bmatrix}$$

Backpropagation – on a neural network



Model:

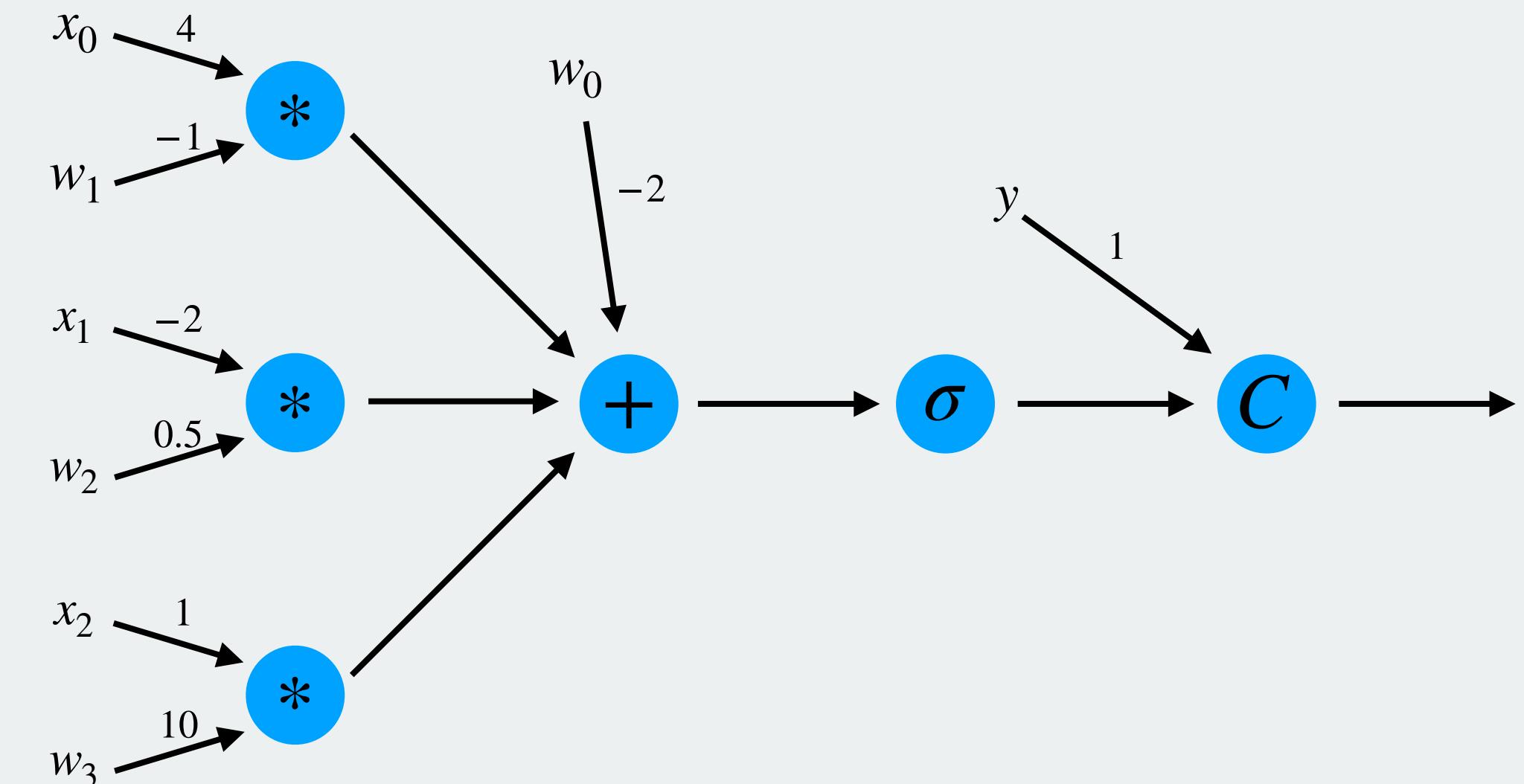
$$w_0 + x_0 w_1 + x_1 w_2 + x_2 w_3 = z(\mathbf{x})$$

$$\sigma(x) = \frac{1}{1 + \exp(-x)}$$

$$C(\tilde{y}, y) = (\tilde{y} - y)^2$$

As a computational graph:

Forward pass



Weights:

$$\mathbf{b} = [-2]$$

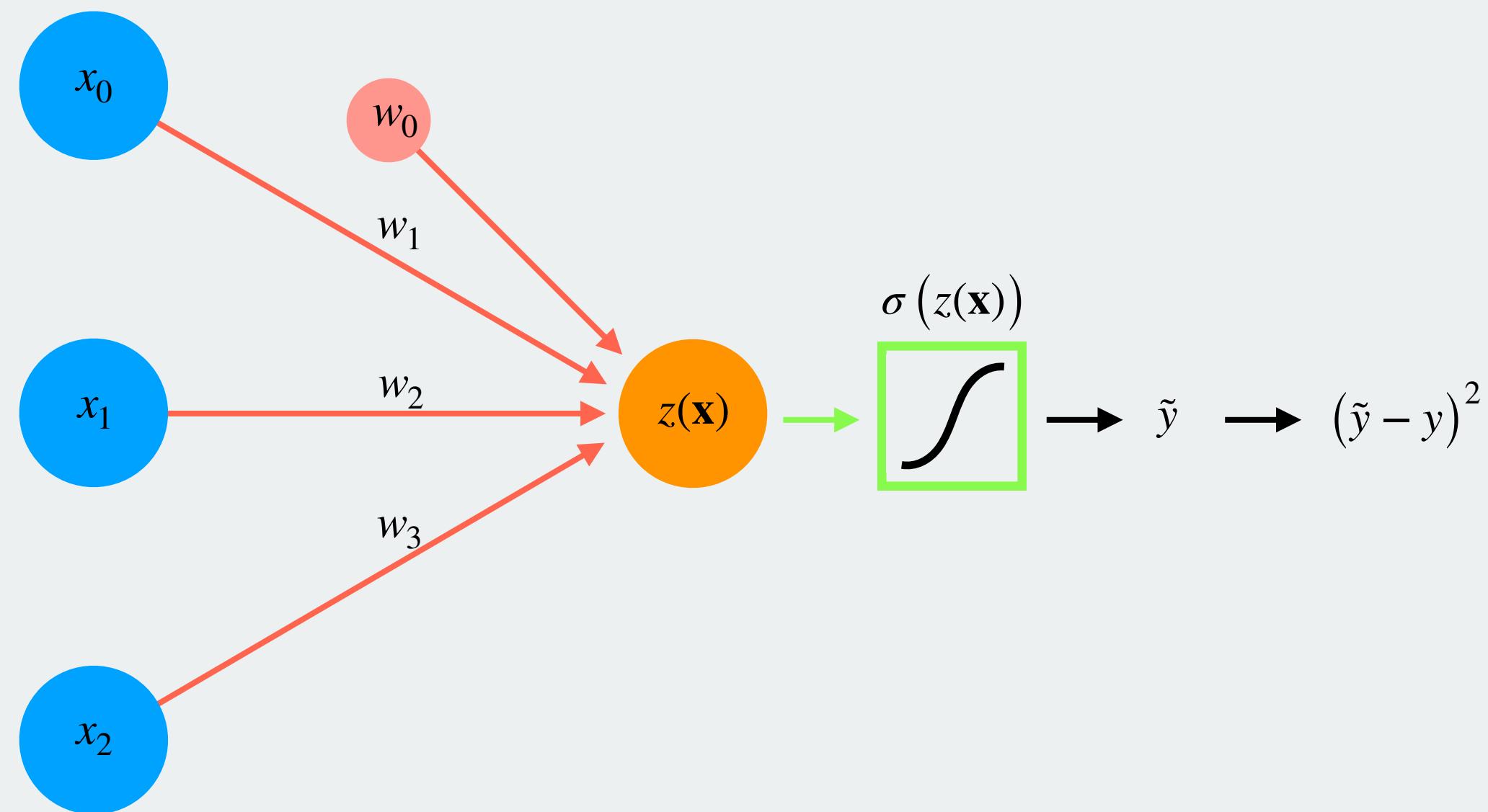
$$\mathbf{W} = \begin{bmatrix} -1 & 0.5 & 10 \end{bmatrix}$$

Data:

$$\mathbf{x} = \begin{bmatrix} 4 \\ -2 \\ 1 \end{bmatrix}$$

$$\mathbf{y} = \begin{bmatrix} 1 \end{bmatrix}$$

Backpropagation – on a neural network



Model:

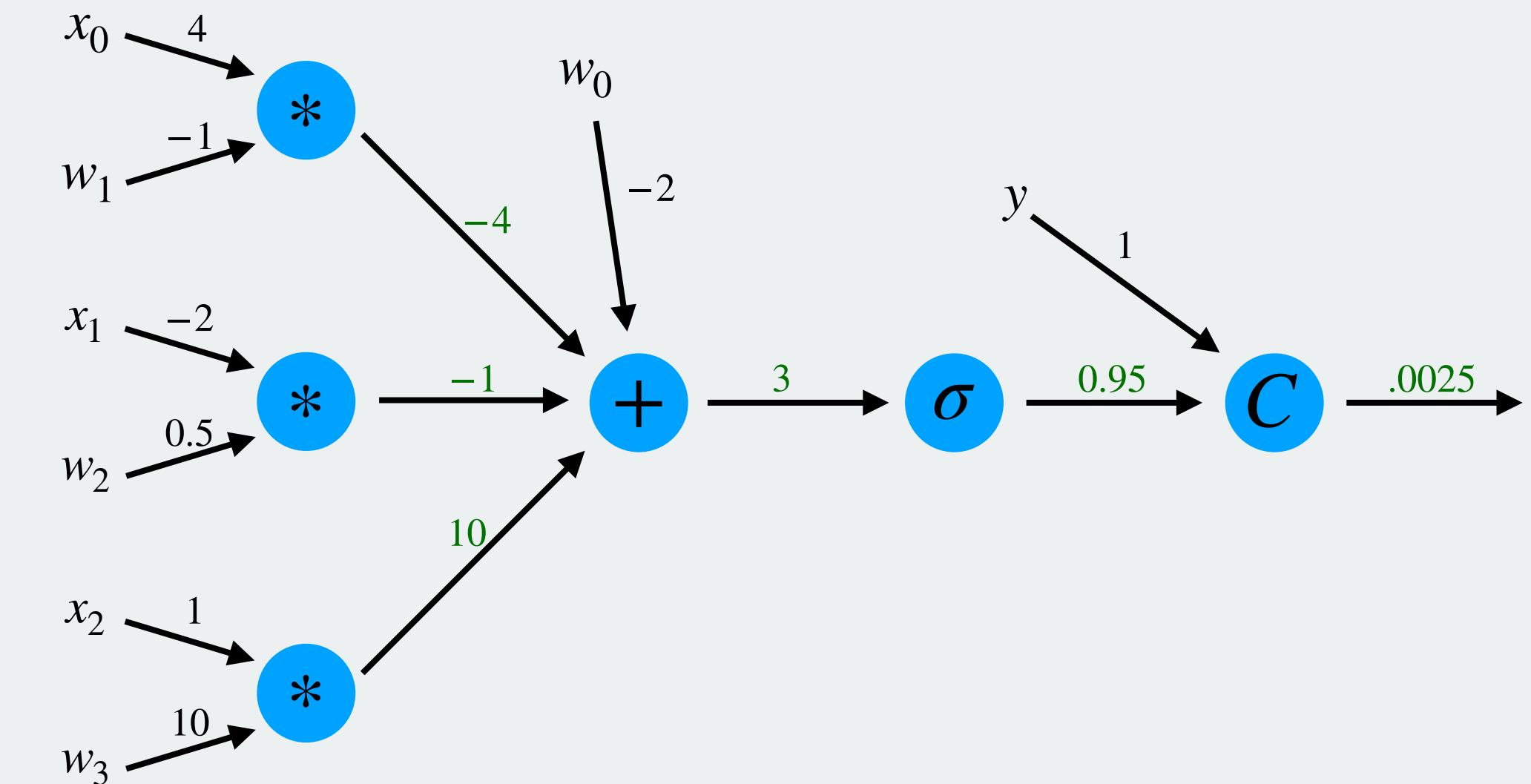
$$w_0 + x_0 w_1 + x_1 w_2 + x_2 w_3 = z(\mathbf{x})$$

$$\sigma(x) = \frac{1}{1 + \exp(-x)}$$

$$C(\tilde{y}, y) = (\tilde{y} - y)^2$$

As a computational graph:

Forward pass



Weights:

$$\mathbf{b} = \begin{bmatrix} -2 \\ 3 \end{bmatrix}$$

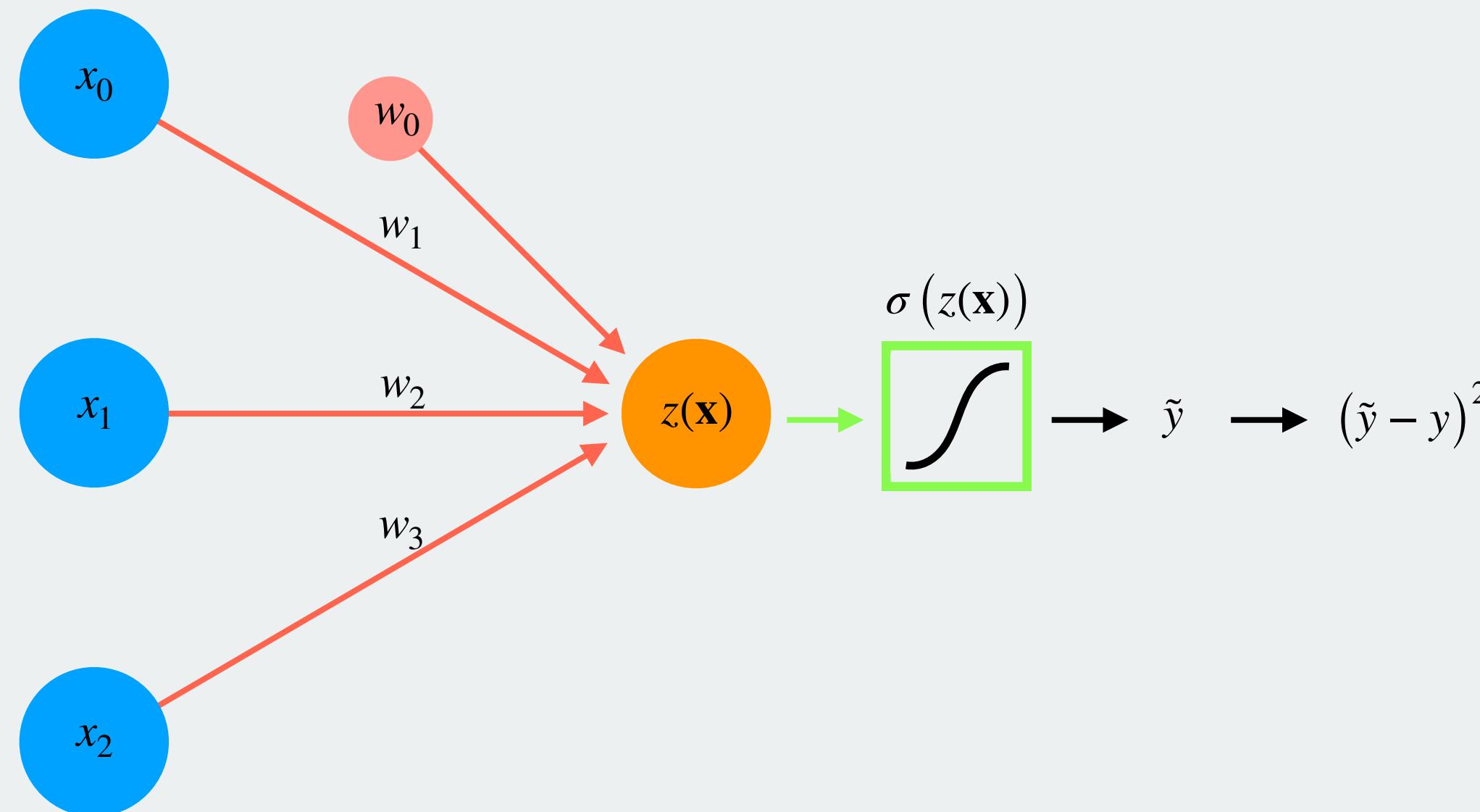
$$\mathbf{W} = \begin{bmatrix} -1 & 0.5 & 10 \end{bmatrix}$$

Data:

$$\mathbf{x} = \begin{bmatrix} 4 \\ -2 \\ 1 \end{bmatrix}$$

$$\mathbf{y} = \begin{bmatrix} 1 \end{bmatrix}$$

Backpropagation – on a neural network



Model:

$$w_0 + x_0 w_1 + x_1 w_2 + x_2 w_3 = z(\mathbf{x})$$

$$\sigma(x) = \frac{1}{1 + \exp(-x)}$$

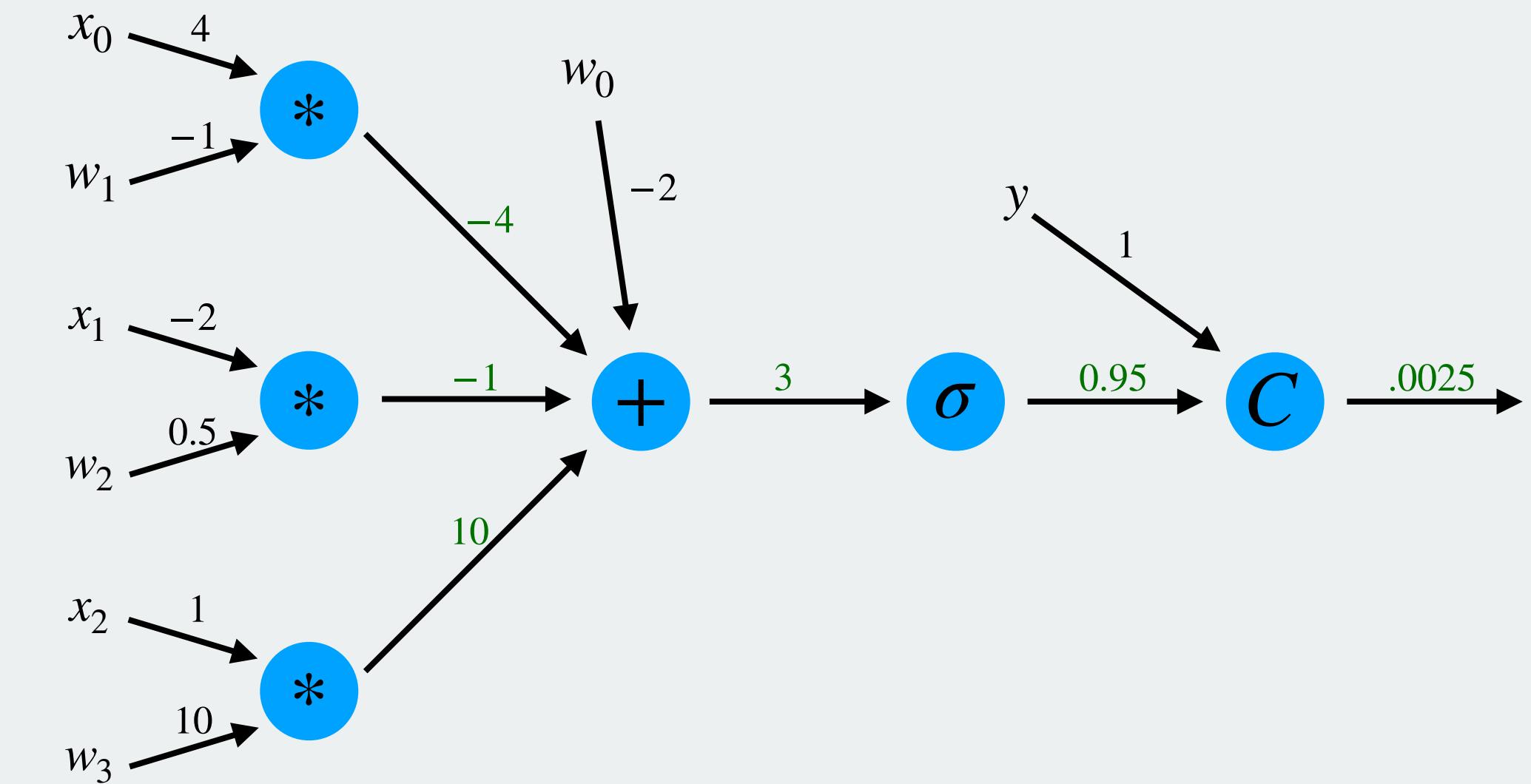
$$C(\tilde{y}, y) = (\tilde{y} - y)^2$$

Model derivatives:

$$C'(\tilde{y}, y) = ?$$

As a computational graph:

Backward pass



Weights:

$$\mathbf{b} = [-2]$$

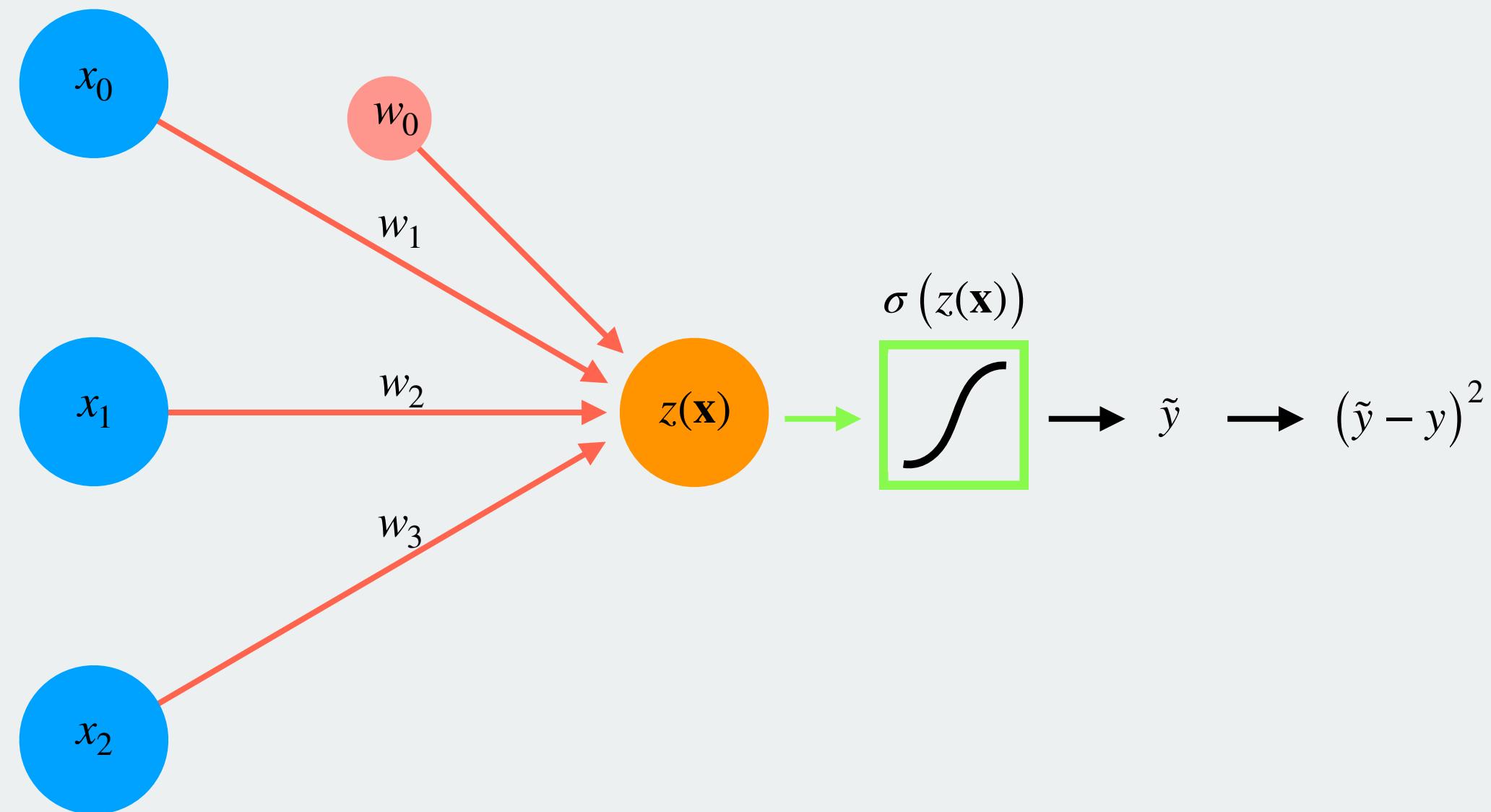
$$\mathbf{W} = [-1 \ 0.5 \ 10]$$

Data:

$$\mathbf{x} = \begin{bmatrix} 4 \\ -2 \\ 1 \end{bmatrix}$$

$$\mathbf{y} = \begin{bmatrix} 1 \end{bmatrix}$$

Backpropagation – on a neural network



Model:

$$w_0 + x_0 w_1 + x_1 w_2 + x_2 w_3 = z(\mathbf{x})$$

$$\sigma(x) = \frac{1}{1 + \exp(-x)}$$

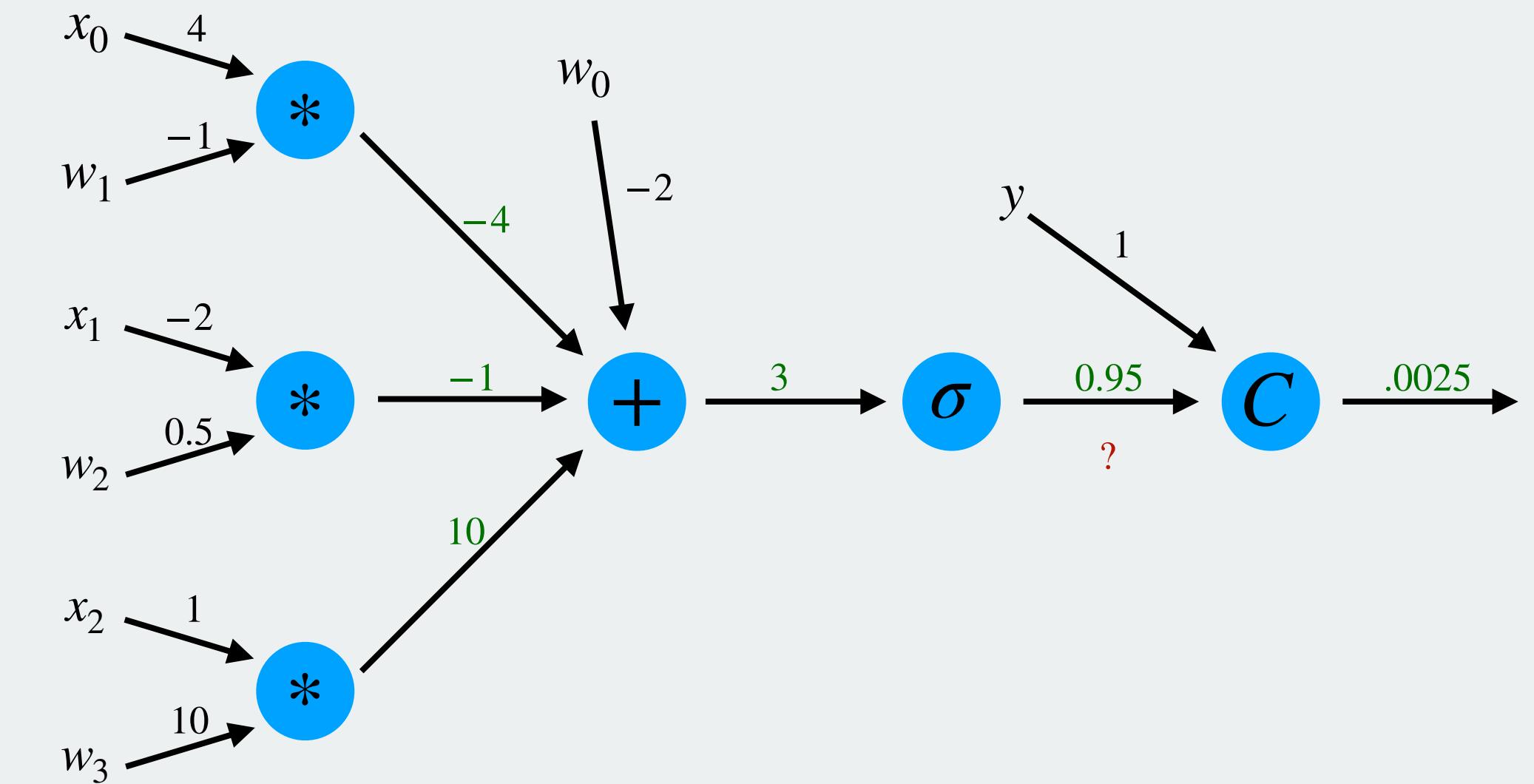
$$C(\tilde{y}, y) = (\tilde{y} - y)^2$$

Model derivatives:

$$C'(\tilde{y}, y) = 2(\tilde{y} - y)$$

As a computational graph:

Backward pass



Weights:

$$\mathbf{b} = [-2]$$

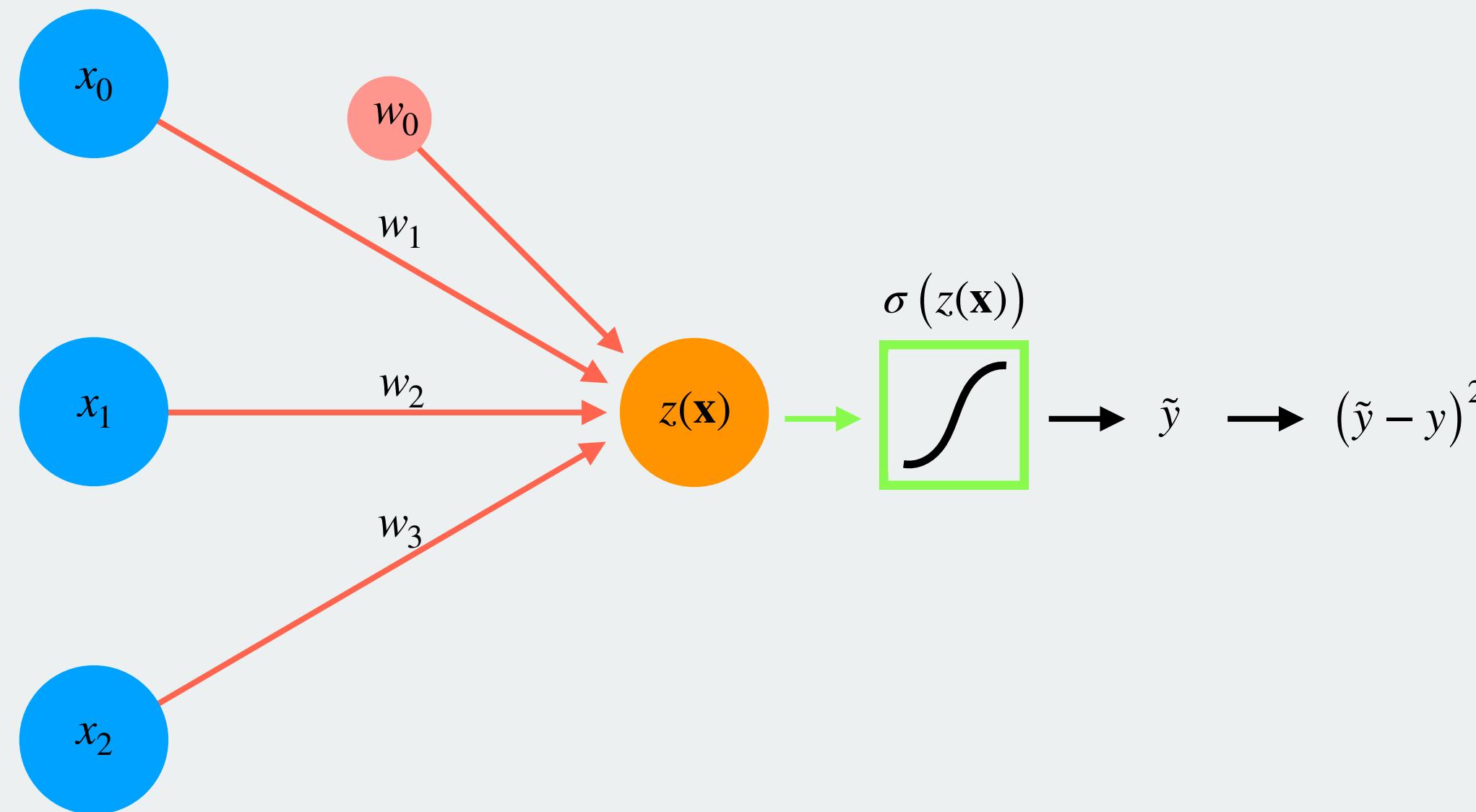
$$\mathbf{W} = [-1 \ 0.5 \ 10]$$

Data:

$$\mathbf{x} = \begin{bmatrix} 4 \\ -2 \\ 1 \end{bmatrix}$$

$$\mathbf{y} = \begin{bmatrix} 1 \end{bmatrix}$$

Backpropagation – on a neural network



Model:

$$w_0 + x_0 w_1 + x_1 w_2 + x_2 w_3 = z(\mathbf{x})$$

$$\sigma(x) = \frac{1}{1 + \exp(-x)}$$

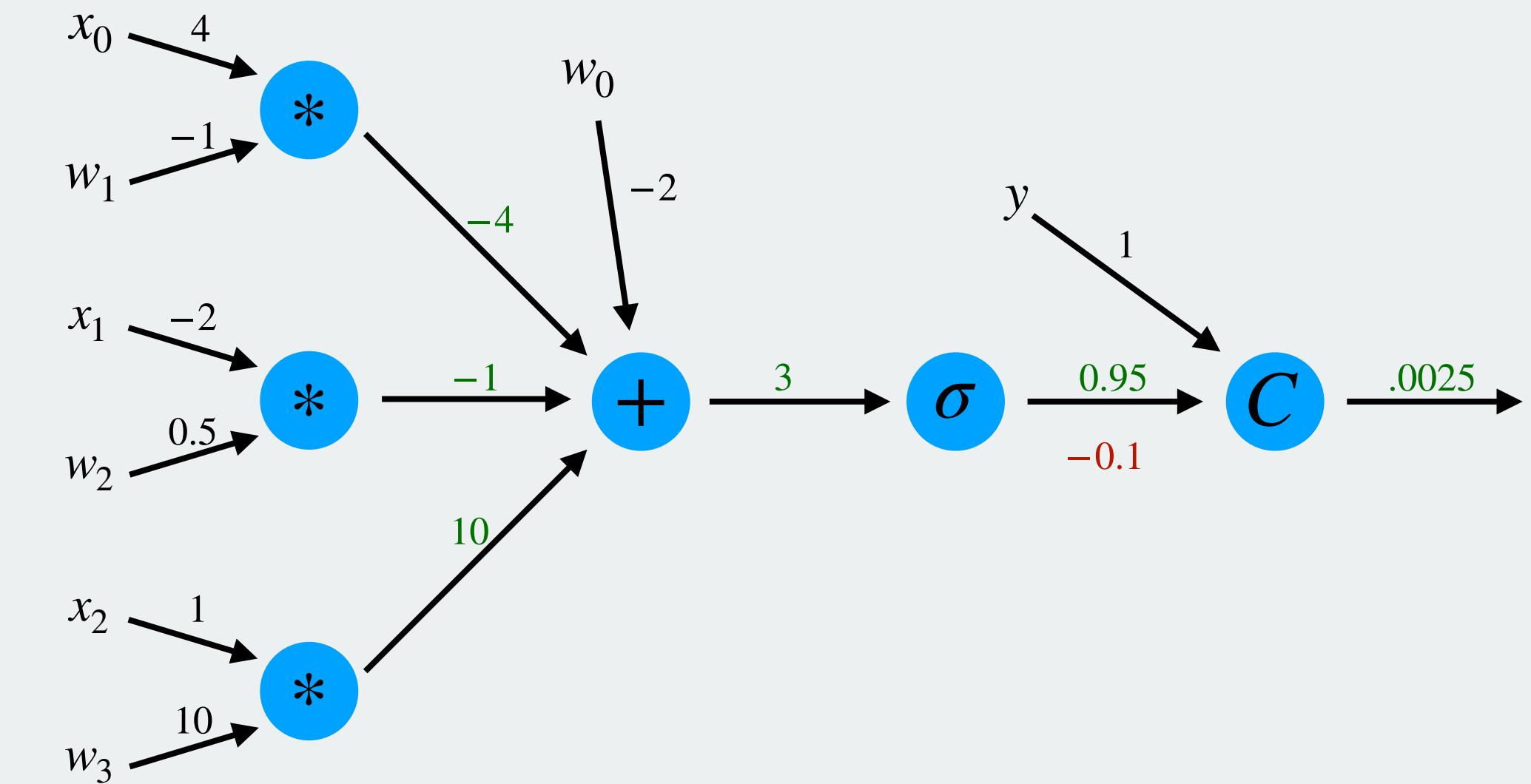
$$C(\tilde{y}, y) = (\tilde{y} - y)^2$$

Model derivatives:

$$C'(\tilde{y}, y) = 2(\tilde{y} - y)$$

As a computational graph:

Backward pass



Weights:

$$\mathbf{b} = [-2]$$

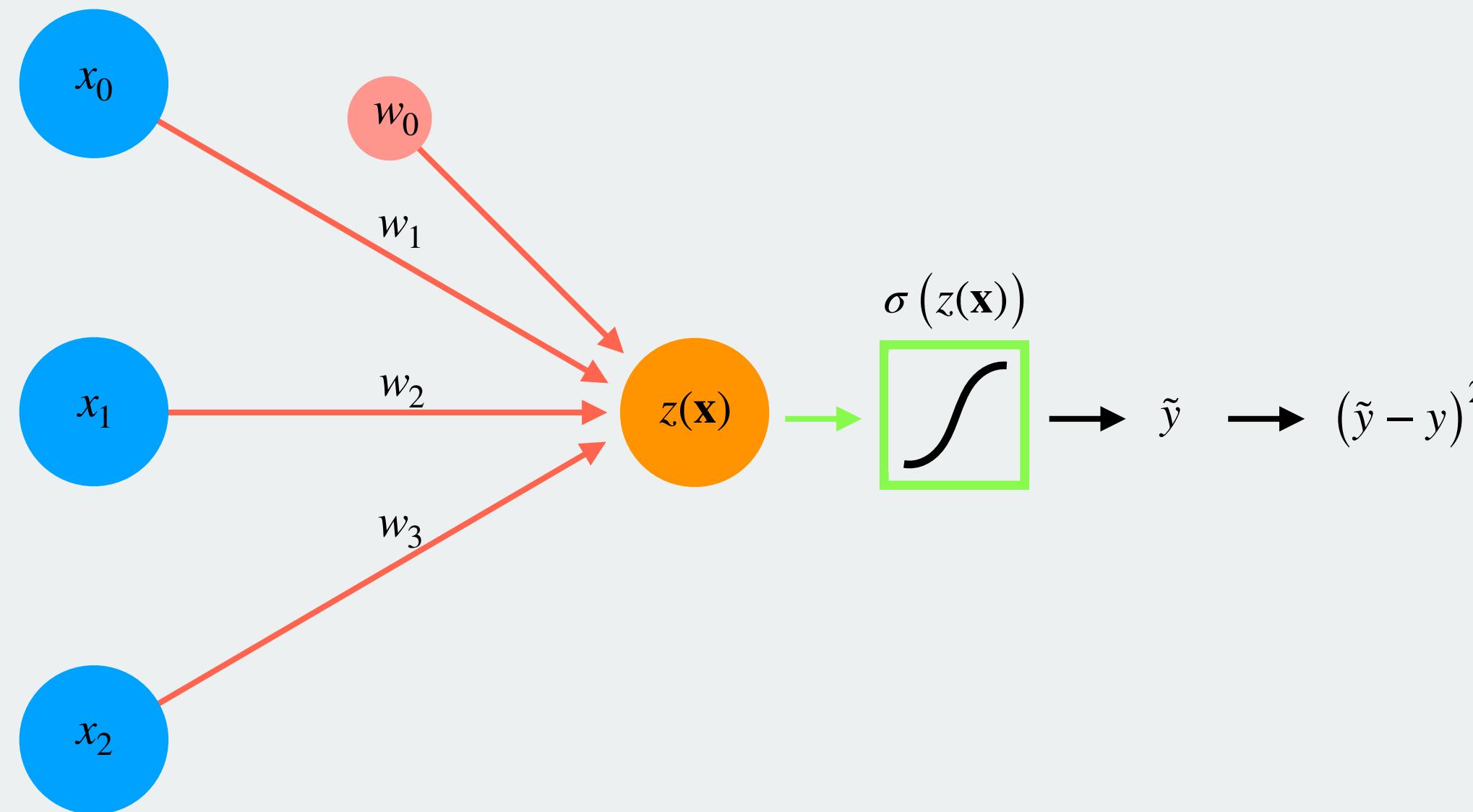
$$\mathbf{W} = [-1 \ 0.5 \ 10]$$

Data:

$$\mathbf{x} = \begin{bmatrix} 4 \\ -2 \\ 1 \end{bmatrix}$$

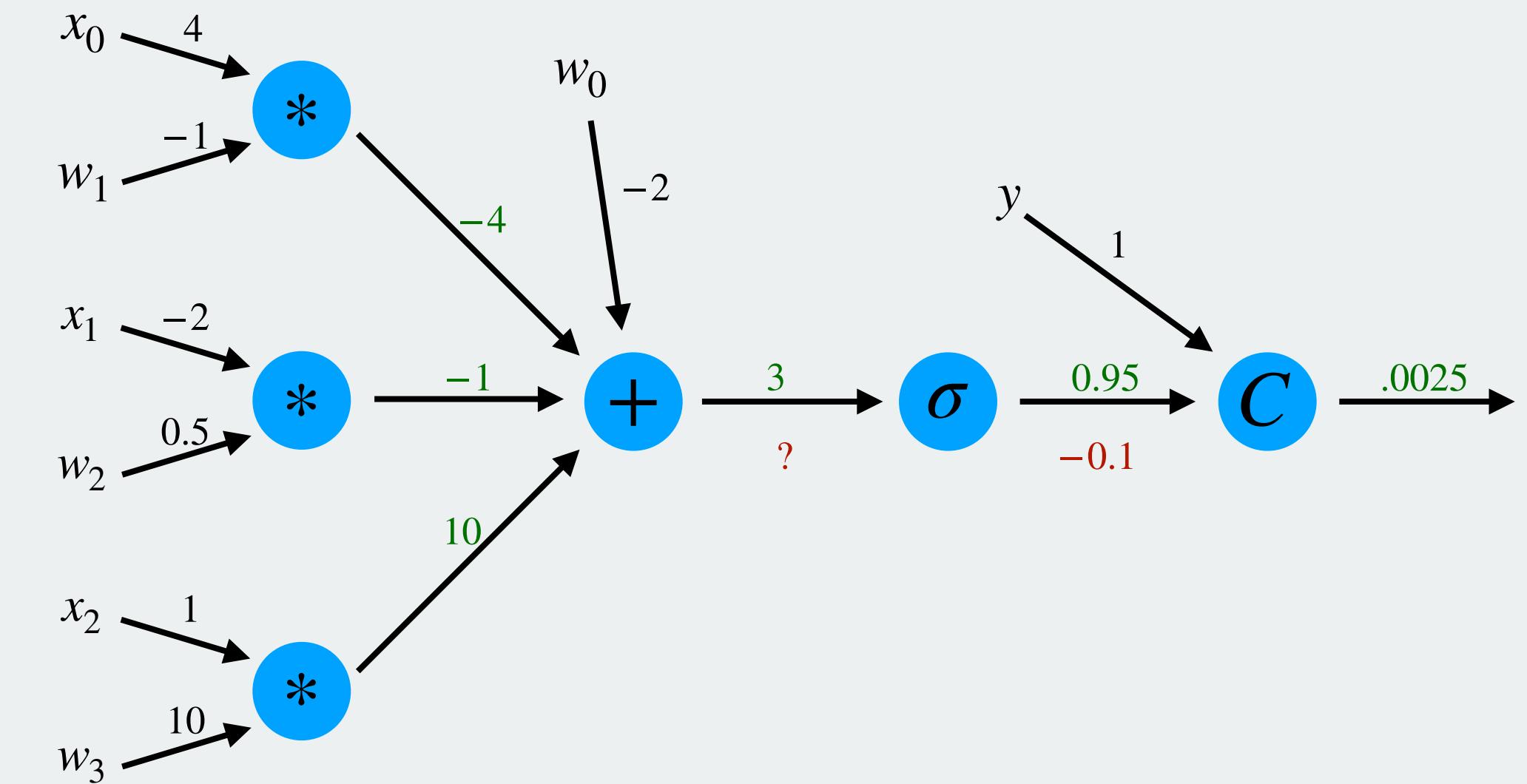
$$\mathbf{y} = \begin{bmatrix} 1 \end{bmatrix}$$

Backpropagation – on a neural network



As a computational graph:

Backward pass



Model:

$$w_0 + x_0w_1 + x_1w_2 + x_2w_3 = z(\mathbf{x})$$

$$\sigma(x) = \frac{1}{1 + \exp(-x)}$$

$$C(\tilde{y}, y) = (\tilde{y} - y)^2$$

Model derivatives:

$$\sigma'(x) = \sigma(x)(1 - \sigma(x))$$

$$C'(\tilde{y}, y) = 2(\tilde{y} - y)$$

Weights:

$$\mathbf{b} = [-2]$$

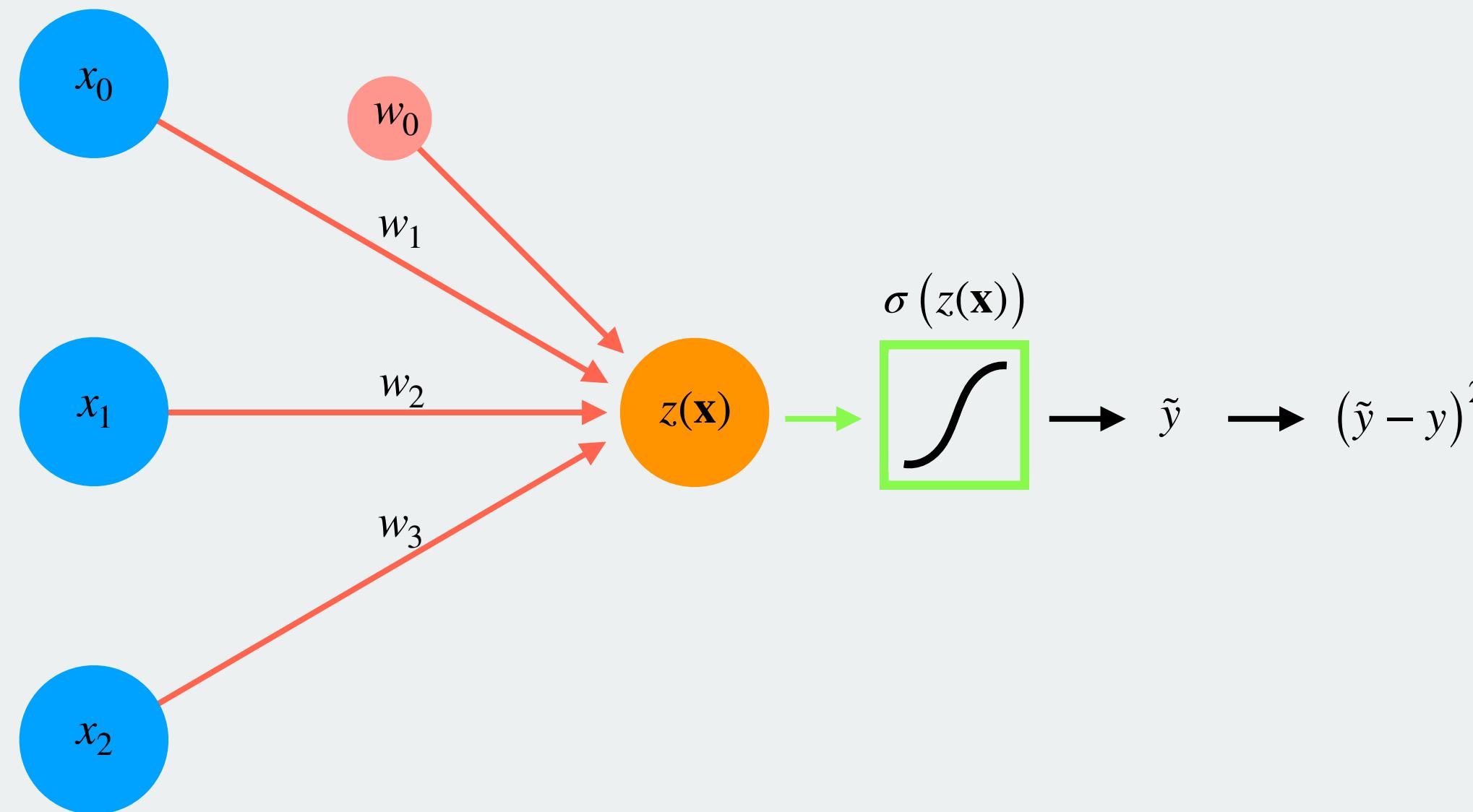
$$\mathbf{W} = [-1 \ 0.5 \ 10]$$

Data:

$$\mathbf{x} = \begin{bmatrix} 4 \\ -2 \\ 1 \end{bmatrix}$$

$$\mathbf{y} = \begin{bmatrix} 1 \end{bmatrix}$$

Backpropagation – on a neural network



Model:

$$w_0 + x_0 w_1 + x_1 w_2 + x_2 w_3 = z(\mathbf{x})$$

$$\sigma(x) = \frac{1}{1 + \exp(-x)}$$

$$C(\tilde{y}, y) = (\tilde{y} - y)^2$$

Model derivatives:

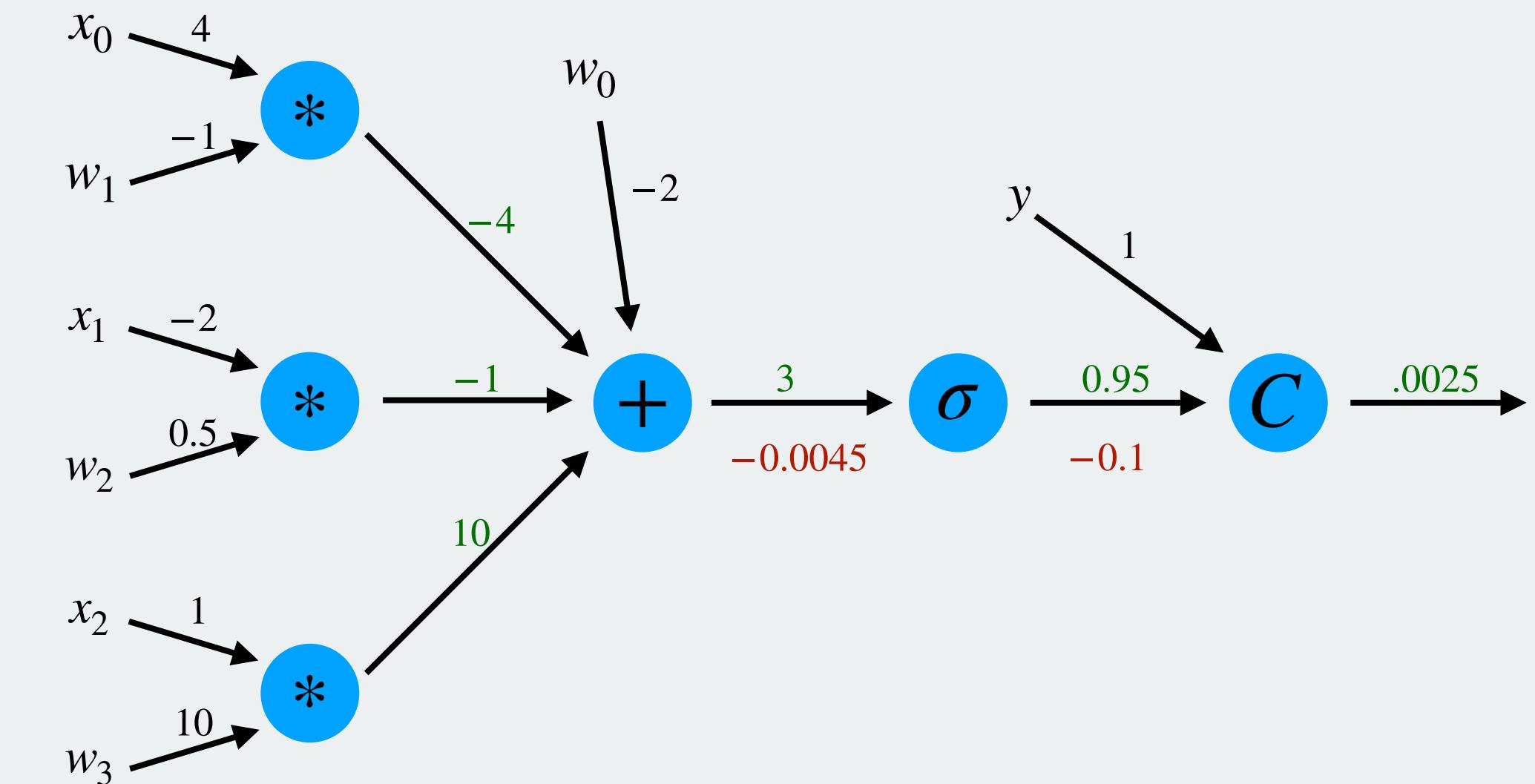
The + gate is like a function: $y = \sum_{i=0}^N z_n$

$$\sigma'(x) = \sigma(x)(1 - \sigma(x))$$

$$C'(\tilde{y}, y) = 2(\tilde{y} - y)$$

As a computational graph:

Backward pass



Weights:

$$\mathbf{b} = [-2]$$

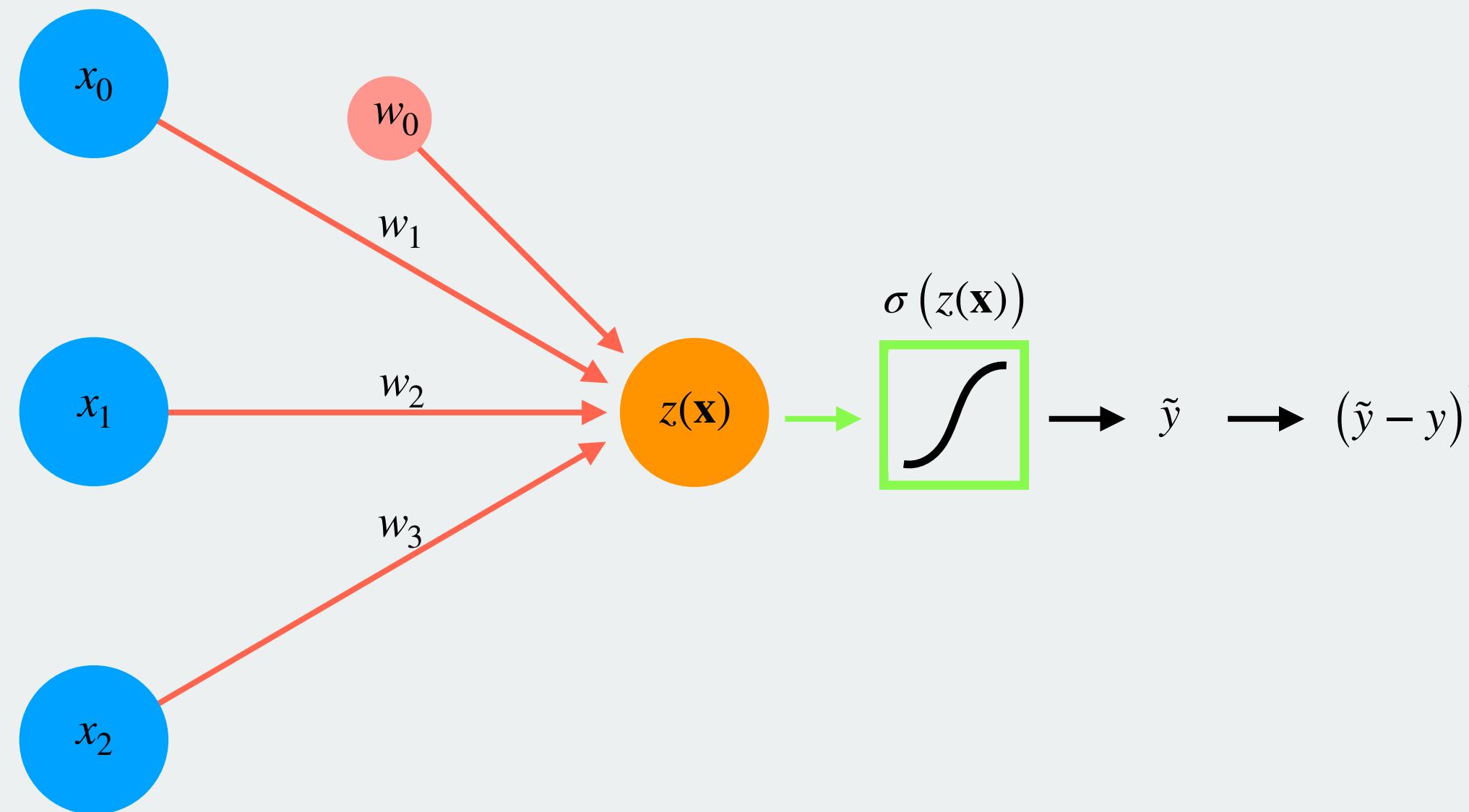
$$\mathbf{W} = [-1 \ 0.5 \ 10]$$

Data:

$$\mathbf{x} = \begin{bmatrix} 4 \\ -2 \\ 1 \end{bmatrix}$$

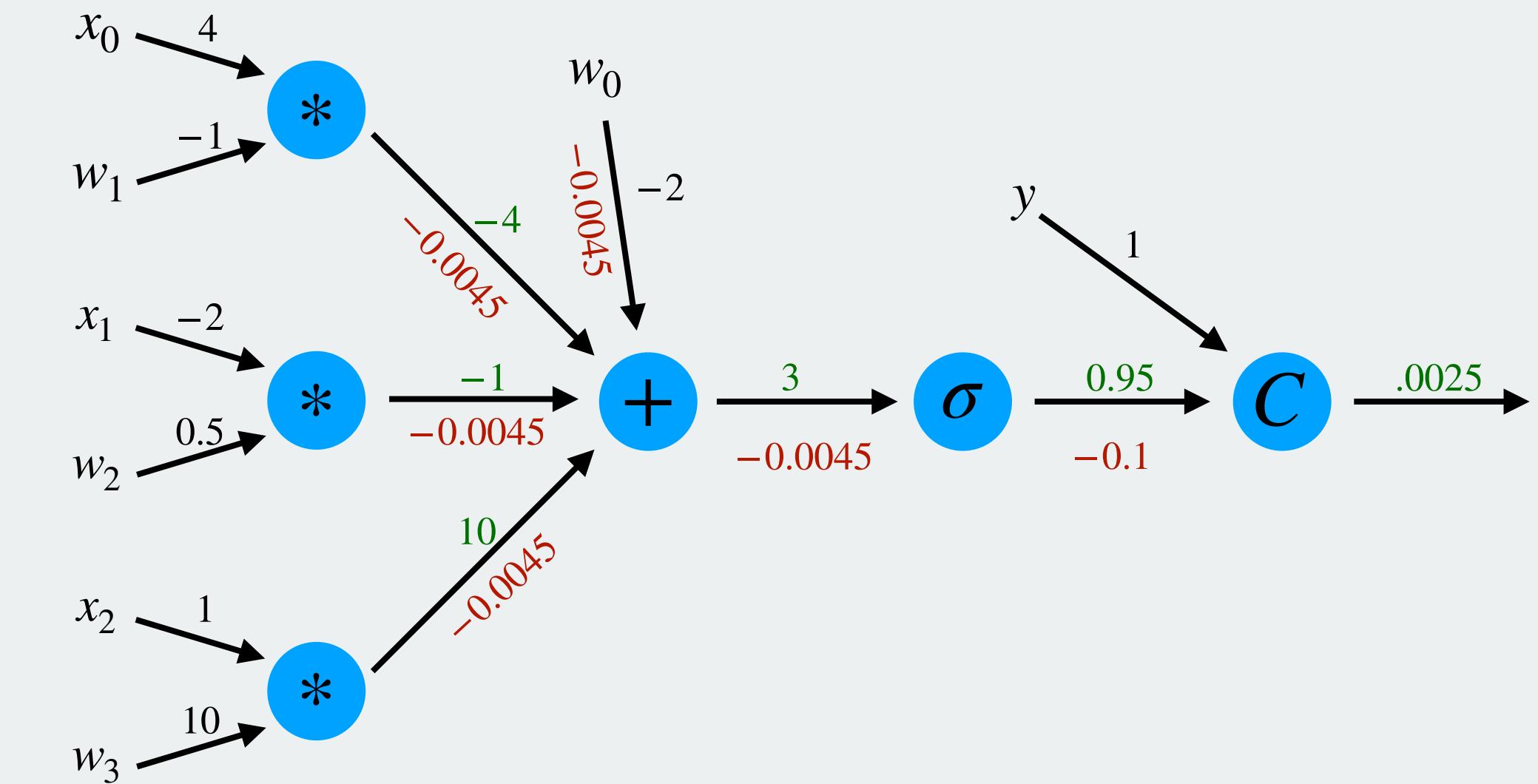
$$\mathbf{y} = [1]$$

Backpropagation – on a neural network



As a computational graph:

Backward pass



Model:

$$w_0 + x_0 w_1 + x_1 w_2 + x_2 w_3 = z(\mathbf{x})$$

$$\sigma(x) = \frac{1}{1 + \exp(-x)}$$

$$C(\tilde{y}, y) = (\tilde{y} - y)^2$$

Model derivatives:

The + gate is like a function:

$$\sigma'(x) = \sigma(x)(1 - \sigma(x))$$

$$C'(\tilde{y}, y) = 2(\tilde{y} - y)$$

Weights:

$$\mathbf{b} = \begin{bmatrix} -2 \end{bmatrix}$$

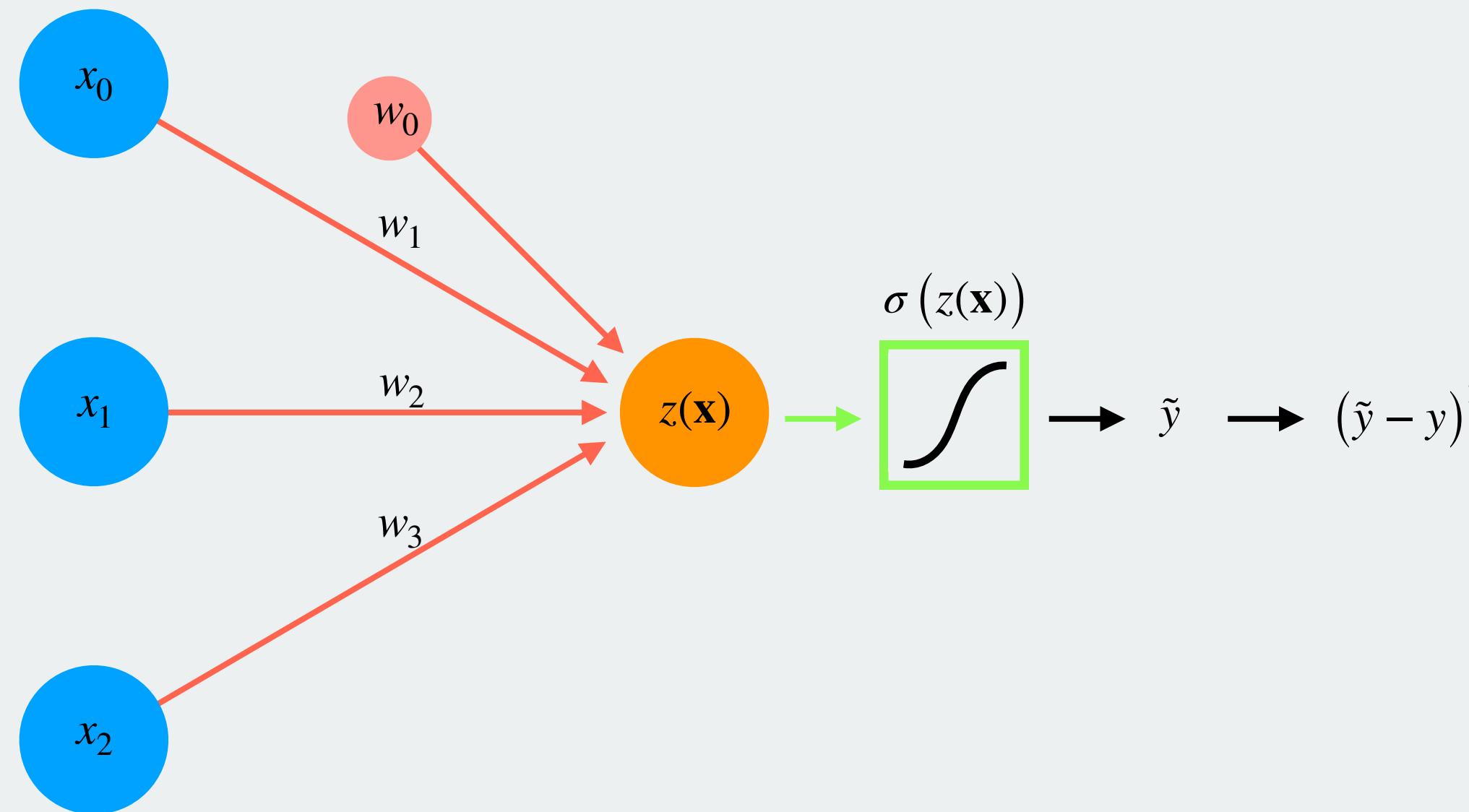
$$\mathbf{W} = \begin{bmatrix} -1 & 0.5 & 10 \end{bmatrix}$$

Data:

$$\mathbf{x} = \begin{bmatrix} 4 \\ -2 \\ 1 \end{bmatrix}$$

$$\mathbf{y} = \begin{bmatrix} 1 \end{bmatrix}$$

Backpropagation – on a neural network



Model:

$$w_0 + x_0w_1 + x_1w_2 + x_2w_3 = z(\mathbf{x})$$

$$\sigma(x) = \frac{1}{1 + \exp(-x)}$$

$$C(\tilde{y}, y) = (\tilde{y} - y)^2$$

Model derivatives:

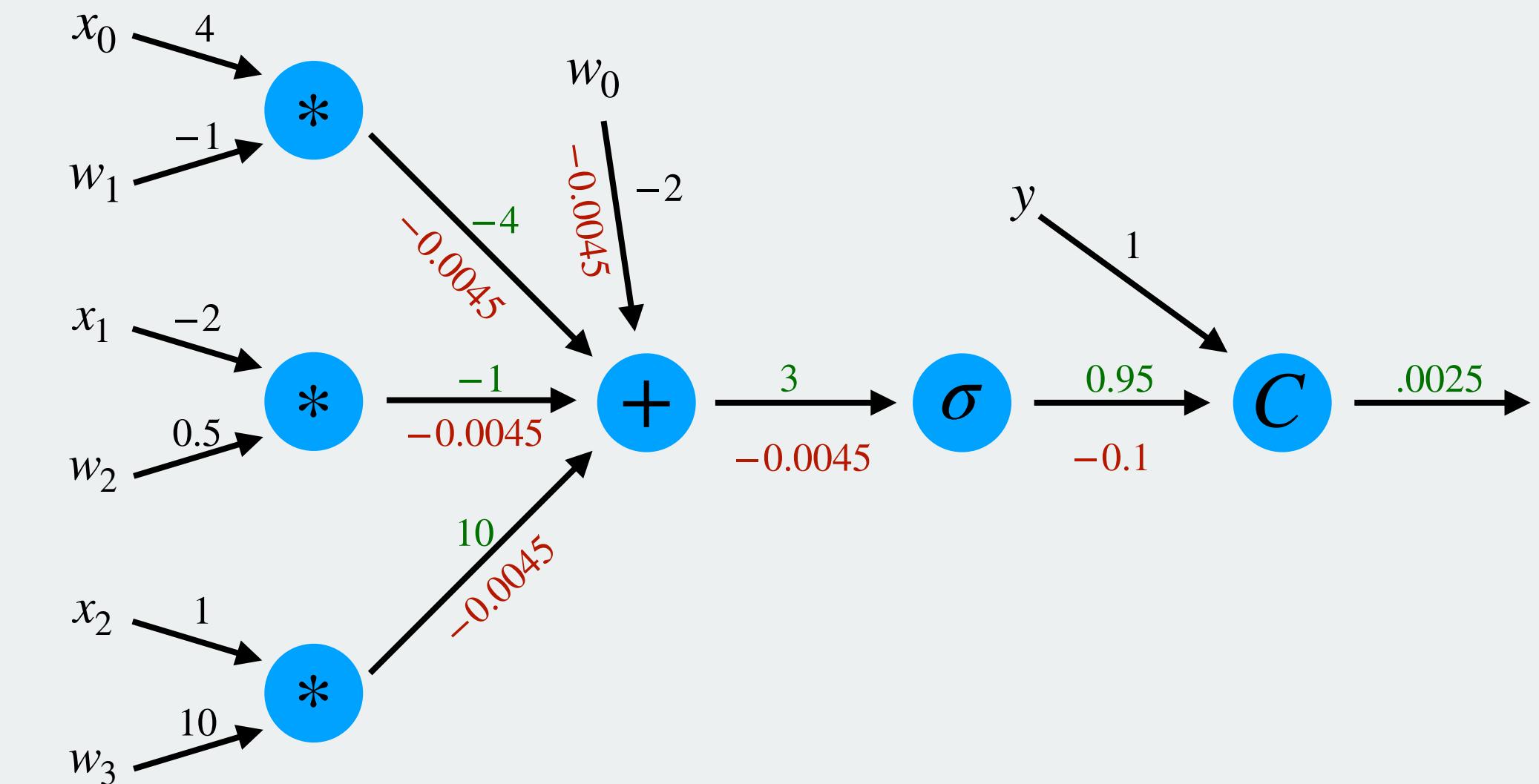
Each branch is a function:
 $y = wx$

$$\sigma'(x) = \sigma(x)(1 - \sigma(x))$$

$$C'(\tilde{y}, y) = 2(\tilde{y} - y)$$

As a computational graph:

Backward pass



Weights:

$$\mathbf{b} = \begin{bmatrix} -2 \end{bmatrix}$$

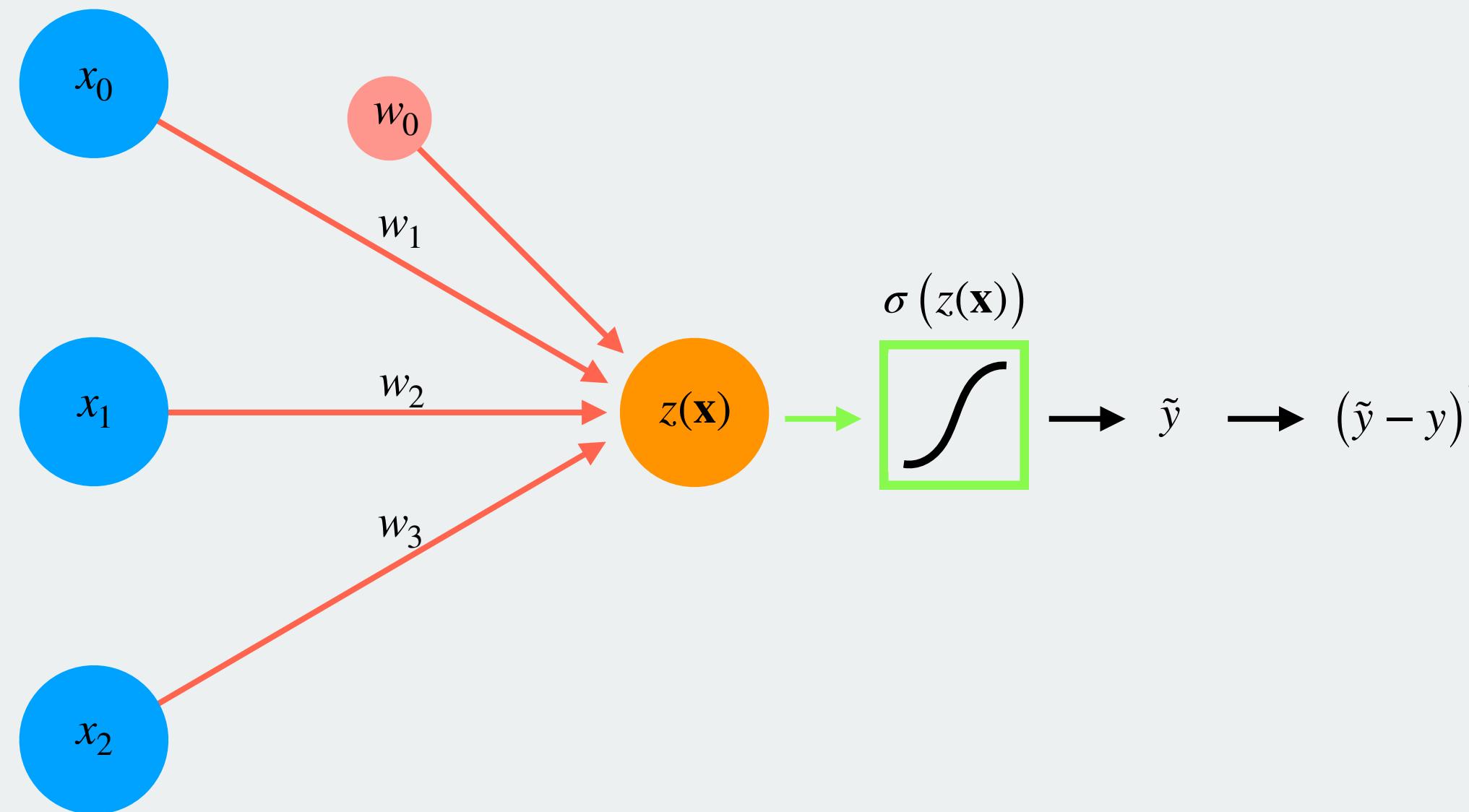
$$\mathbf{W} = \begin{bmatrix} -1 & 0.5 & 10 \end{bmatrix}$$

Data:

$$\mathbf{x} = \begin{bmatrix} 4 \\ -2 \\ 1 \end{bmatrix}$$

$$\mathbf{y} = \begin{bmatrix} 1 \end{bmatrix}$$

Backpropagation – on a neural network



Model:

$$w_0 + x_0 w_1 + x_1 w_2 + x_2 w_3 = z(\mathbf{x})$$

$$\sigma(x) = \frac{1}{1 + \exp(-x)}$$

$$C(\tilde{y}, y) = (\tilde{y} - y)^2$$

Model derivatives:

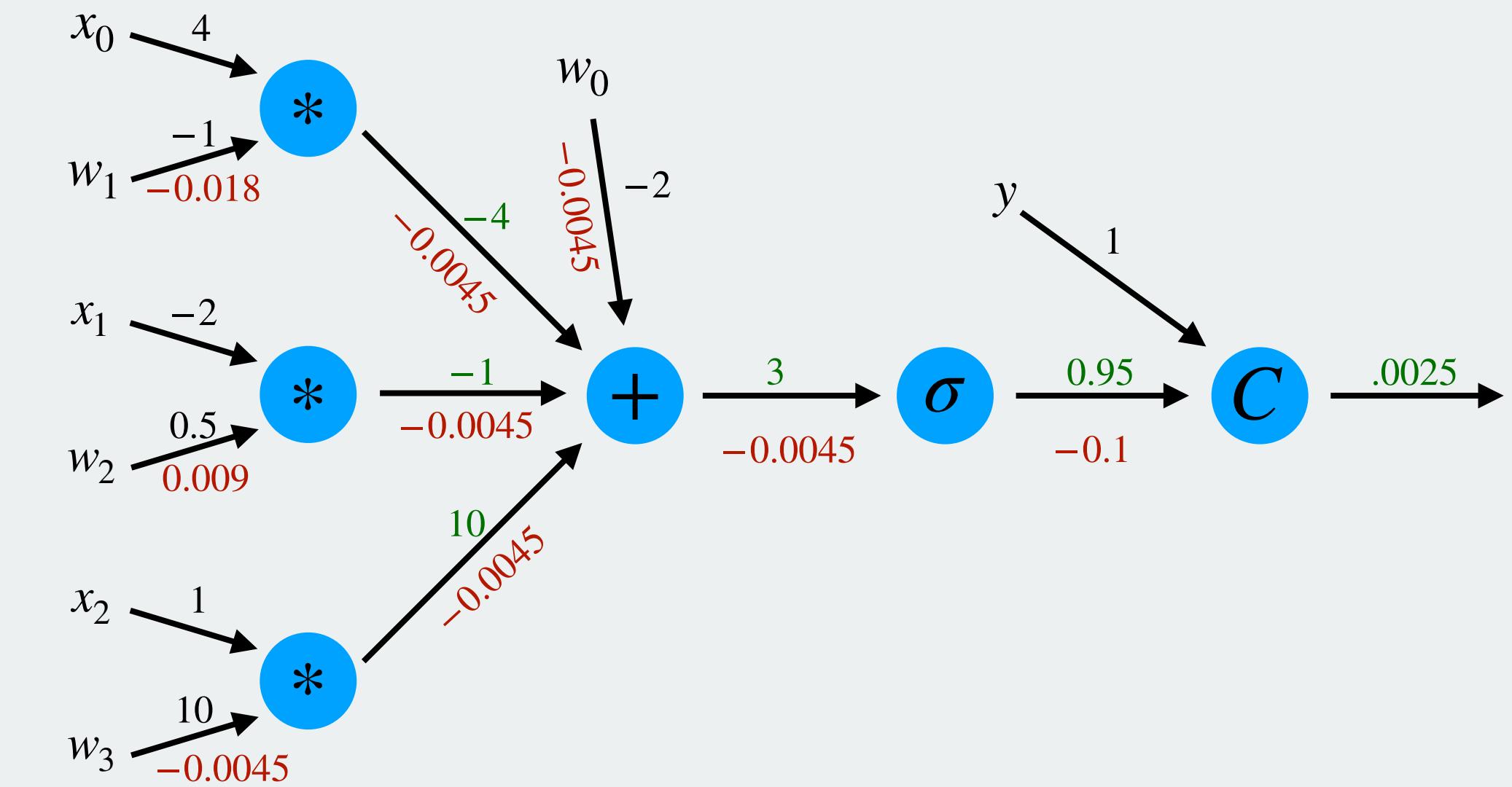
Each branch is a function:
 $y = wx$

$$\sigma'(x) = \sigma(x)(1 - \sigma(x))$$

$$C'(\tilde{y}, y) = 2(\tilde{y} - y)$$

As a computational graph:

Backward pass



Weights:

$$\mathbf{b} = \begin{bmatrix} -2 \end{bmatrix}$$

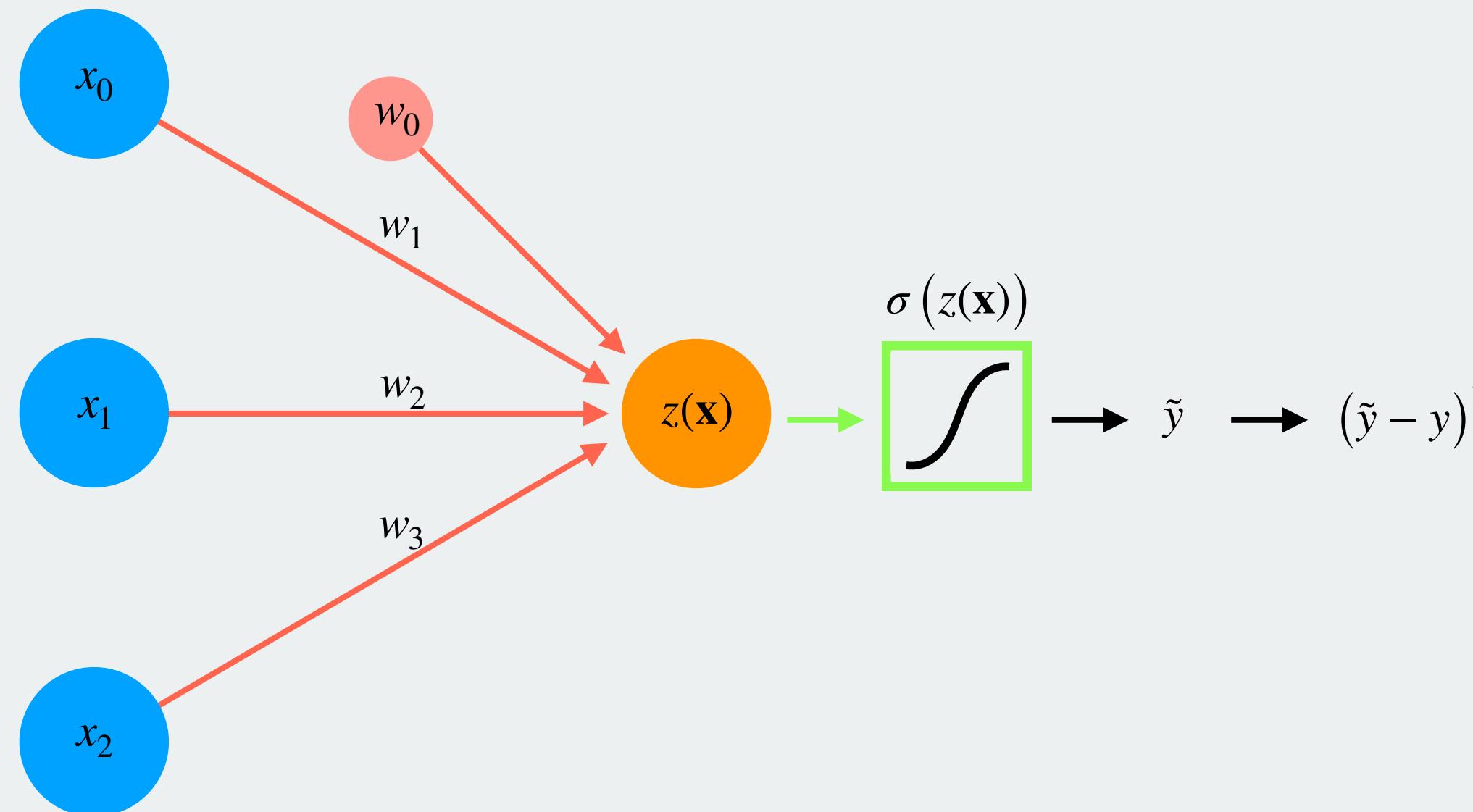
$$\mathbf{W} = \begin{bmatrix} -1 & 0.5 & 10 \end{bmatrix}$$

Data:

$$\mathbf{x} = \begin{bmatrix} 4 \\ -2 \\ 1 \end{bmatrix}$$

$$\mathbf{y} = \begin{bmatrix} 1 \end{bmatrix}$$

Backpropagation – on a neural network



Model:

$$w_0 + x_0 w_1 + x_1 w_2 + x_2 w_3 = z(\mathbf{x})$$

$$\sigma(x) = \frac{1}{1 + \exp(-x)}$$

$$C(\tilde{y}, y) = (\tilde{y} - y)^2$$

Model derivatives:

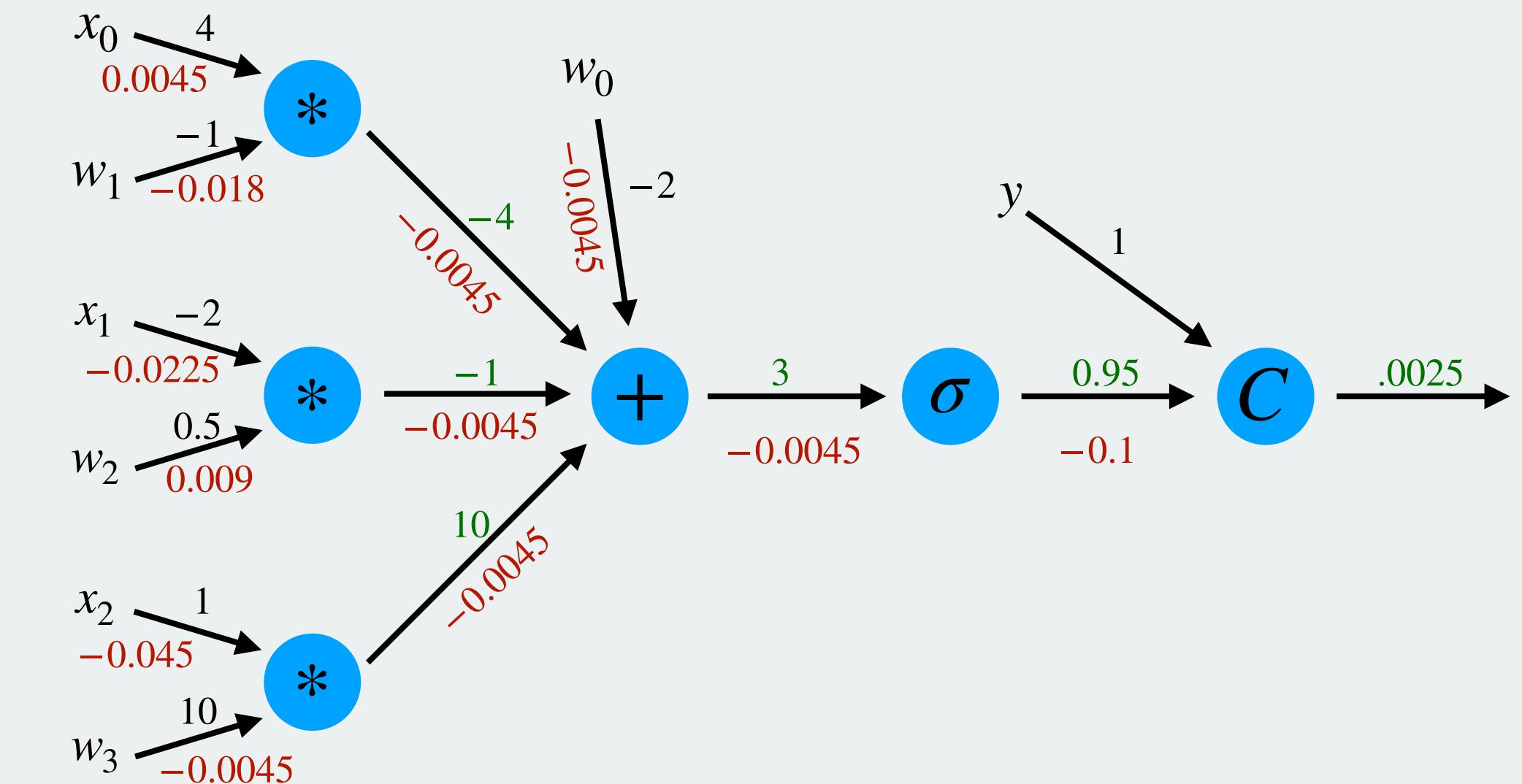
Each branch is a function:
 $y = wx$

$$\sigma'(x) = \sigma(x)(1 - \sigma(x))$$

$$C'(\tilde{y}, y) = 2(\tilde{y} - y)$$

As a computational graph:

Backward pass



Weights:

$$\mathbf{b} = \begin{bmatrix} -2 \end{bmatrix}$$

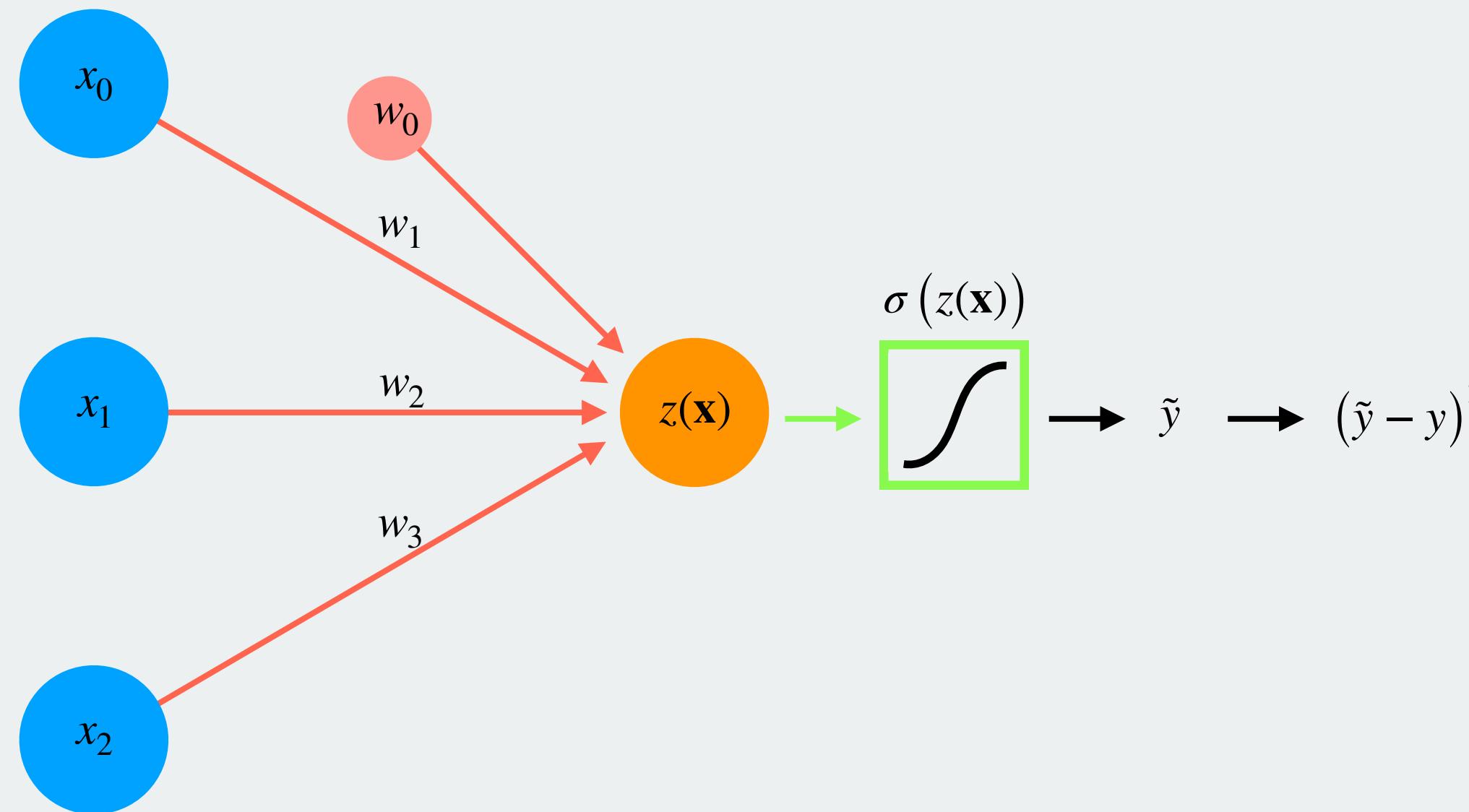
$$\mathbf{W} = \begin{bmatrix} -1 & 0.5 & 10 \end{bmatrix}$$

Data:

$$\mathbf{x} = \begin{bmatrix} 4 \\ -2 \\ 1 \end{bmatrix}$$

$$\mathbf{y} = \begin{bmatrix} 1 \end{bmatrix}$$

Backpropagation – on a neural network



Model derivatives:

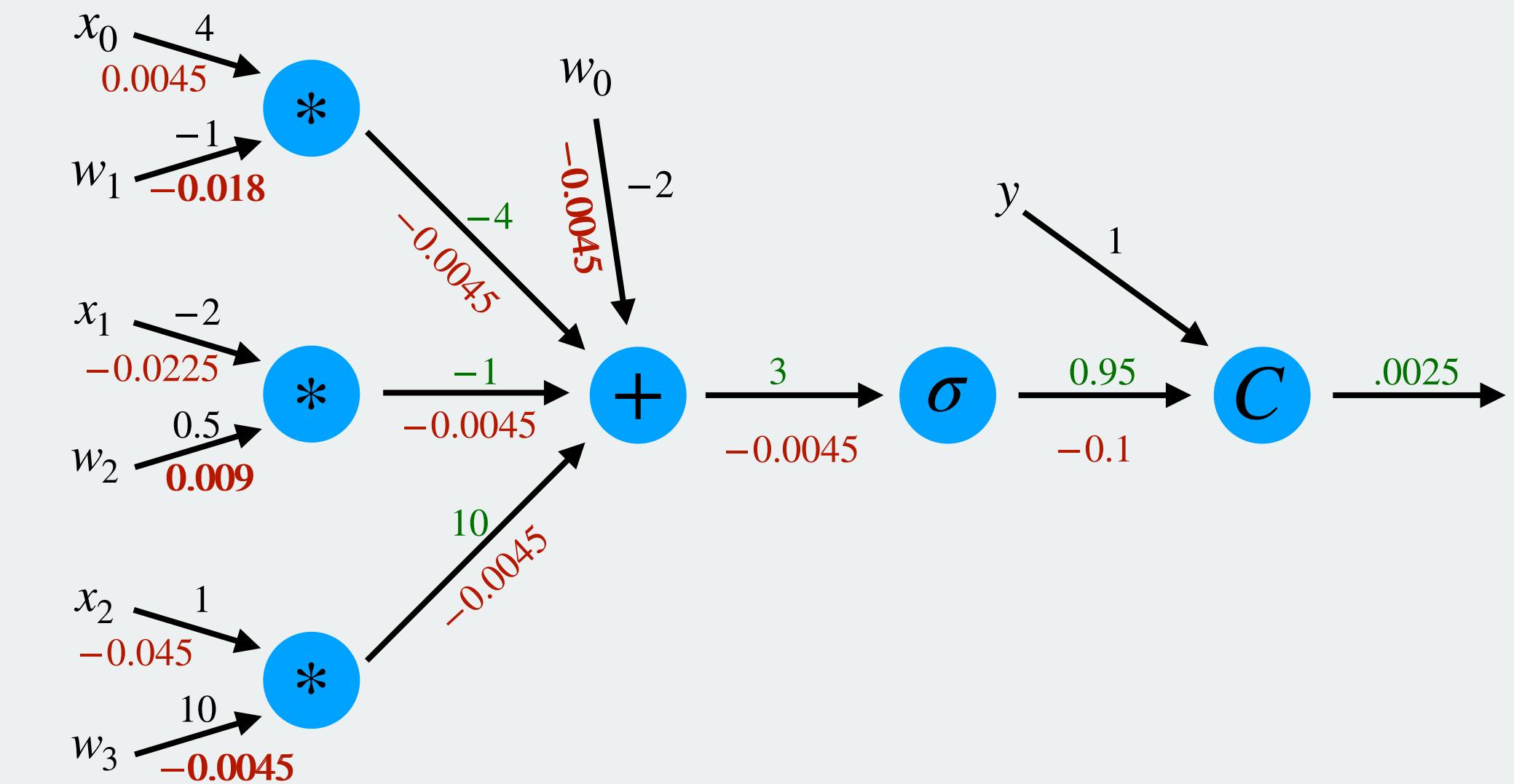
Each branch is a function:
 $y = wx$

$$\sigma'(x) = \sigma(x)(1 - \sigma(x))$$

$$C'(\tilde{y}, y) = 2(\tilde{y} - y)$$

As a computational graph:

Backward pass



Weights:

$$\mathbf{b} = \begin{bmatrix} -2 \end{bmatrix}$$

$$\mathbf{W} = \begin{bmatrix} -1 & 0.5 & 10 \end{bmatrix}$$

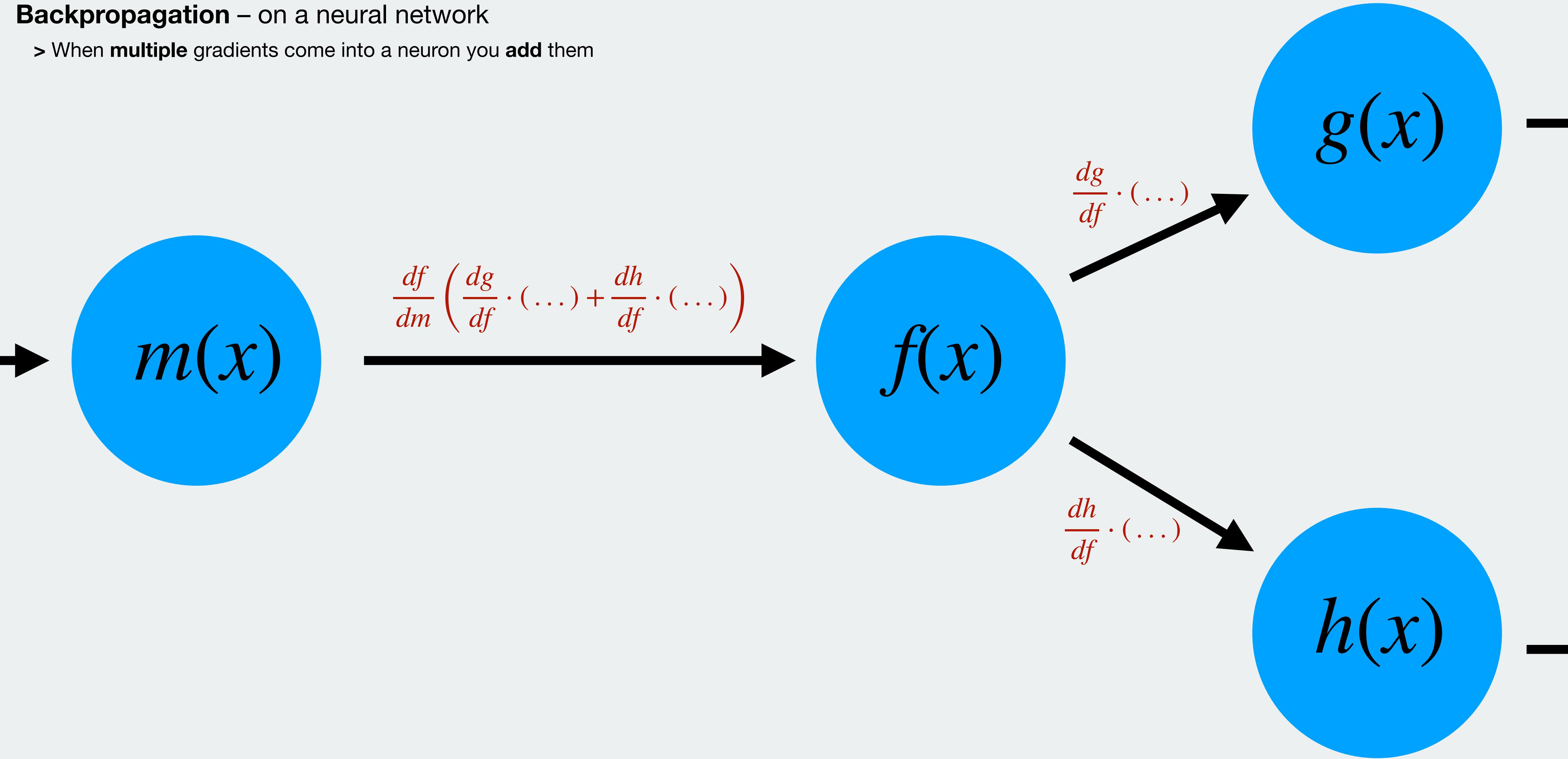
Data:

$$\mathbf{x} = \begin{bmatrix} 4 \\ -2 \\ 1 \end{bmatrix}$$

$$\mathbf{y} = \begin{bmatrix} 1 \end{bmatrix}$$

Backpropagation – on a neural network

> When **multiple** gradients come into a neuron you **add** them



Backpropagation – in the computer

	2	5	0	4	7	9	...	Average over all training data
w_0	-0.08	+0.02	-0.02	+0.11	-0.05	-0.14	...	→ -0.08
w_1	-0.11	+0.11	+0.07	+0.02	+0.09	+0.05	...	
w_2	-0.07	-0.04	-0.01	+0.02	+0.13	-0.15	...	
:	:	:	:	:	:	:	⋮	
$w_{13,001}$	+0.13	+0.08	-0.06	-0.09	-0.02	+0.04	...	

<https://www.youtube.com/watch?v=llg3gGewQ5U>

Exercises:

- Understand and explain (the rest of) a pure Python code implementation of a neural network
- Hand calculate gradients in a small network using the backpropagation algorithm