Artificial Neural Networks and Deep Learning

Week 2

Gradient descent and backpropagation

RECAP

Logistic regression classifier

¡Logistic regression! classifier

(Input)

Variables

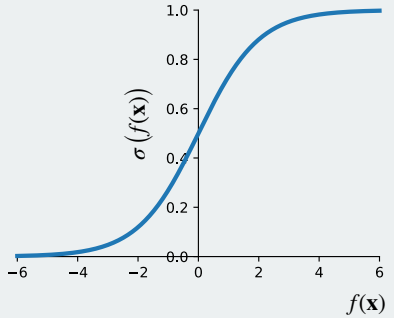
Weights

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

Output

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

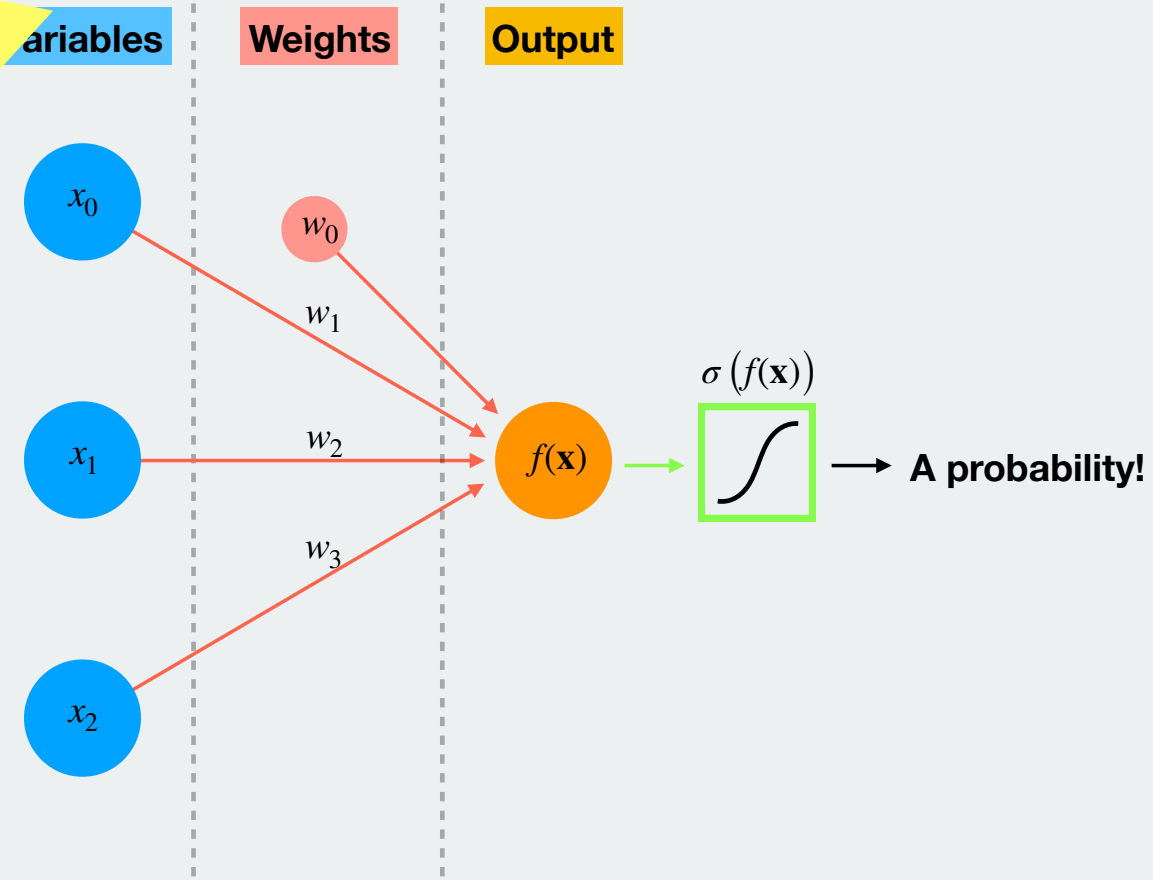
Activation function



Logistic reg chematic illustration

¡Perceptron!

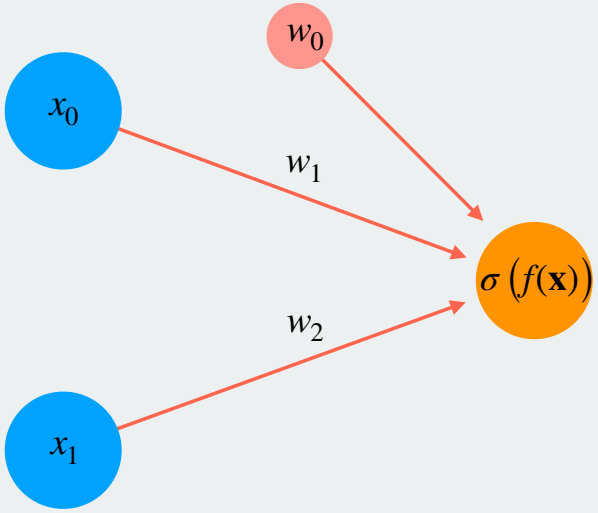
Invented by **Frank Rosenblat** in 1957

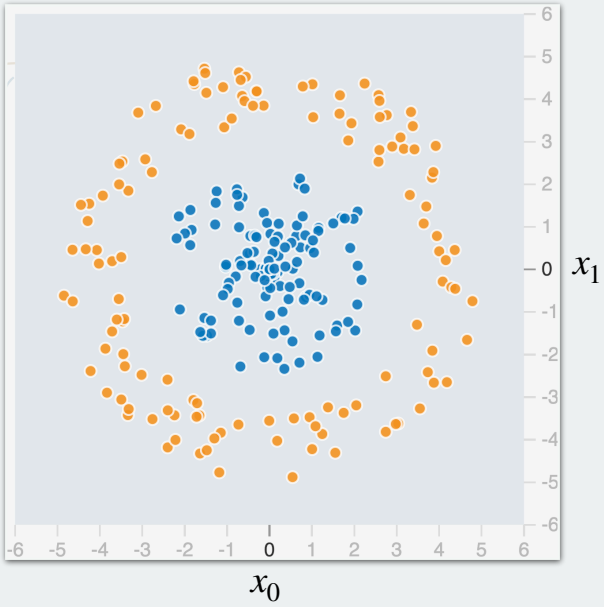


Logistic regression not so simple problem

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

Model too simple

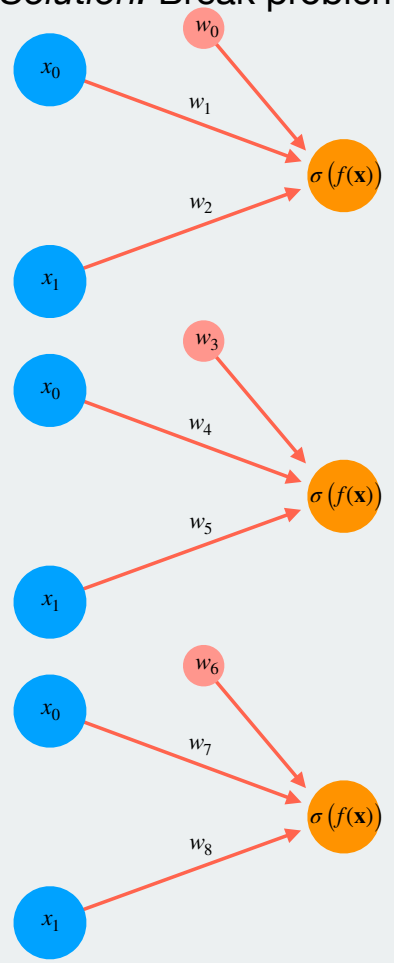




Reminder: $w_0 + x_0 w_1 + x_1 w_2 = f(\mathbf{x})$

Logistic regression not so simple problem

> *Solution:* Break problem into subproblems



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

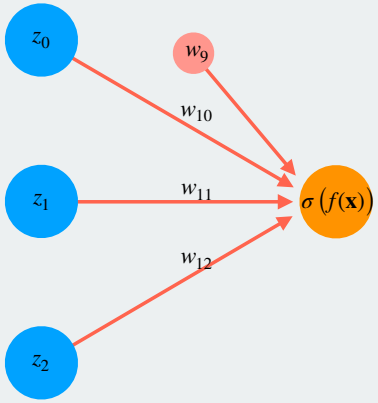
$$\sigma(w_3 + x_0 w_4 + x_1 w_5) = z_1(\mathbf{x})$$

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

New problem: Given **Z**, predict **y**

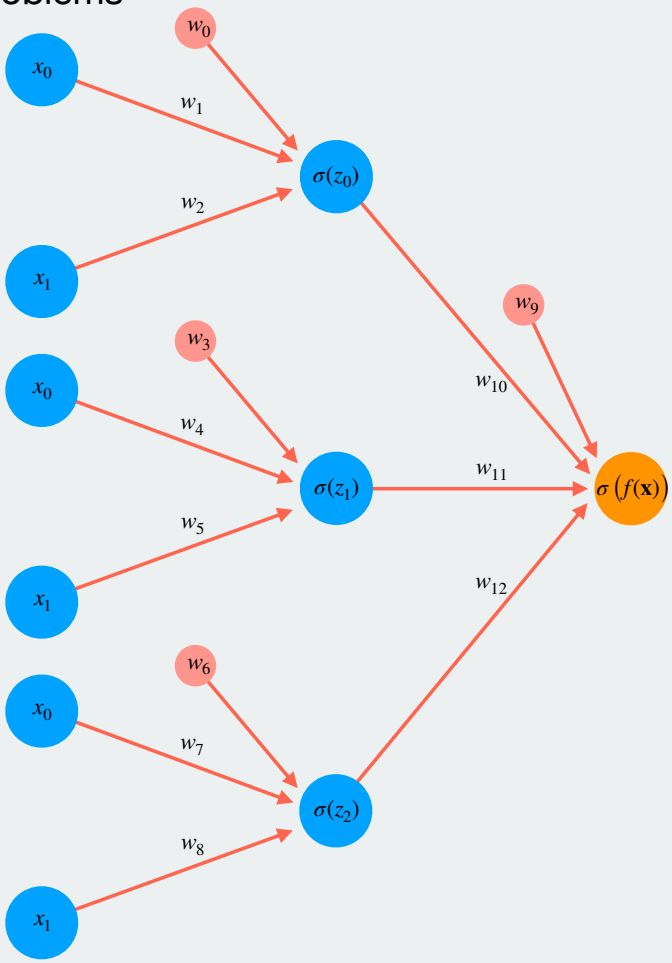
$$\mathbf{Z} = \begin{array}{|c|c|c|} \hline z_0 & z_1 & z_2 \\ \hline 0{,}3 & 0{,}75 & 0{,}78 \\ \hline 0{,}25 & 0{,}1 & 0{,}95 \\ \hline \ldots & \ldots & \\ \hline 0{,}79 & 0{,}99 & 0{,}3 \\ \hline 0{,}34 & 0{,}6 & 0{,}1 \\ \hline \end{array} \qquad \mathbf{y} = \begin{array}{|c|} \hline y \\ \hline 0 \\ \hline 1 \\ \hline \ldots \\ \hline 1 \\ \hline 0 \\ \hline \end{array}$$

Solution: Why not use a logistic regression?



Logistic regression not so simple problem

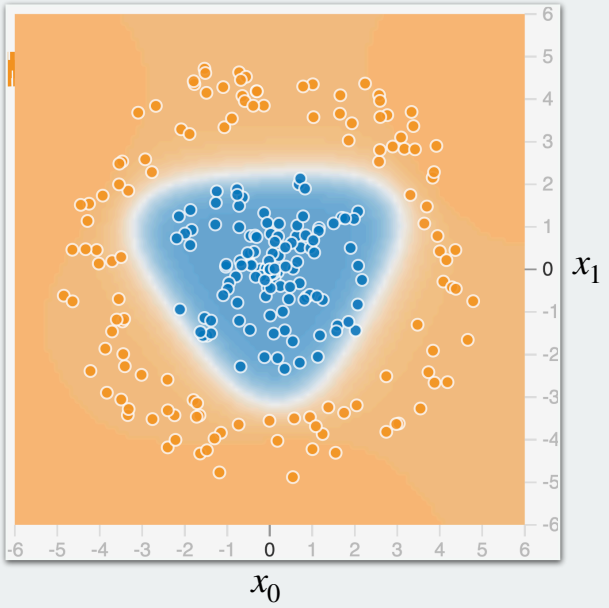
> *Solution:* Connect subproblems



Mathematical form

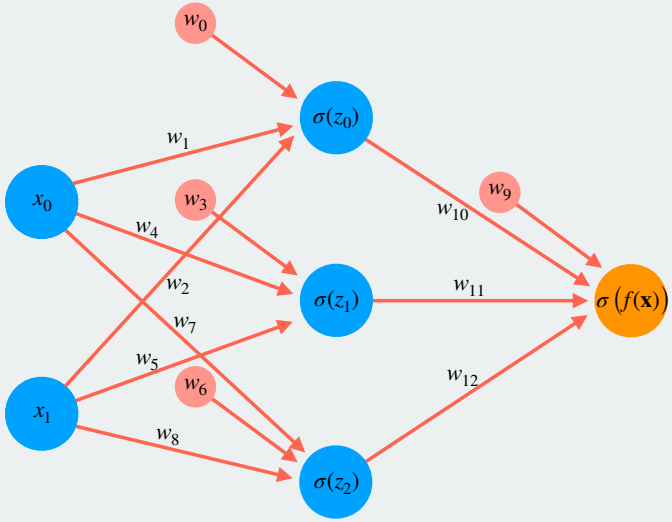
$$\sigma\Big(w_9 + \sigma(w_0 + x_0 w_1 + x_1 w_2) w_{10} \quad + \\ \sigma(w_3 + x_0 w_4 + x_1 w_5) w_{11} \quad + \\ \sigma(w_6 + x_0 w_7 + x_1 w_8) w_{12}\Big) = \sigma\big(f(x)\big)$$

Solution



Logistic regression not so simple problem

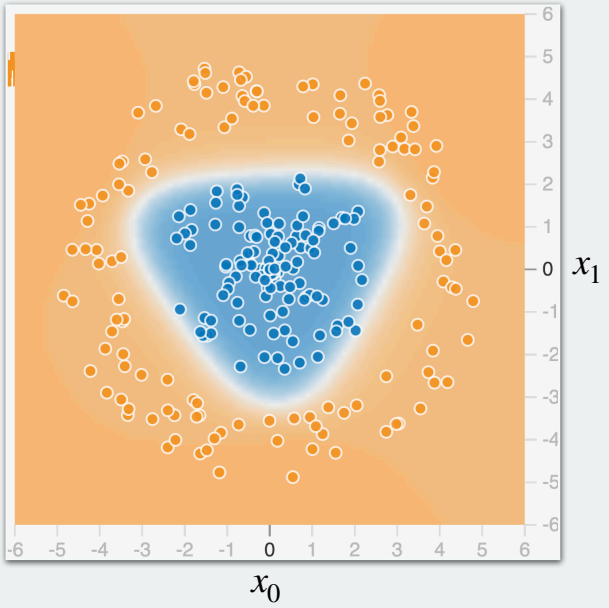
> *Solution:* Connect subproblems



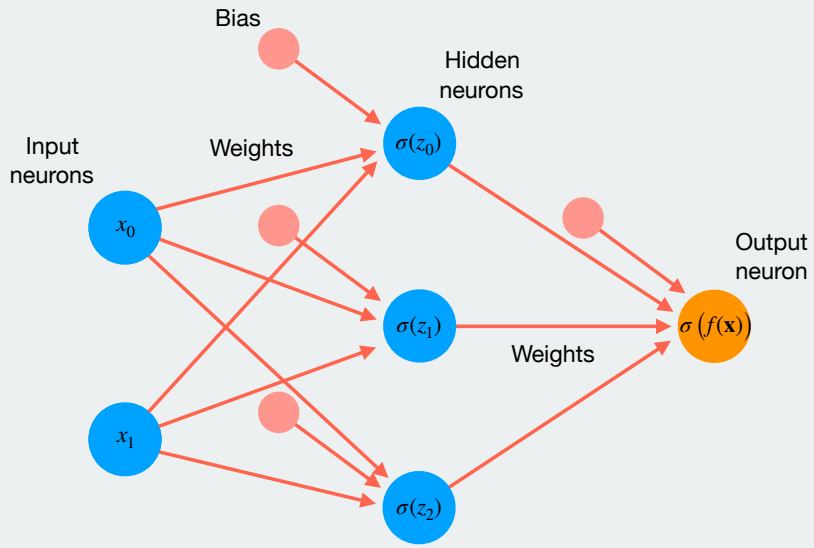
Mathematical form

$$\sigma\Big(w_9 + \sigma(w_0 + x_0 w_1 + x_1 w_2) w_{10} \quad + \\ \sigma(w_3 + x_0 w_4 + x_1 w_5) w_{11} \quad + \\ \sigma(w_6 + x_0 w_7 + x_1 w_8) w_{12}\Big) = \sigma\big(f(x)\big)$$

Solution



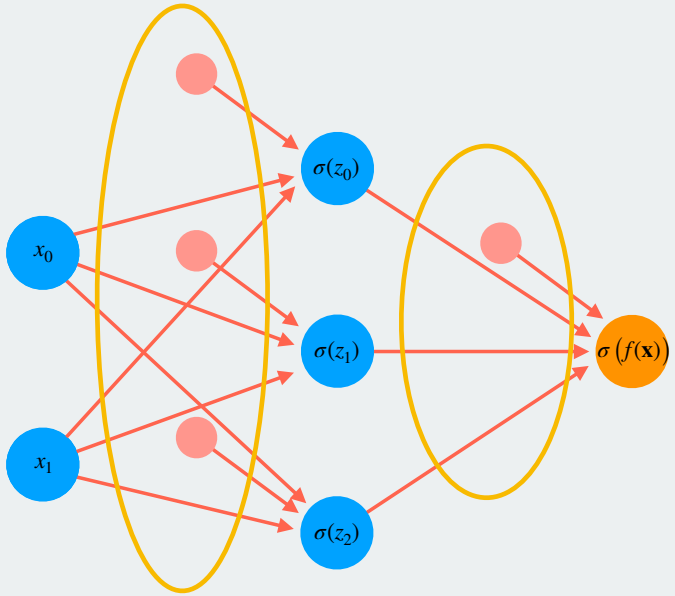
Multilayer perceptron – the structure



$$\sigma\left(\mathbf{b_n} + \mathbf{W_n a_{n-1}}\right) = \mathbf{a_n}$$

TODAY: How neural networks learn

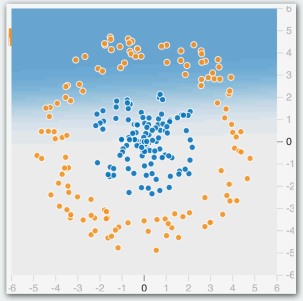
Specifically: how to **set weights** so a network makes **good predictions**

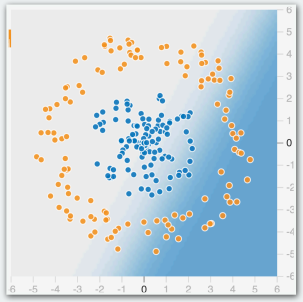


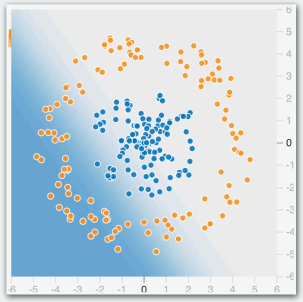
STEPS:

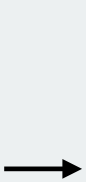
1. Gradient descent
2. Backpropagation

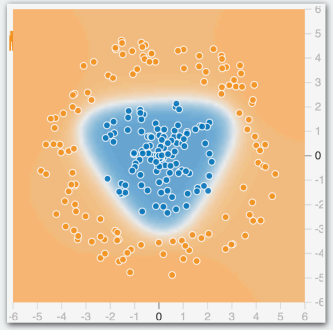
How do we find the weights that give the best classification?





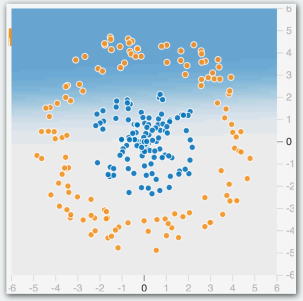


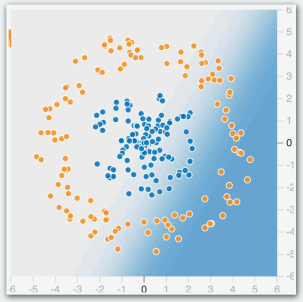


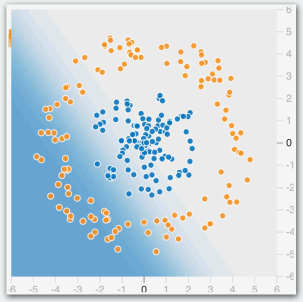


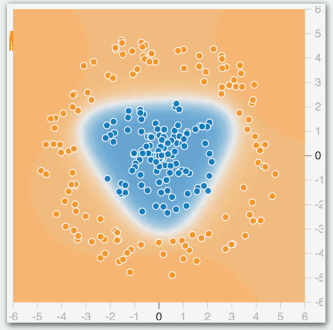
example

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









predicted from X given W

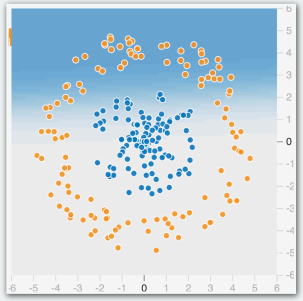
$$\tilde{\mathbf{y}} = \begin{bmatrix} 0.96 \\ 0.10 \\ 0.04 \\ \cdots \\ 0.70 \\ 0.02 \\ 0.99 \end{bmatrix}$$

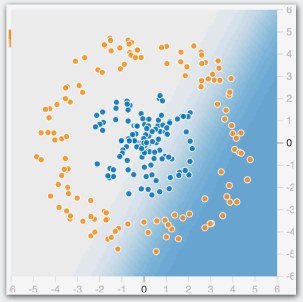
true

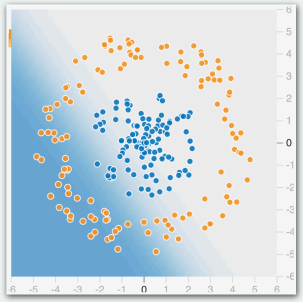
$$\mathbf{y} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \cdots \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

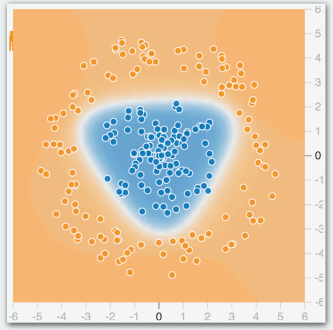
$$\begin{aligned} C(\mathbf{W}) &= \frac{1}{N} \sum_i (\tilde{y}_i - y_i)^2 \\ &= (0.96 - 1)^2 \\ &+ (0.10 - 0)^2 \\ &+ (0.04 - 0)^2 \\ &+ \ldots \\ &+ (0.70 - 1)^2 \\ &+ (0.02 - 0)^2 \\ &+ (0.99 - 1)^2 \end{aligned}$$

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









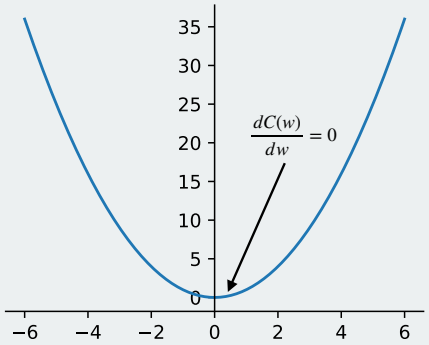
predicted from X given W

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

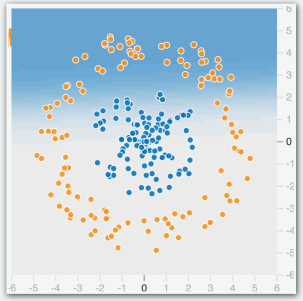
true

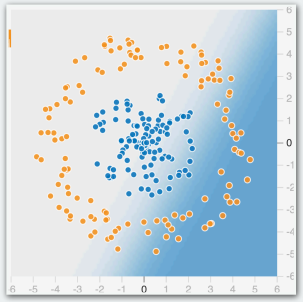
$$\mathbf{y} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \cdots \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

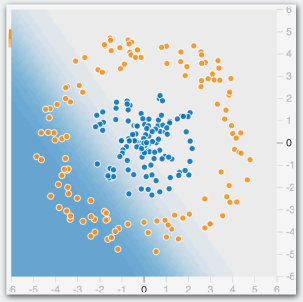
$$\begin{aligned} C(\mathbf{W}) &= \frac{1}{N}\sum_i (\tilde{y}_i - y_i)^2 \\ &= (0.96 - 1)^2 \\ &+ (0.10 - 0)^2 \\ &+ (0.04 - 0)^2 \\ &+ \ldots \\ &+ (0.70 - 1)^2 \\ &+ (0.02 - 0)^2 \\ &+ (0.99 - 1)^2 \end{aligned}$$

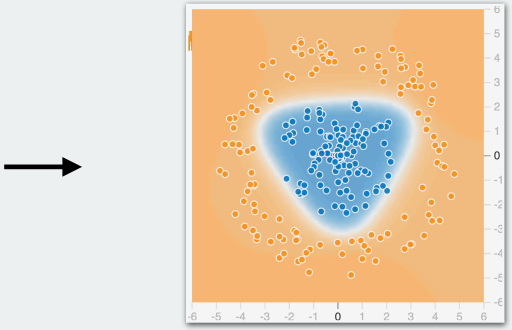


How do we find the weights that give the best classification?









example

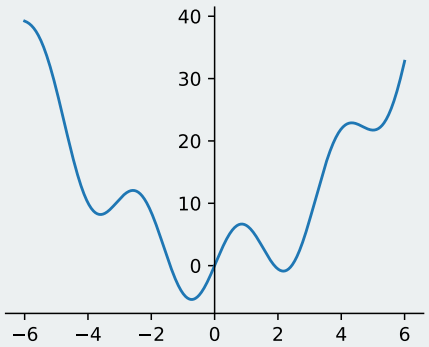
predicted from X given W

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

true

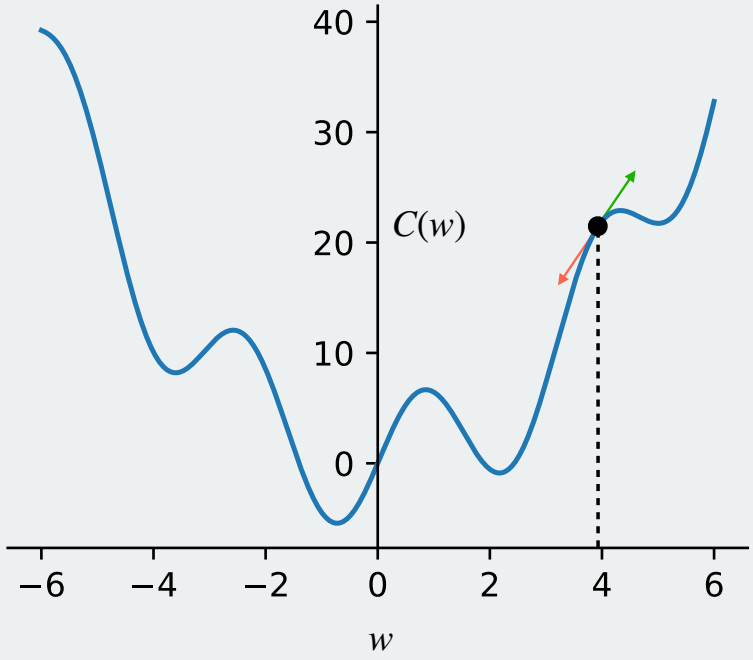
$$\mathbf{y} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \cdots \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

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



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

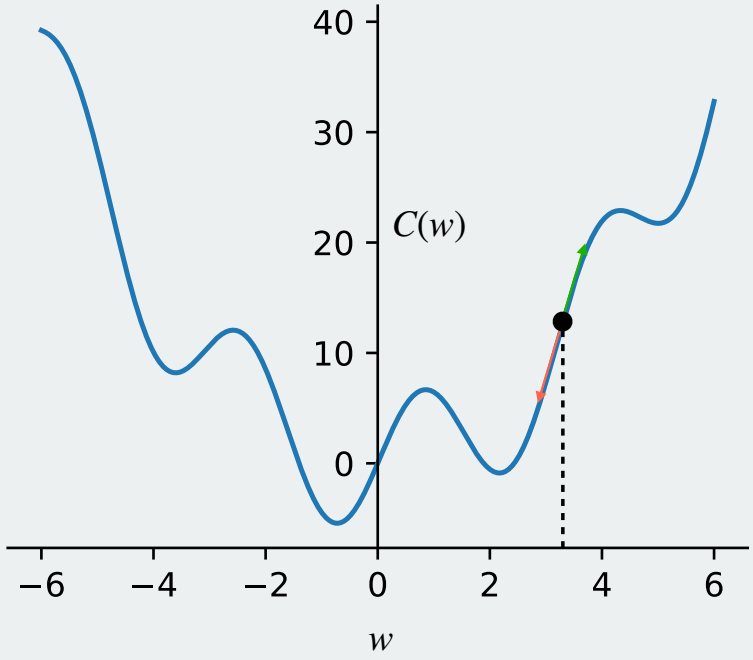
> Gradient Descent



$$\tilde{\mathbf{y}} = \begin{bmatrix} 0.82 \\ 0.47 \\ 0.21 \\ \cdots \\ 0.41 \\ 0.74 \\ 0.23 \end{bmatrix}$$

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

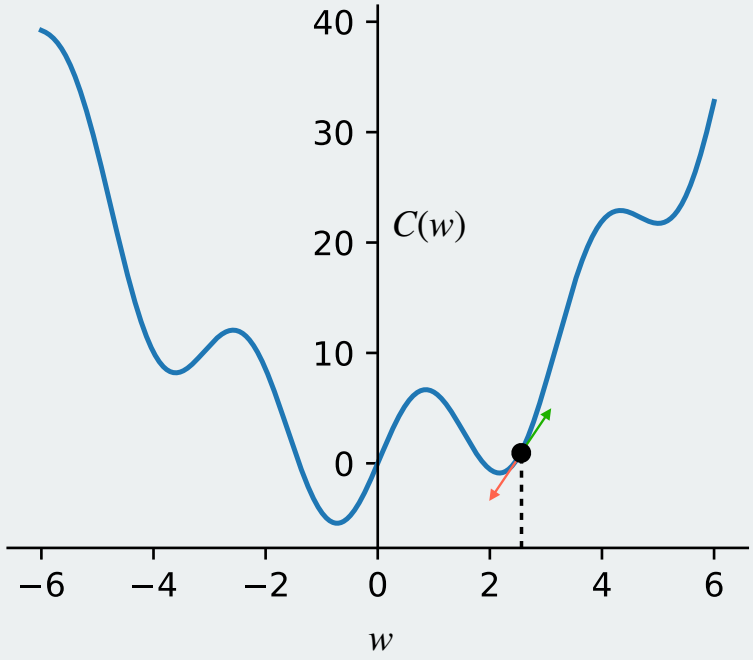
> Gradient Descent



$$\tilde{\mathbf{y}} = \begin{bmatrix} 0.82 \\ 0.32 \\ 0.12 \\ \cdots \\ 0.25 \\ 0.85 \\ 0.11 \end{bmatrix}$$

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

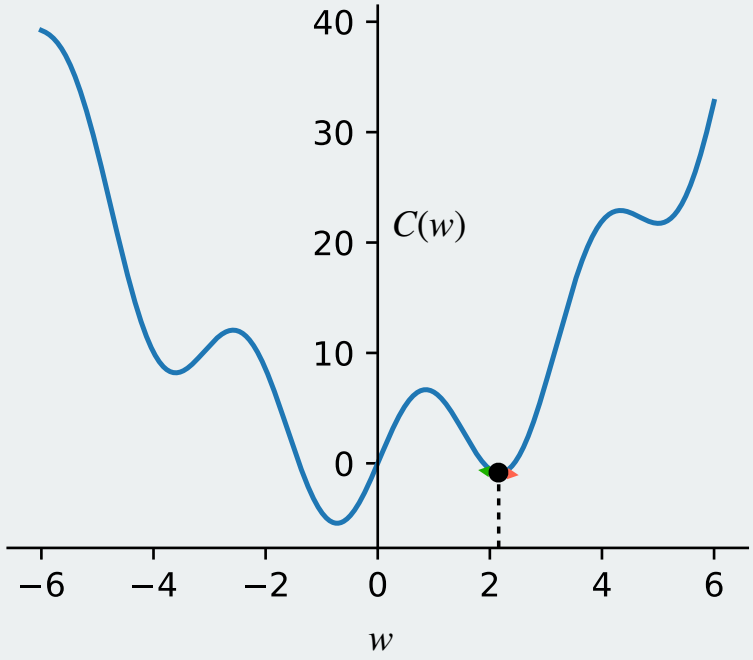
> Gradient Descent



$$\tilde{\mathbf{y}} = \begin{bmatrix} 0.96 \\ 0.08 \\ 0.02 \\ \cdots \\ 0.12 \\ 0.96 \\ 0.01 \end{bmatrix}$$

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

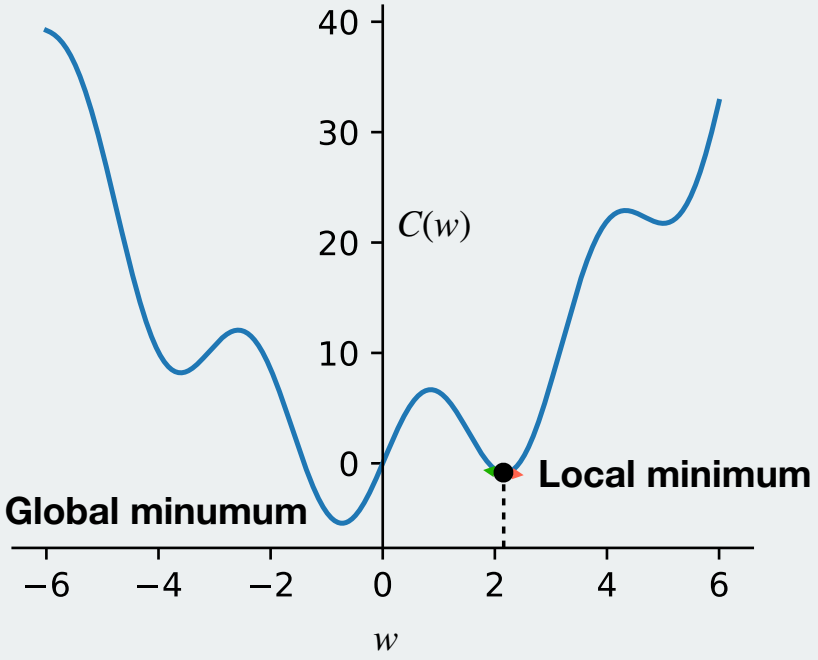
> Gradient Descent



$$\tilde{\mathbf{y}} = \begin{bmatrix} 0.99 \\ 0.02 \\ 0.00 \\ \cdots \\ 0.01 \\ 1.00 \\ 0.00 \end{bmatrix}$$

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

> Gradient Descent

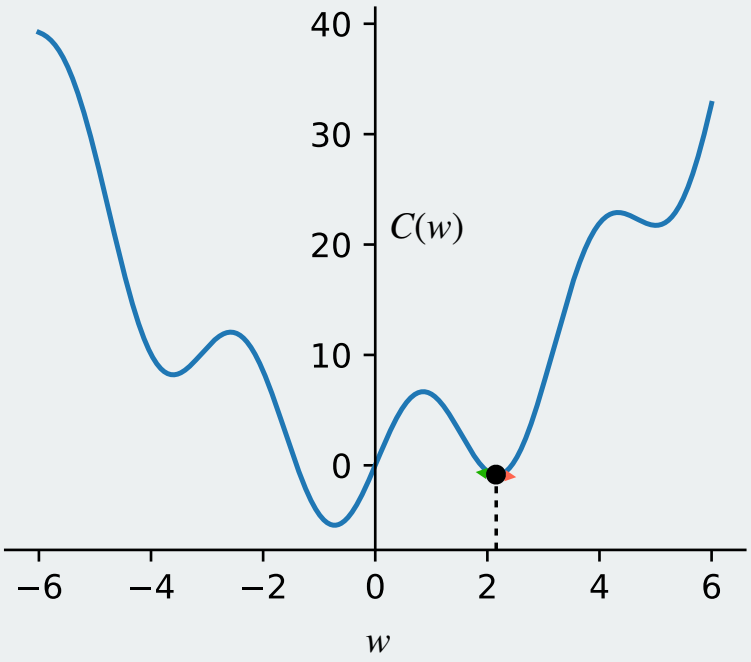


$$\tilde{\mathbf{y}} = \begin{bmatrix} 0.99 \\ 0.02 \\ 0.00 \\ \cdots \\ 0.01 \\ 1.00 \\ 0.00 \end{bmatrix}$$

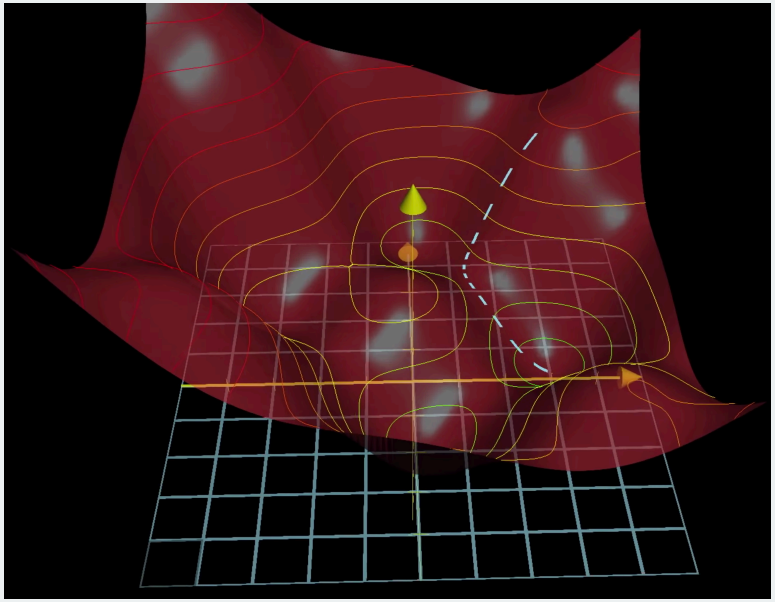
How do we find the weights that give the best classification?

> Gradient Descent

1D



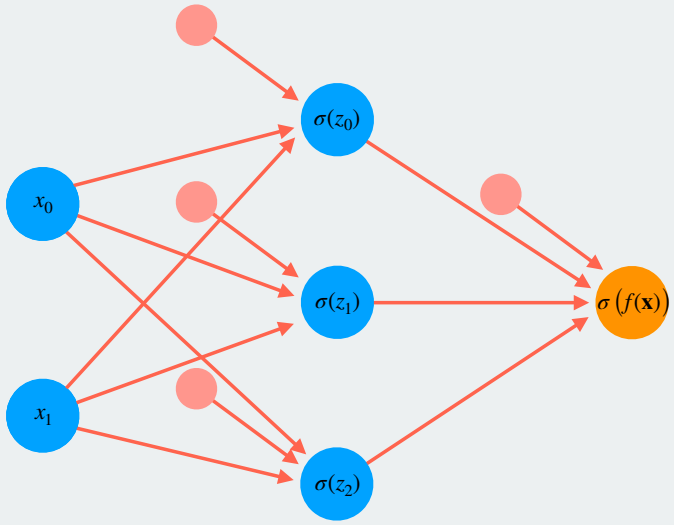
2D



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

> Gradient Descent

13D? Or higher?

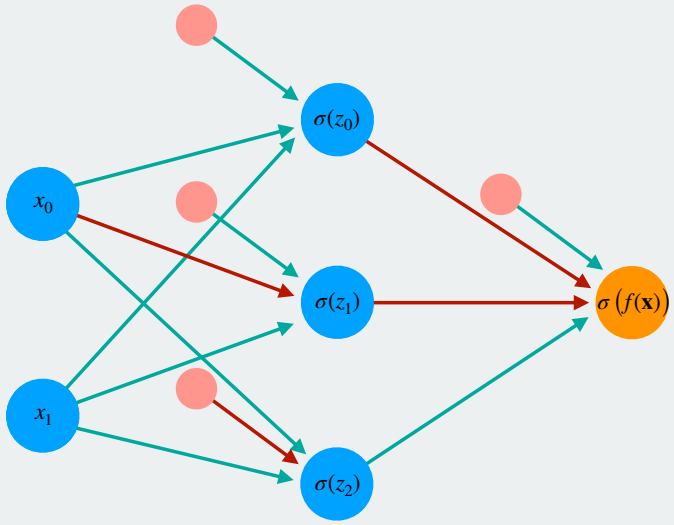


$$\mathbf{W} = \begin{bmatrix} w_{0,0} \\ w_{1,0} \\ w_{2,0} \\ w_{3,0} \\ w_{4,0} \\ w_{5,0} \\ w_{6,0} \\ w_{7,0} \\ w_{8,0} \\ w_{0,1} \\ w_{1,1} \\ w_{2,1} \\ w_{3,1} \end{bmatrix}$$

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

> Gradient Descent

13D? Or higher?

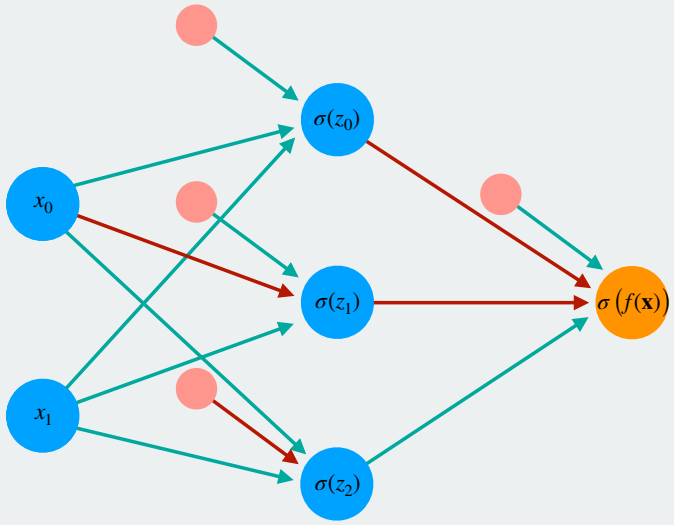


$$\mathbf{W} = \begin{bmatrix} w_{0,0} \\ w_{1,0} \\ w_{2,0} \\ w_{3,0} \\ w_{4,0} \\ w_{5,0} \\ w_{6,0} \\ w_{7,0} \\ w_{8,0} \\ w_{0,1} \\ w_{1,1} \\ w_{2,1} \\ w_{3,1} \end{bmatrix} \qquad -\nabla C(\mathbf{W}) = \begin{bmatrix} -0.23 \\ 1.32 \\ 0.20 \\ 0.38 \\ -1.23 \\ 0.01 \\ 1.20 \\ -2.12 \\ 0.73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

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

> Gradient Descent

13D? Or higher?



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

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

How it all hangs together

(1) The model

Input → Neural network function → **Output**

Datapoints, **X**

Parameters: Weights, **W**

Prediction, $\tilde{\mathbf{y}}$

(2) Its performance

Input → Cost function → **Output**

Weights, **W**

One number, **the cost**

(3) The cost function gradient in **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 **W**

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

(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

Input → Neural network function → **Output**

Datapoints, **X**

Parameters: Weights, **W**

Prediction, $\tilde{\mathbf{y}}$

(2) Its performance

Input → Cost function → **Output**

Weights, **W**

One number, **the cost**

(3) The cost function gradient in **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 **W**

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

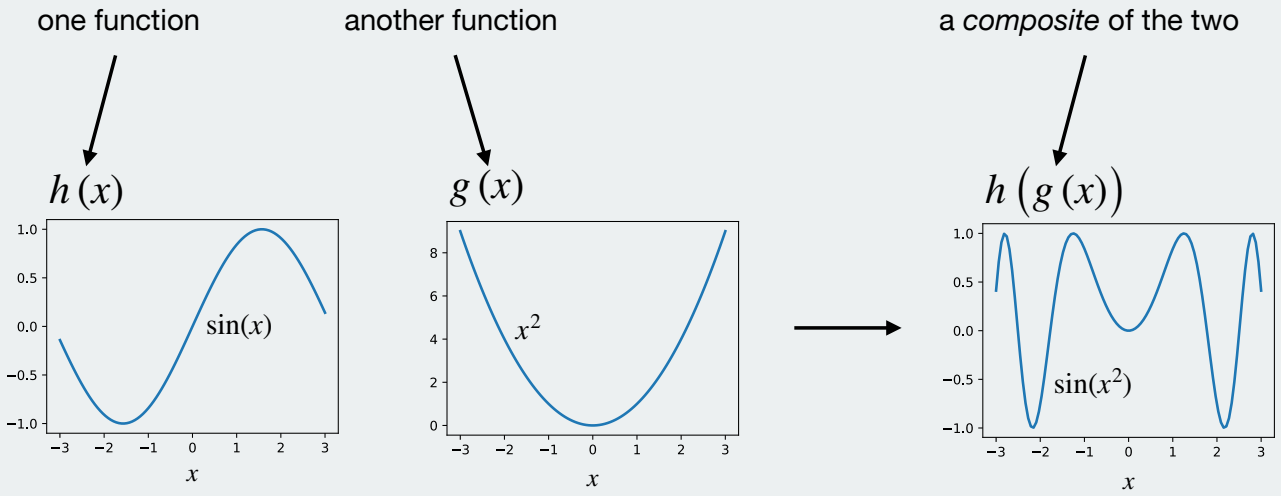
(5) Repeat 3 and 4

r is usually called the *learning rate*

Backpropagation

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

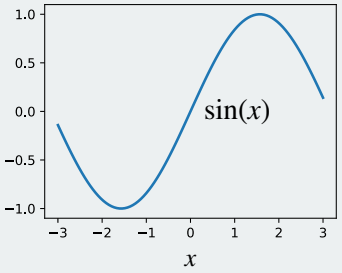
Chain rule – Taking derivatives of *composite functions*



Chain rule – Taking derivatives of *composite functions*

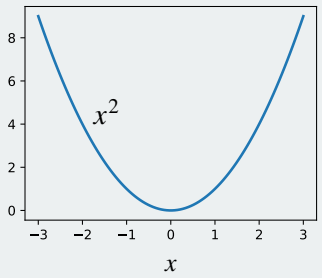
one function

$h(x)$



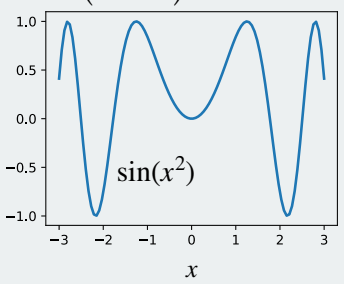
another function

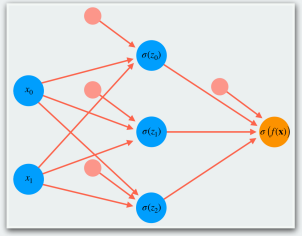
$g(x)$



a *composite* of the two

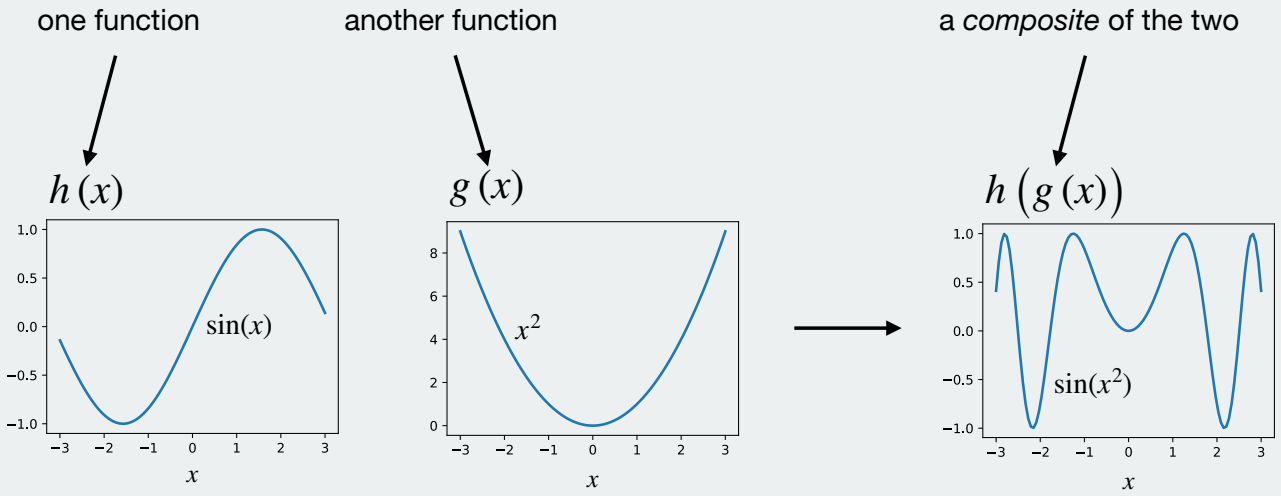
$h\left(g(x)\right)$





$$\sigma\left(w_{0,1} + \sigma(w_{0,0} + x_0 w_{1,0} + x_1 w_{2,0}) w_{1,1} \;+\right.$$
$$\sigma(w_{3,0} + x_0 w_{4,0} + x_1 w_{5,0}) w_{2,1} \;+$$
$$\left.\sigma(w_{6,0} + x_0 w_{7,0} + x_1 w_{8,0}) w_{3,1}\right) = \sigma\left(f(x)\right)$$

Chain rule – Taking derivatives of *composite functions*

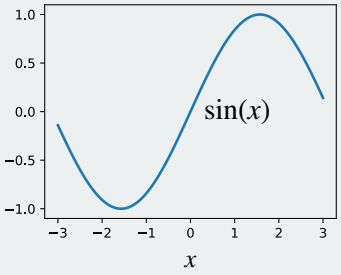


$$h\left(g\left(x\right)\right)$$

Chain rule – Taking derivatives of *composite functions*

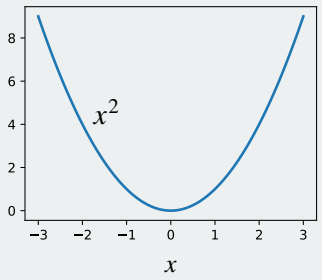
one function

$h(x)$



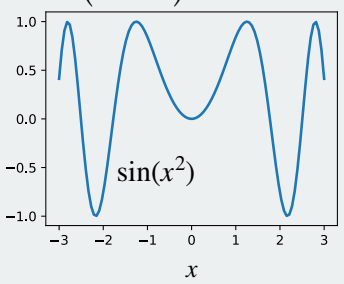
another function

$g(x)$



a *composite* of the two

$h(g(x))$

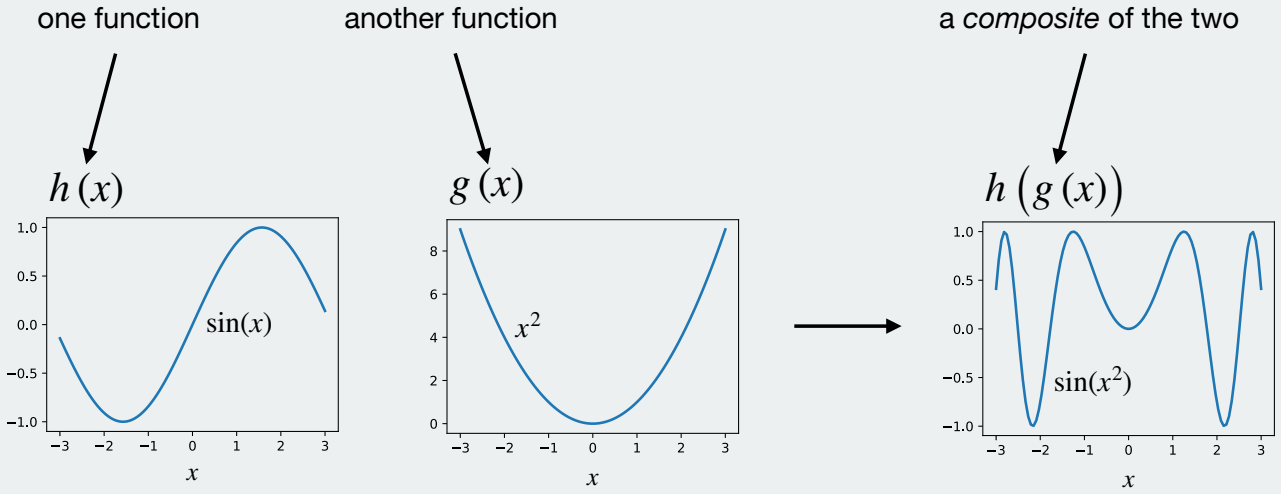


$h(g(x))$

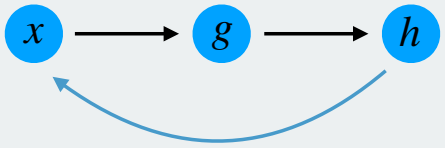
Chain rule says:

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

Chain rule – Taking derivatives of *composite functions*



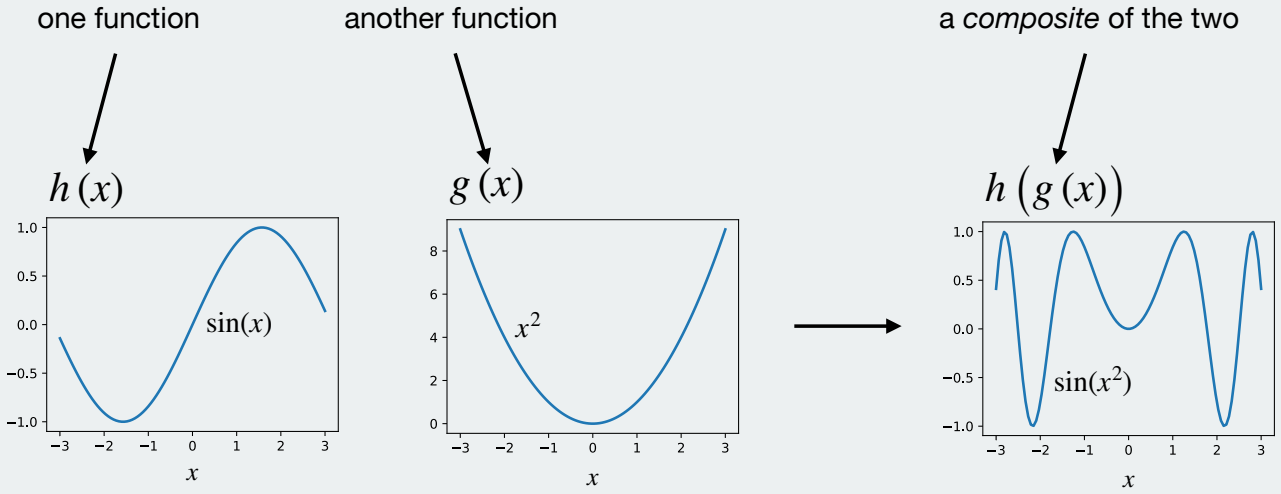
$h(g(x))$



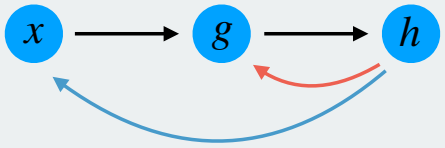
Chain rule says:

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

Chain rule – Taking derivatives of *composite functions*



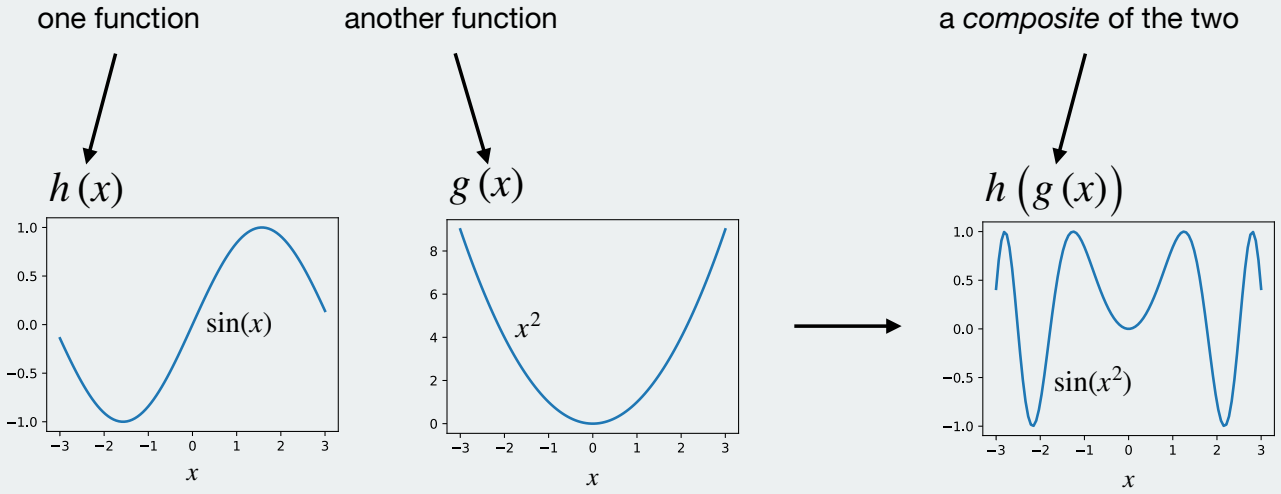
$h(g(x))$



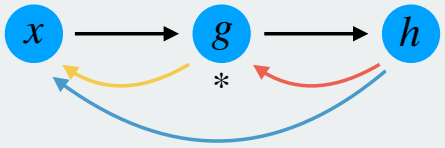
Chain rule says:

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

Chain rule – Taking derivatives of *composite functions*



$h(g(x))$



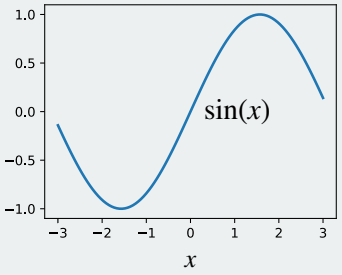
Chain rule says:

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

Chain rule – Taking derivatives of *composite functions*

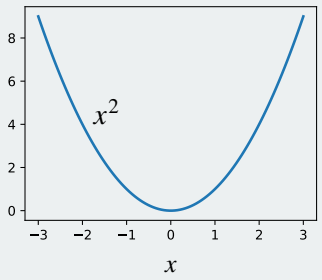
one function

$h(x)$



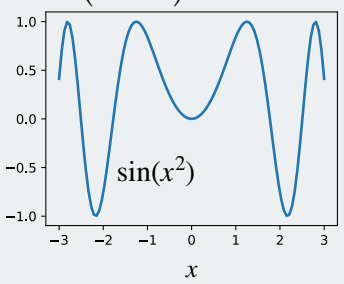
another function

$g(x)$

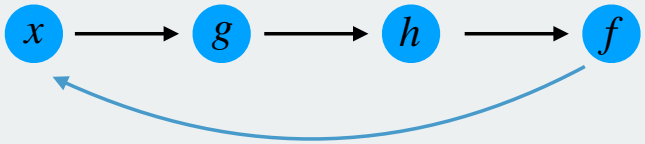


a *composite* of the two

$h(g(x))$



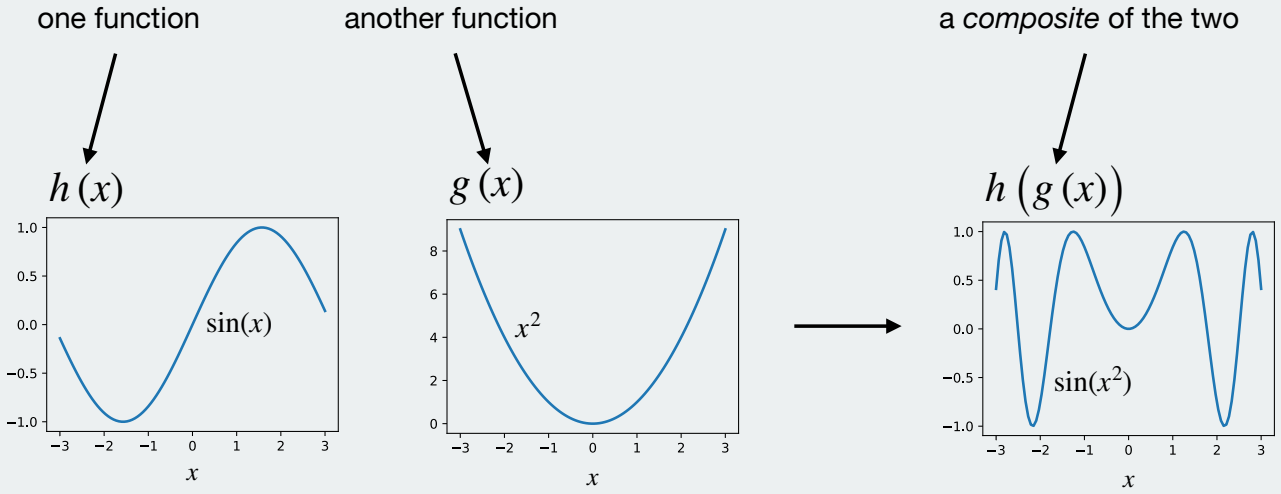
$f(h(g(x)))$



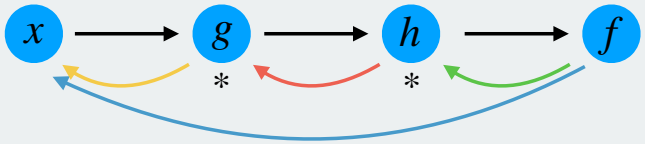
Chain rule says:

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

Chain rule – Taking derivatives of *composite functions*



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



Chain rule says:

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

3Blue1Brown Chain Rule video

Backpropagation – simple example

Model:

$$m(z) = -z$$

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

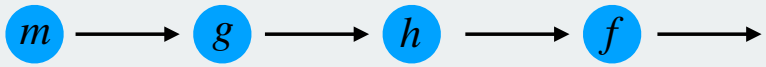
$$h(z) = z + 1$$

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

Data:

$$x_0 = 1$$

$$x_1 = 1.1$$



Backpropagation – simple example

Model:

$$m(z) = -z$$

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

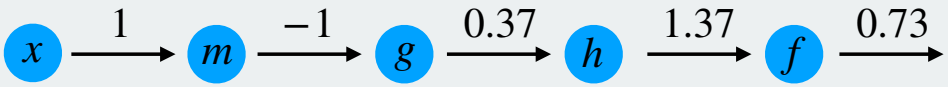
$$h(z) = z + 1$$

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

Data:

$$x_0 = 1$$

$$x_1 = 1.1$$



Backpropagation – simple example

Model:

$$m(z) = -z$$

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

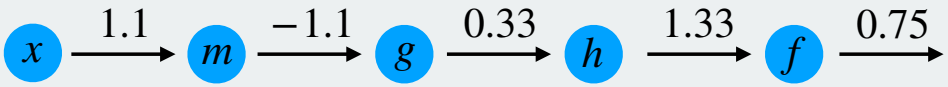
$$h(z) = z + 1$$

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

Data:

$$x_0 = 1$$

$$x_1 = 1.1$$



Backpropagation – simple example

Model:

$$m(z) = -z$$

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

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

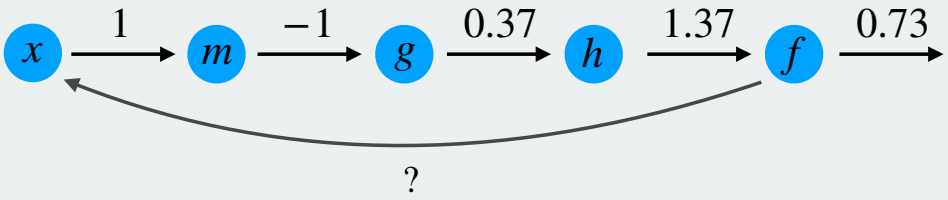
$$f(z) = \frac{1}{z}$$

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:

$$m(z) = -z$$

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

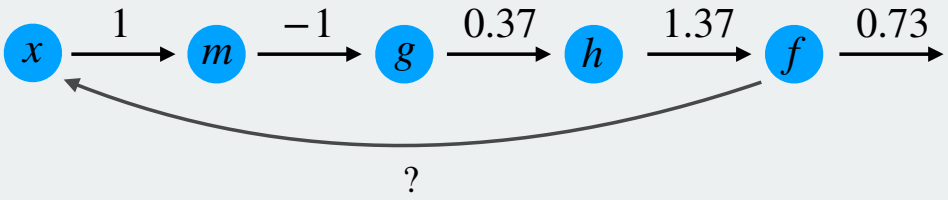
$$h(z) = z + 1$$

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

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:

$$m(z) = -z$$

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

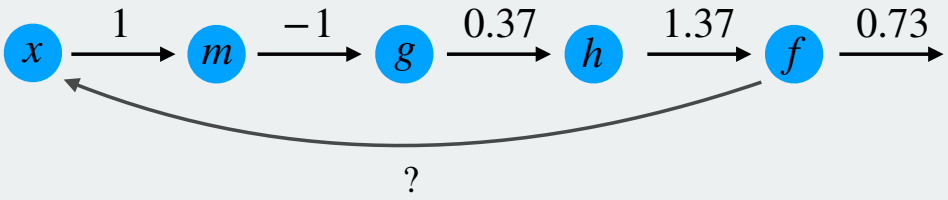
$$h(z) = z + 1$$

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

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 *x*:

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

Model derivatives:

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

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

$$h'(z) = 1$$

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

Backpropagation – simple example

Model:

$$m(z) = -z$$

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

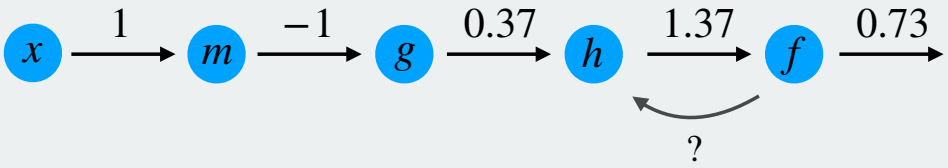
$$h(z) = z + 1$$

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

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 *h*:

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

Model derivatives:

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

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

$$h'(z) = 1$$

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

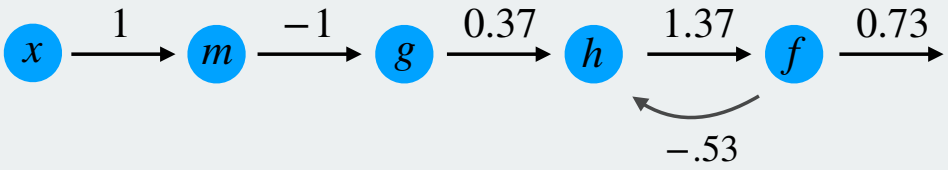
Backpropagation – simple example

Model:

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

Data:

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



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

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:

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

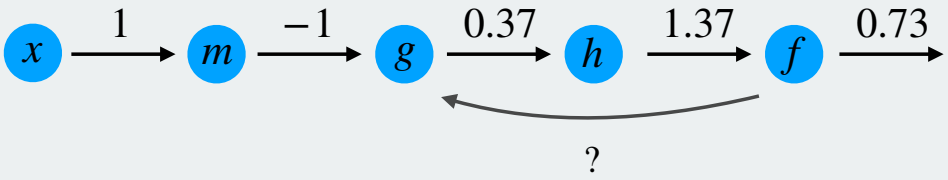
Backpropagation – simple example

Model:

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

Data:

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



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

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:

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

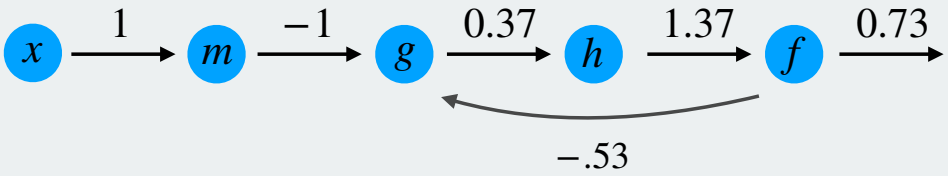
Backpropagation – simple example

Model:

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

Data:

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



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

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:

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

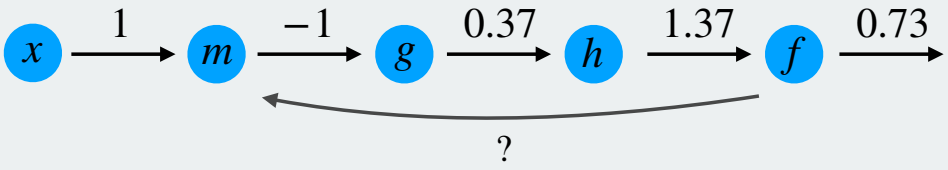
Backpropagation – simple example

Model:

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

Data:

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



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

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:

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

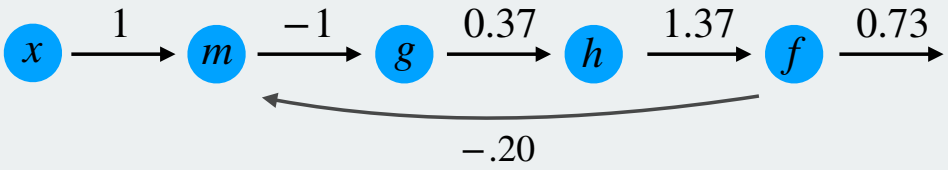
Backpropagation – simple example

Model:

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

Data:

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



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

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:

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

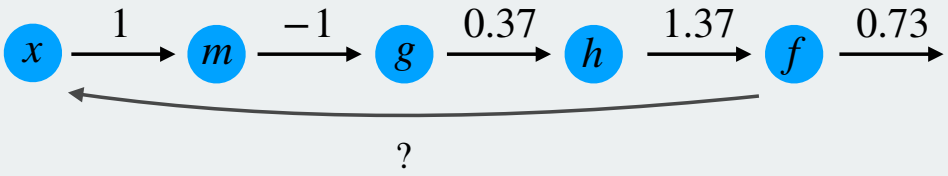
Backpropagation – simple example

Model:

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

Data:

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



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

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:

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

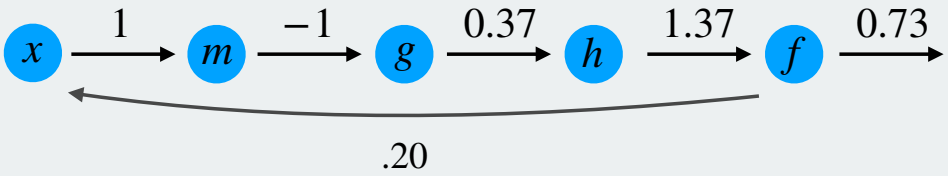
Backpropagation – simple example

Model:

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

Data:

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



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

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:

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

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 $f(h(g(m(x))))$?

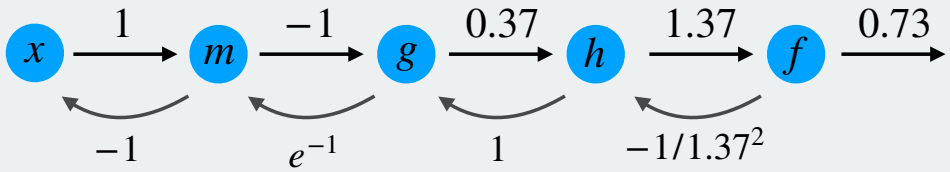
A: Propagate gradients backwards using the chain rule!

Chain rule:

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

Model derivatives:

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



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

Sigmoid function:

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

Sigmoid derivative:

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

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 $f(h(g(m(x))))$?

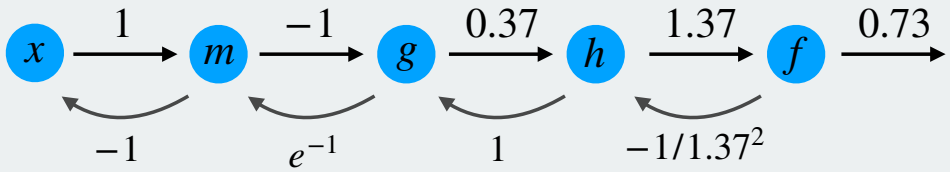
A: Propagate gradients backwards using the chain rule!

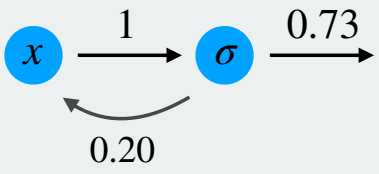
Chain rule:

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

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 – 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 $f(h(g(m(x))))$?

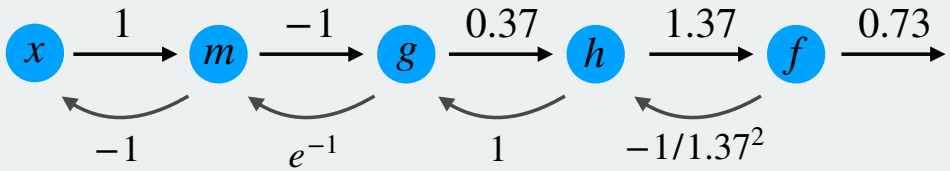
A: Propagate gradients backwards using the chain rule!

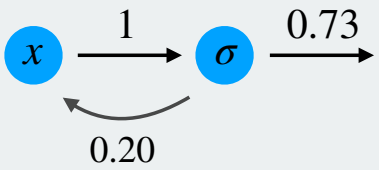
Chain rule:

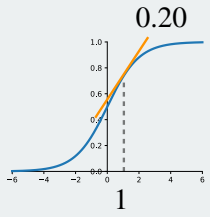
$$\frac{df}{dx} = \frac{df}{dh}\frac{dh}{dg}\frac{dg}{dm}\frac{dm}{dx}$$

Model derivatives:

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







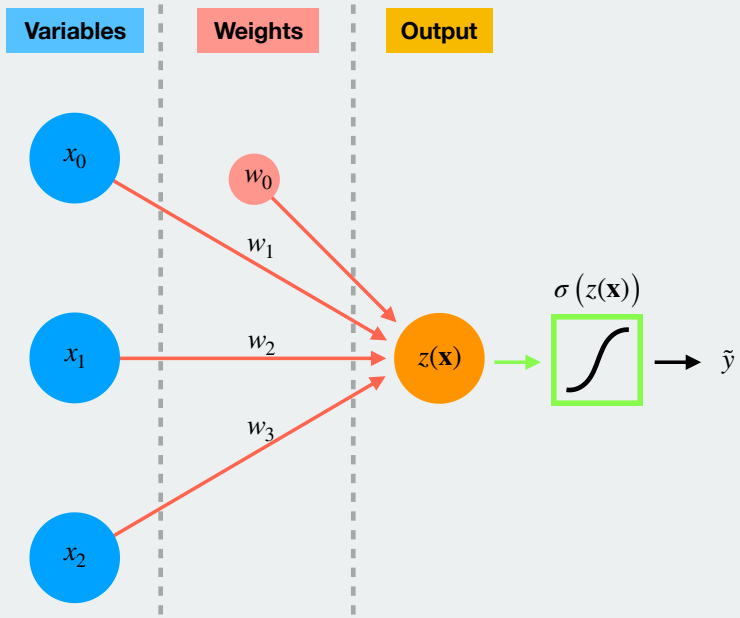
Sigmoid function:

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

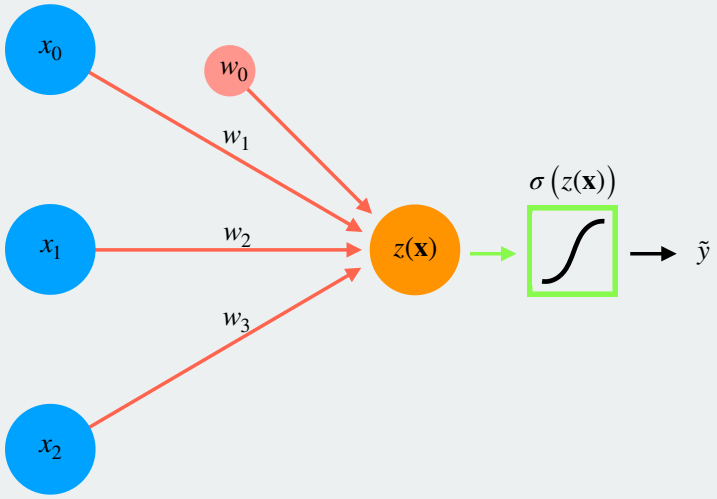
Sigmoid derivative:

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

Backpropagation – on a neural network



Backpropagation – on a neural network



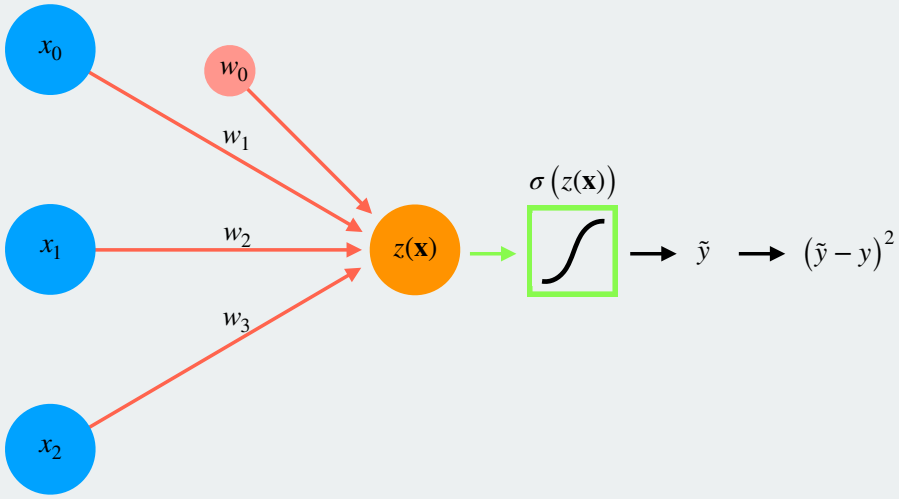
Model:

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

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

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

Backpropagation – on a neural network



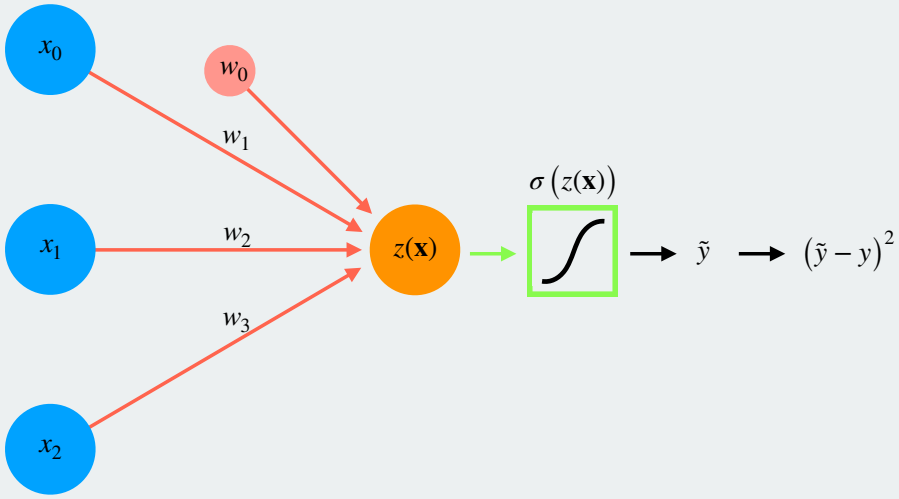
Model:

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

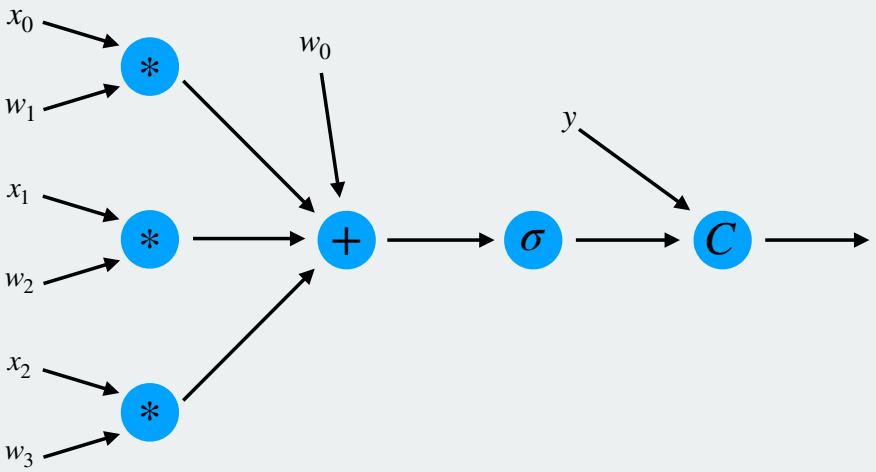
$$\sigma(x) = \frac{1}{1 + \exp(-x)}$$

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

Backpropagation – on a neural network



As a computational graph:



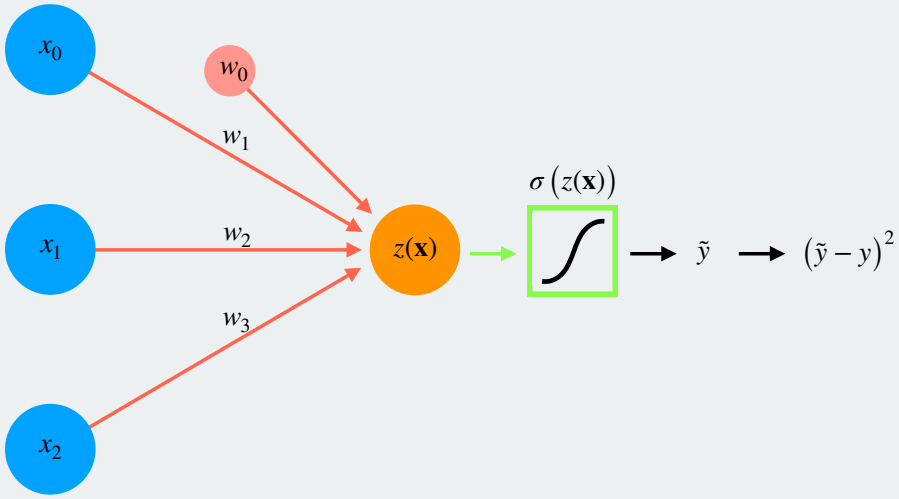
Model:

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

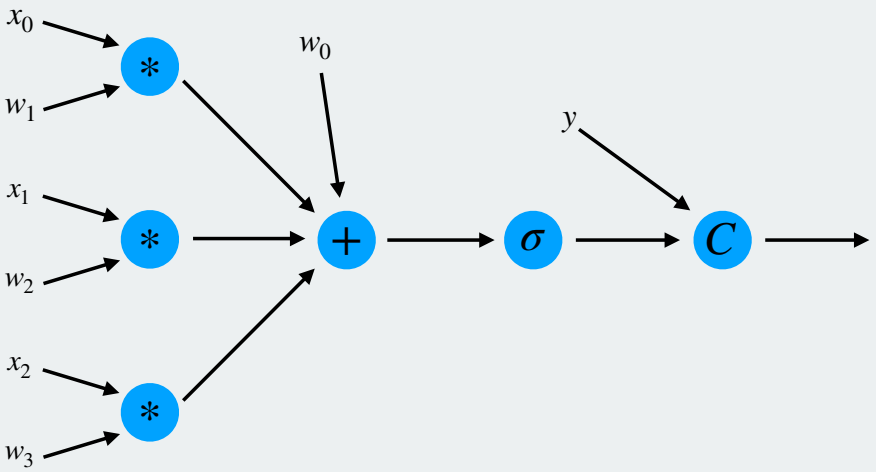
$$\sigma(x) = \frac{1}{1 + \exp(-x)}$$

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

Backpropagation – on a neural network



As a computational graph:



Model:

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

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

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

Weights:

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

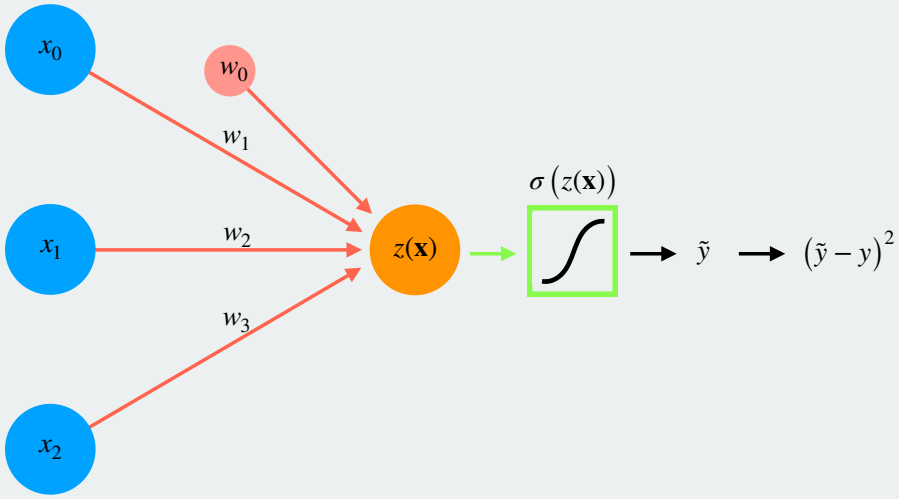
$$\mathbf{W} = \begin{bmatrix} -1 & 0.5 & 10 \end{bmatrix}$$

Data:

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

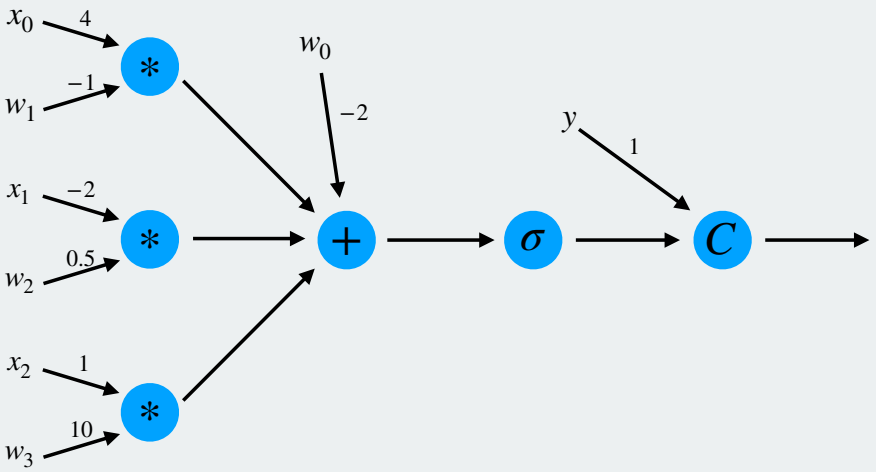
$$\mathbf{y} = \begin{bmatrix} 1 \end{bmatrix}$$

Backpropagation – on a neural network



As a computational graph:

Forward pass



Model:

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

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

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

Weights:

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

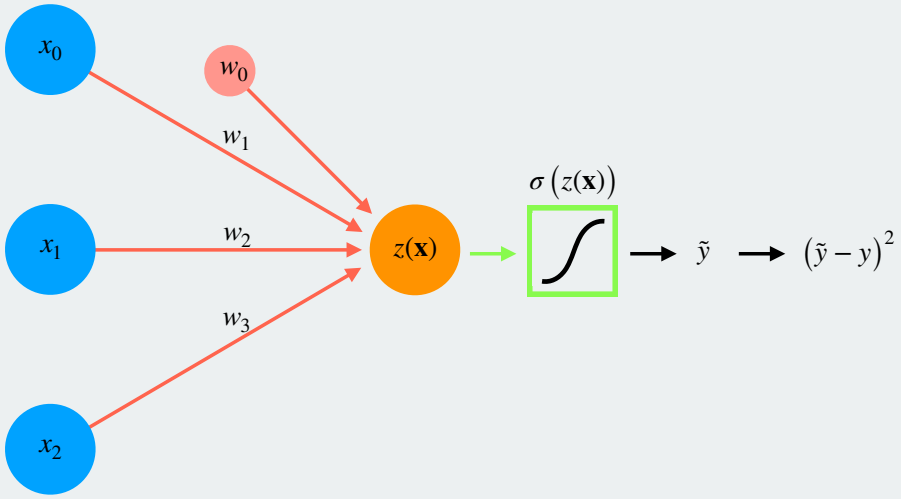
$$\mathbf{W} = \begin{bmatrix} -1 & 0.5 & 10 \end{bmatrix}$$

Data:

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

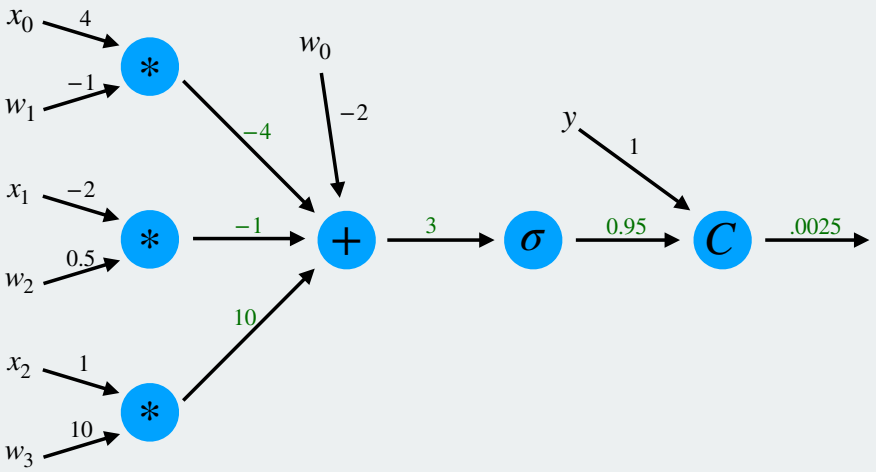
$$\mathbf{y} = \begin{bmatrix} 1 \end{bmatrix}$$

Backpropagation – on a neural network



As a computational graph:

Forward pass



Model:

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

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

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

Weights:

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

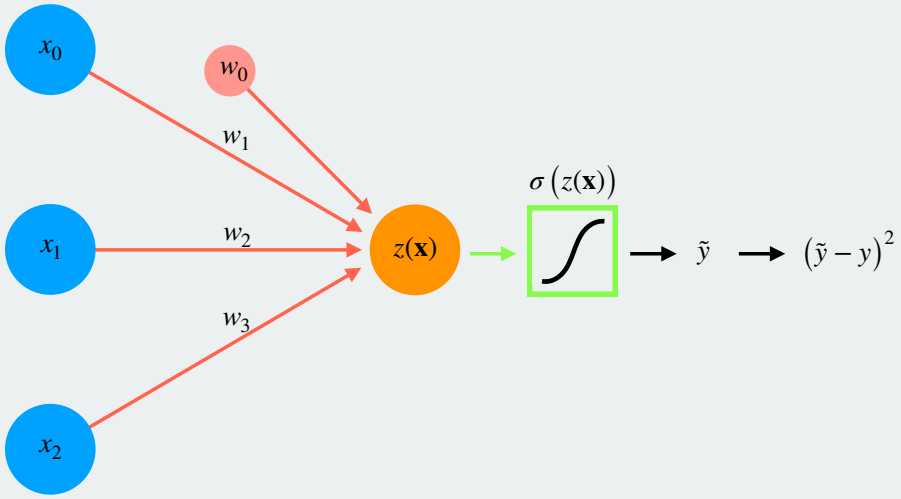
$$\mathbf{W} = \begin{bmatrix} -1 & 0.5 & 10 \end{bmatrix}$$

Data:

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

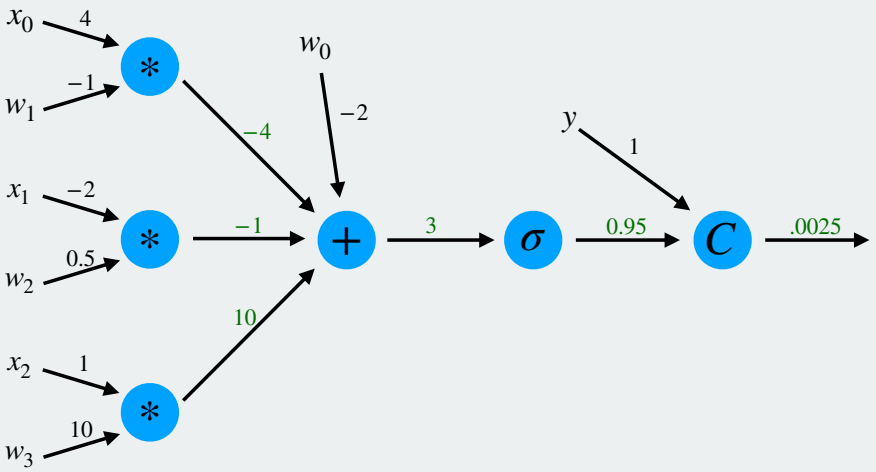
$$\mathbf{y} = \begin{bmatrix} 1 \end{bmatrix}$$

Backpropagation – on a neural network



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:

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

Weights:

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

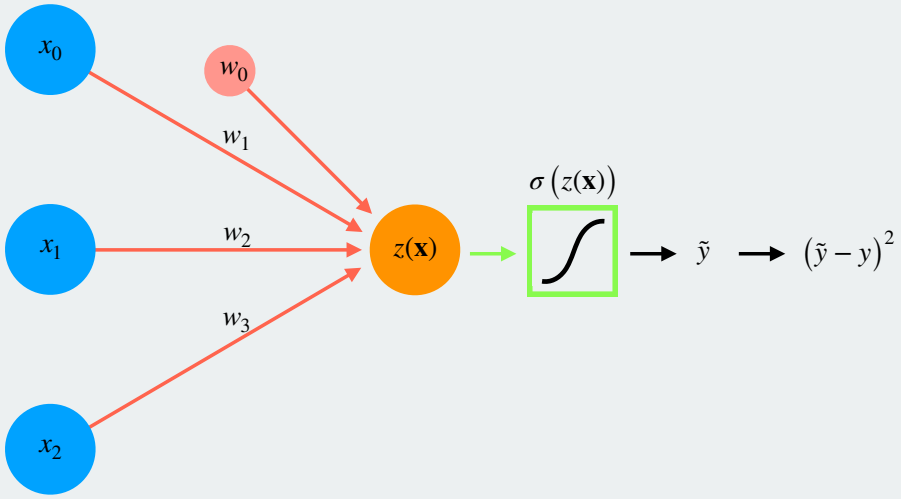
$$\mathbf{W} = \begin{bmatrix} -1 & 0.5 & 10 \end{bmatrix}$$

Data:

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

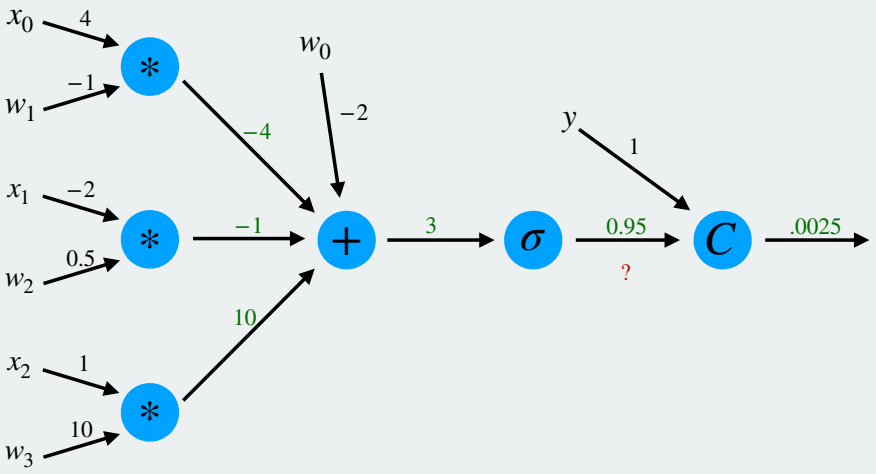
$$\mathbf{y} = \begin{bmatrix} 1 \end{bmatrix}$$

Backpropagation – on a neural network



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:

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

Weights:

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

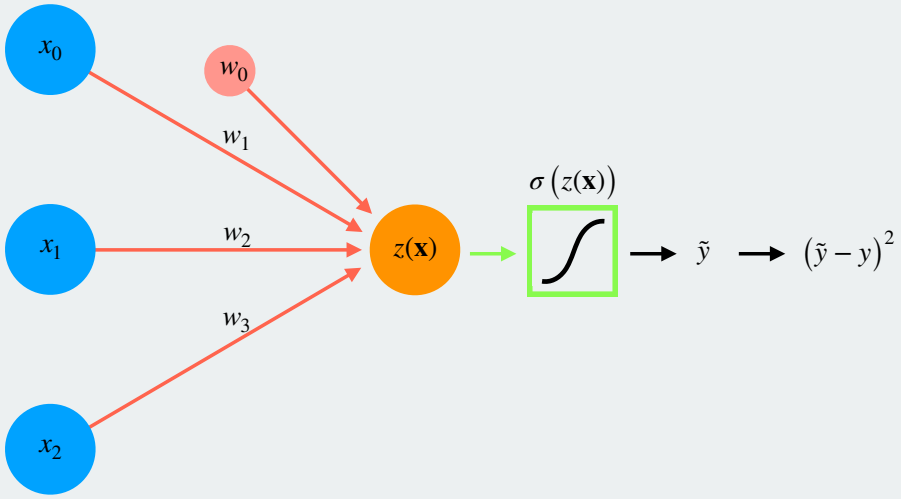
$$\mathbf{W} = \begin{bmatrix} -1 & 0.5 & 10 \end{bmatrix}$$

Data:

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

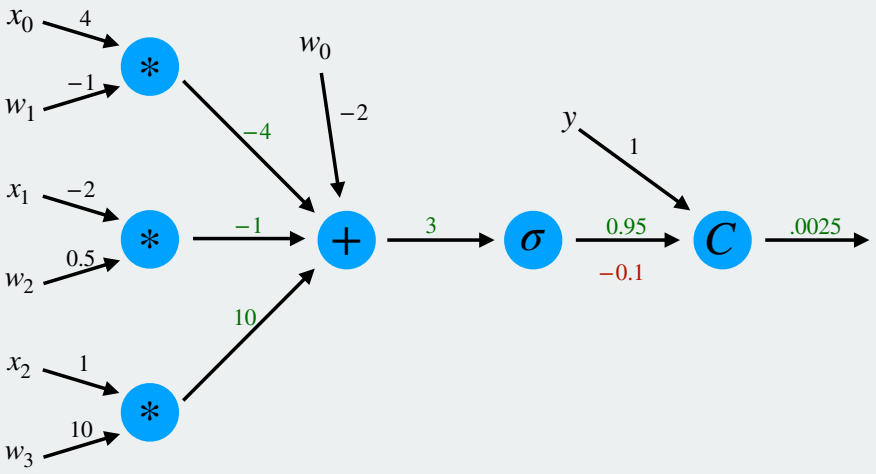
$$\mathbf{y} = \begin{bmatrix} 1 \end{bmatrix}$$

Backpropagation – on a neural network



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:

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

Weights:

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

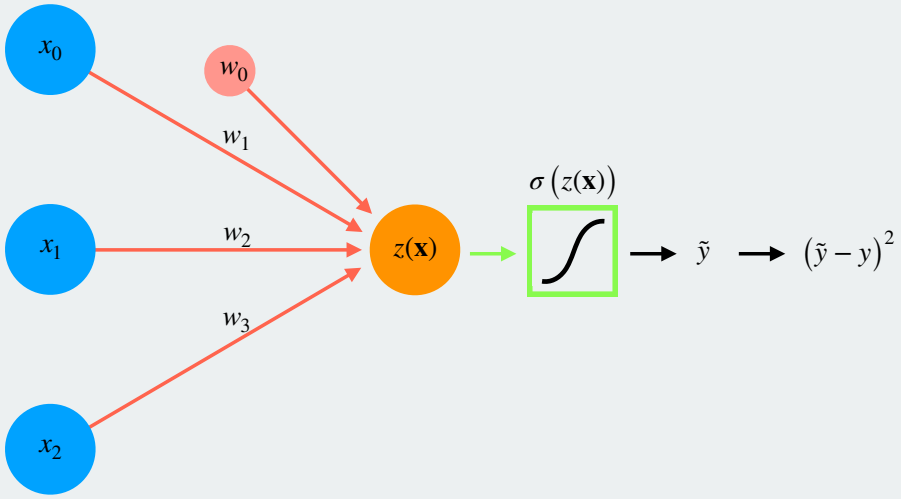
$$\mathbf{W} = \begin{bmatrix} -1 & 0.5 & 10 \end{bmatrix}$$

Data:

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

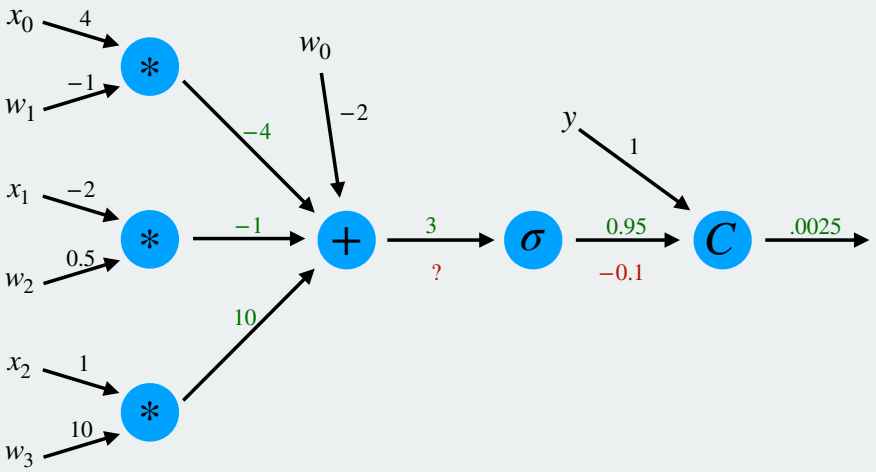
$$\mathbf{y} = \begin{bmatrix} 1 \end{bmatrix}$$

Backpropagation – on a neural network



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:

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

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

Weights:

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

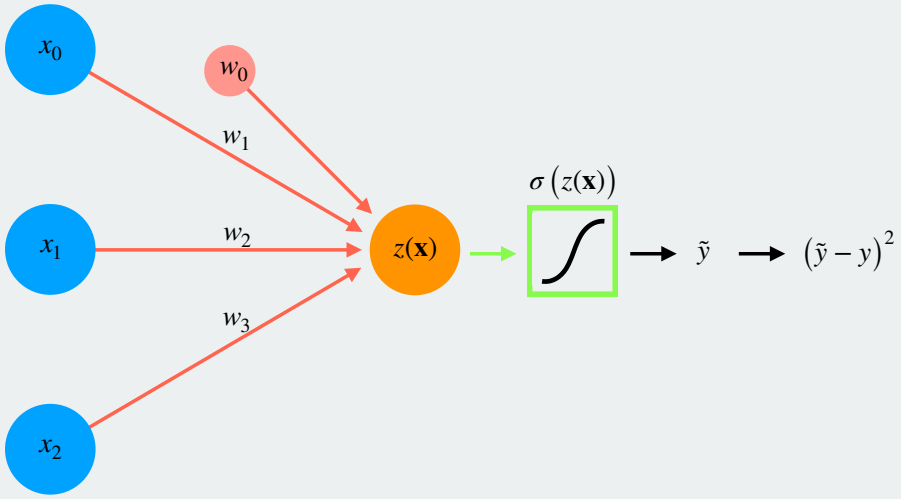
$$\mathbf{W} = \begin{bmatrix} -1 & 0.5 & 10 \end{bmatrix}$$

Data:

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

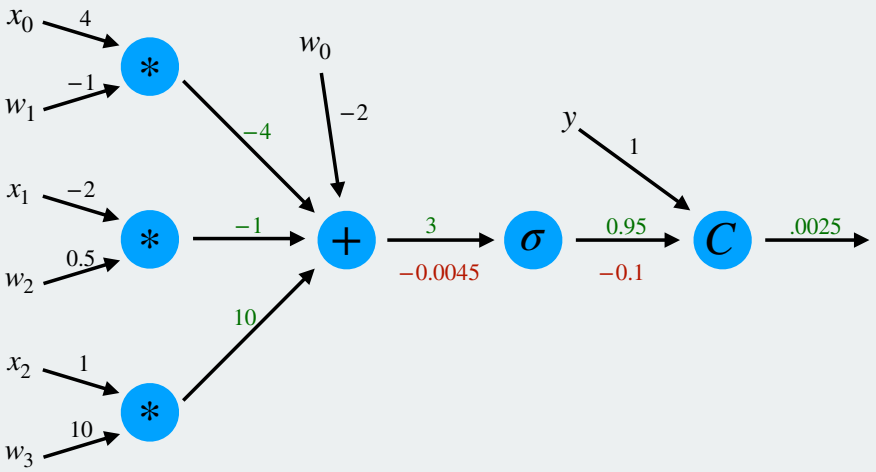
$$\mathbf{y} = \begin{bmatrix} 1 \end{bmatrix}$$

Backpropagation – on a neural network



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: $y = \sum_{i=0}^{N} z_n$

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

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

Weights:

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

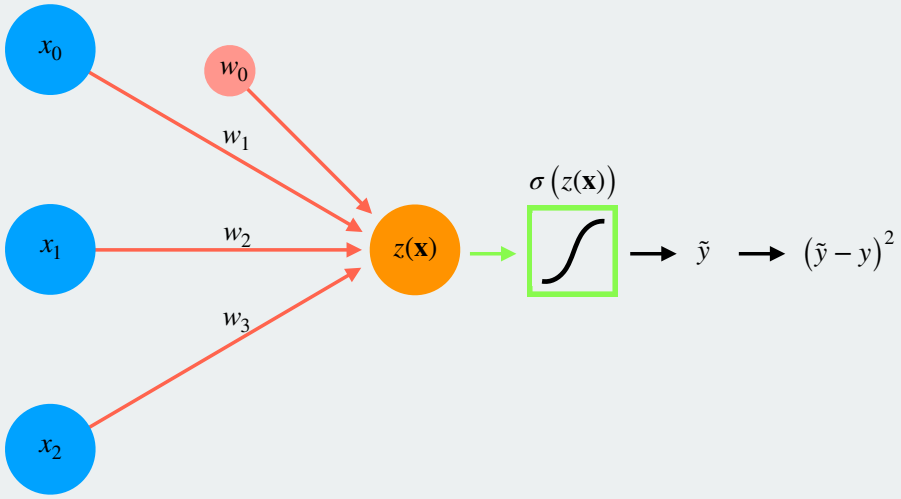
$$\mathbf{W} = \begin{bmatrix} -1 & 0.5 & 10 \end{bmatrix}$$

Data:

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

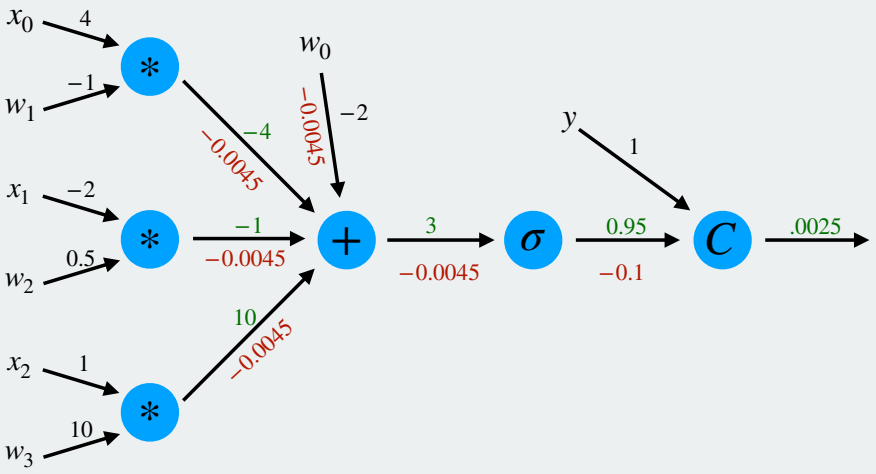
$$\mathbf{y} = \begin{bmatrix} 1 \end{bmatrix}$$

Backpropagation – on a neural network



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: $y = \sum_{i=0}^{N} z_n$

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

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

Weights:

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

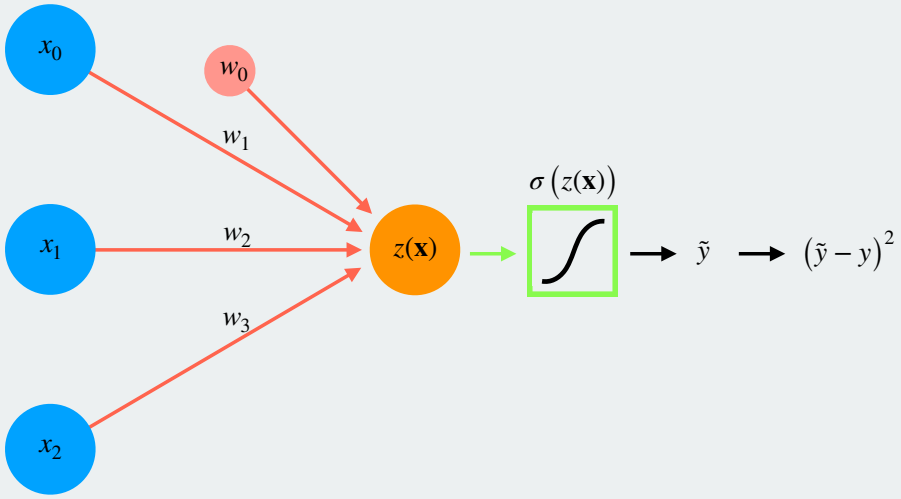
$$\mathbf{W} = \begin{bmatrix} -1 & 0.5 & 10 \end{bmatrix}$$

Data:

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

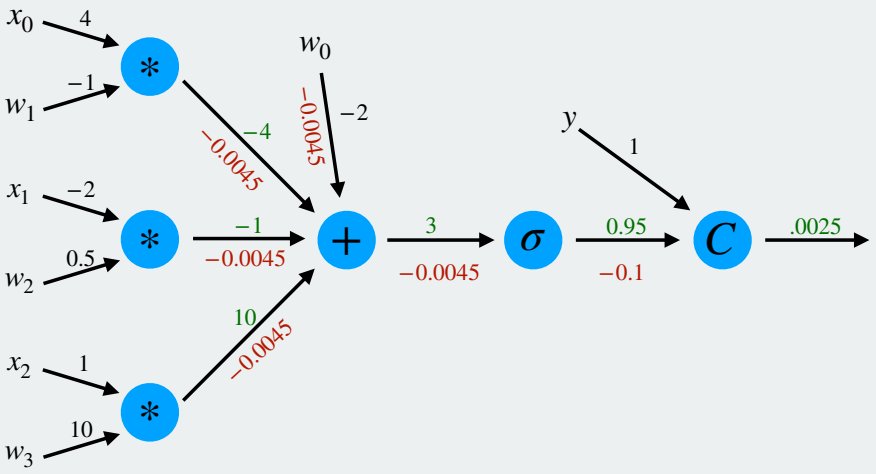
$$\mathbf{y} = \begin{bmatrix} 1 \end{bmatrix}$$

Backpropagation – on a neural network



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:

Each branch is a function: $y = wx$

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

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

Weights:

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

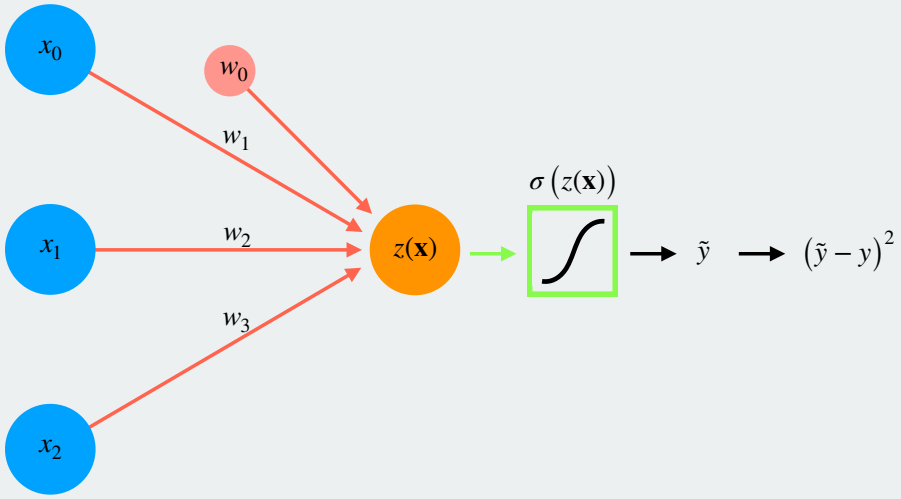
$$\mathbf{W} = \begin{bmatrix} -1 & 0.5 & 10 \end{bmatrix}$$

Data:

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

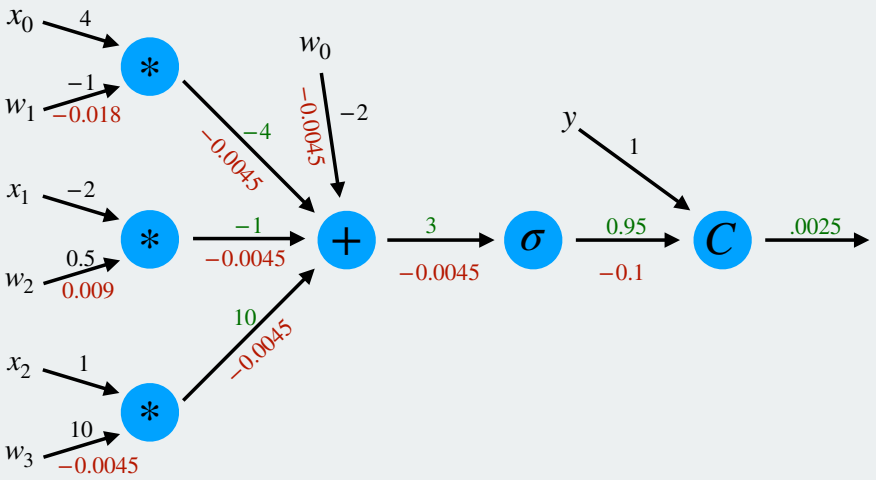
$$\mathbf{y} = \begin{bmatrix} 1 \end{bmatrix}$$

Backpropagation – on a neural network



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:

Each branch is a function: $y = wx$

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

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

Weights:

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

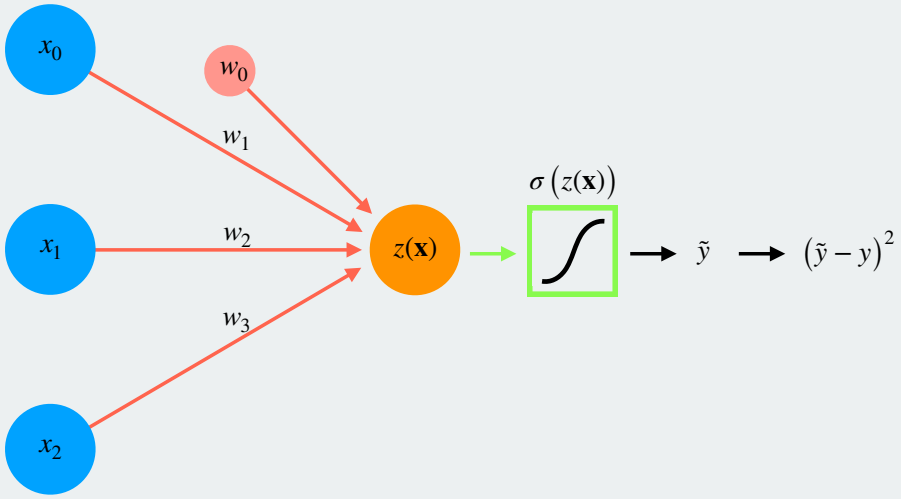
$$\mathbf{W} = \begin{bmatrix} -1 & 0.5 & 10 \end{bmatrix}$$

Data:

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

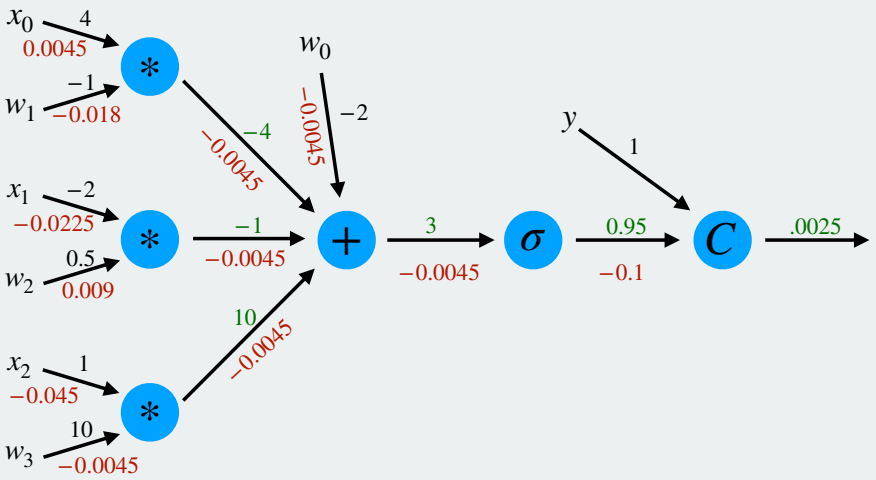
$$\mathbf{y} = \begin{bmatrix} 1 \end{bmatrix}$$

Backpropagation – on a neural network



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:

Each branch is a function: $y = wx$

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

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

Weights:

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

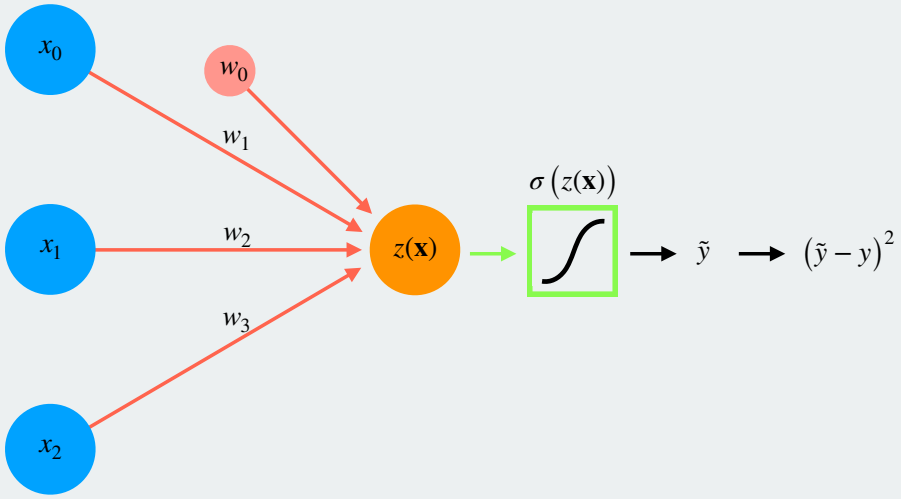
$$\mathbf{W} = \begin{bmatrix} -1 & 0.5 & 10 \end{bmatrix}$$

Data:

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

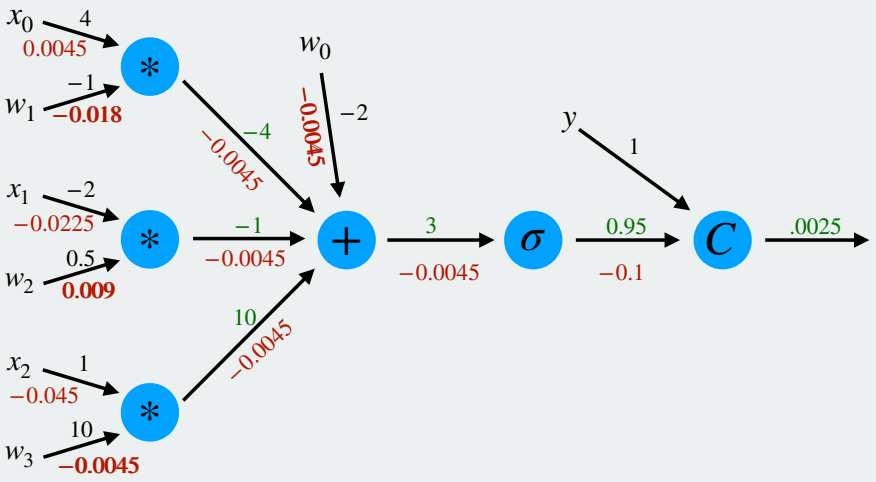
$$\mathbf{y} = \begin{bmatrix} 1 \end{bmatrix}$$

Backpropagation – on a neural network



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:

Each branch is a function: $y = wx$

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

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

Weights:

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

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

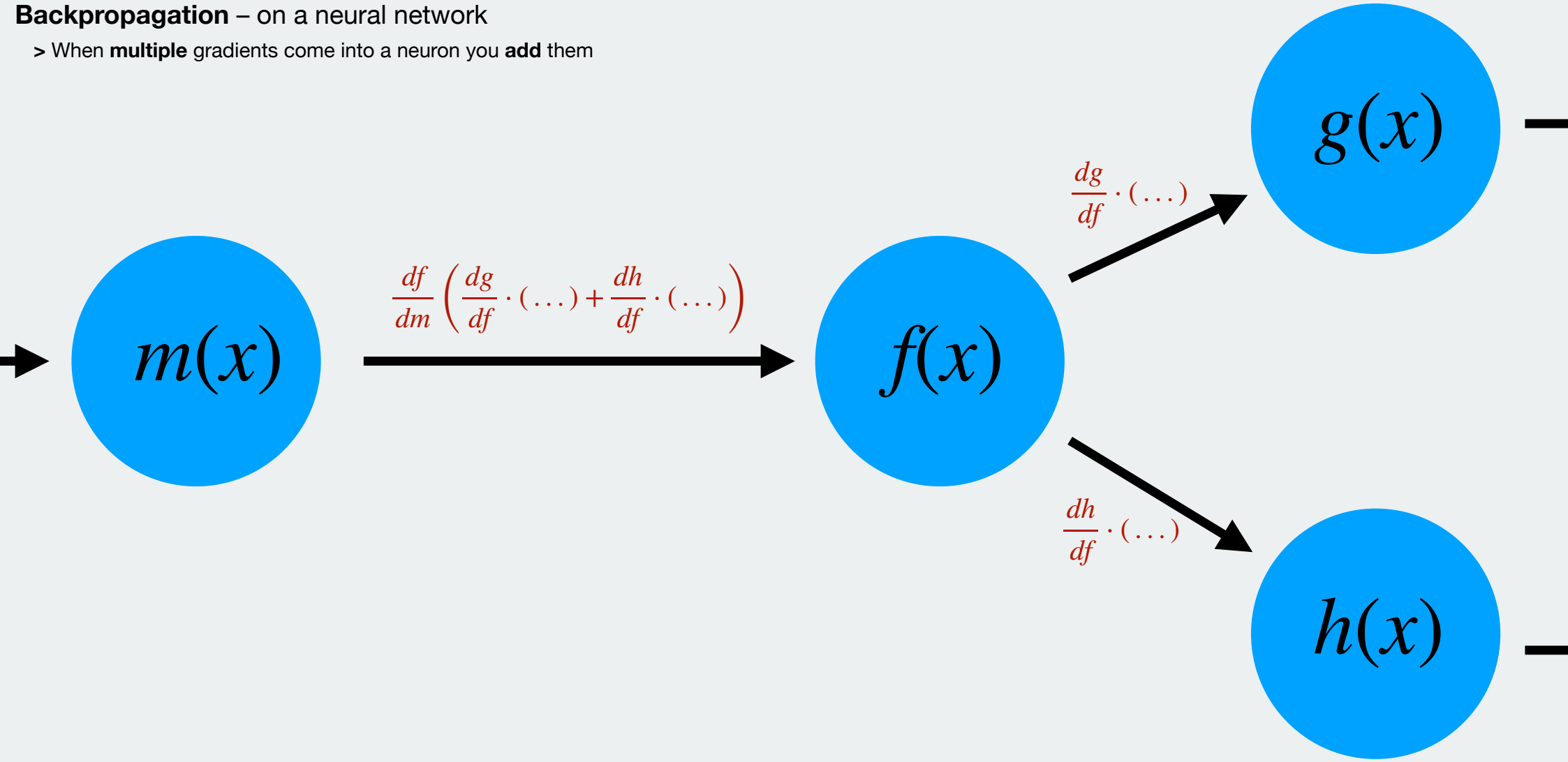
Data:

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

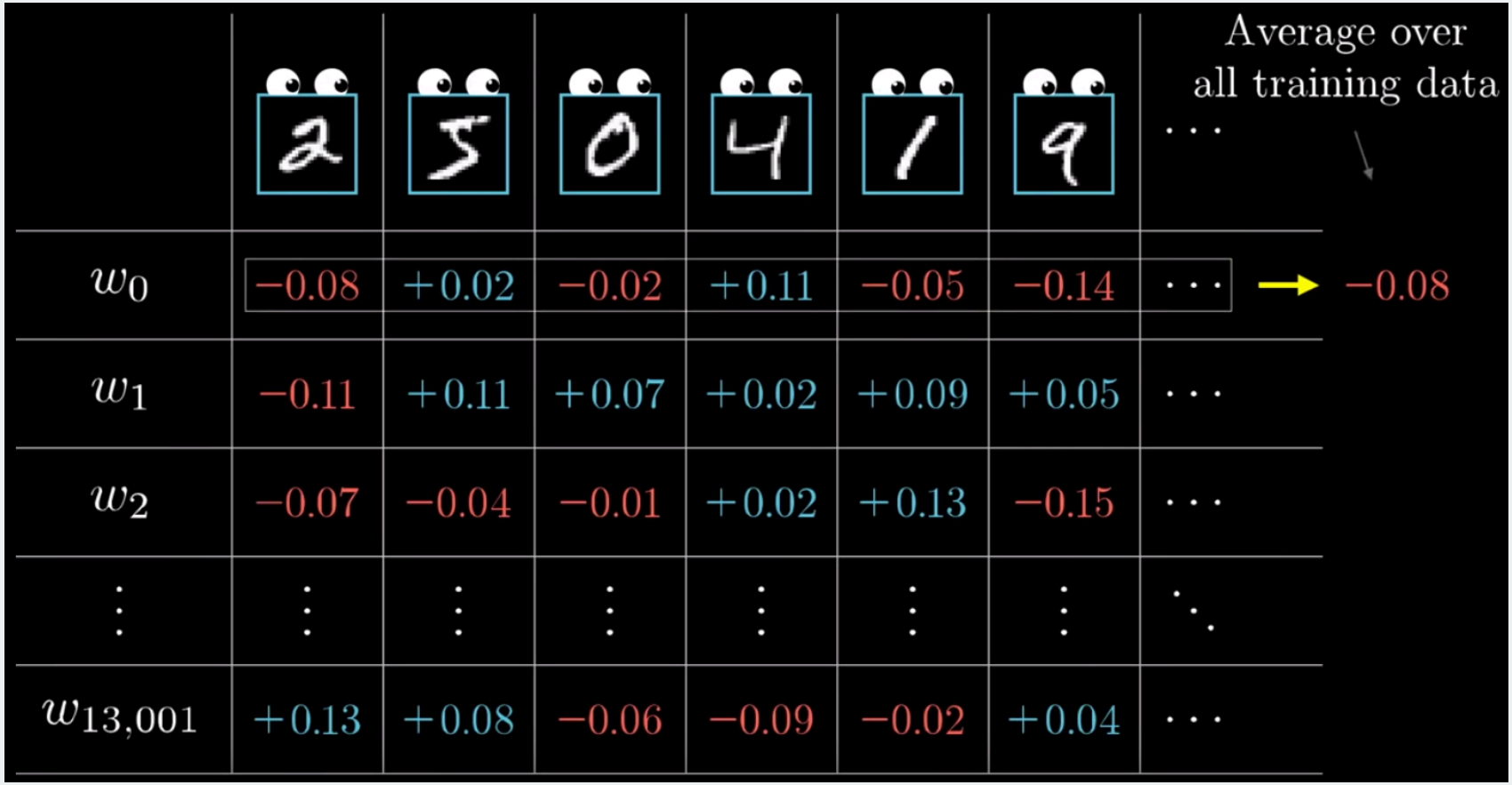
$$\mathbf{y} = \begin{bmatrix} 1 \end{bmatrix}$$

Backpropagation – on a neural network

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



Backpropagation – in the computer



https://www.youtube.com/watch?v=IIg3gGewQ5U