

# **Backpropagation**

**Instructor:** Mark Kramer

## **Part 1** **Backpropagation**

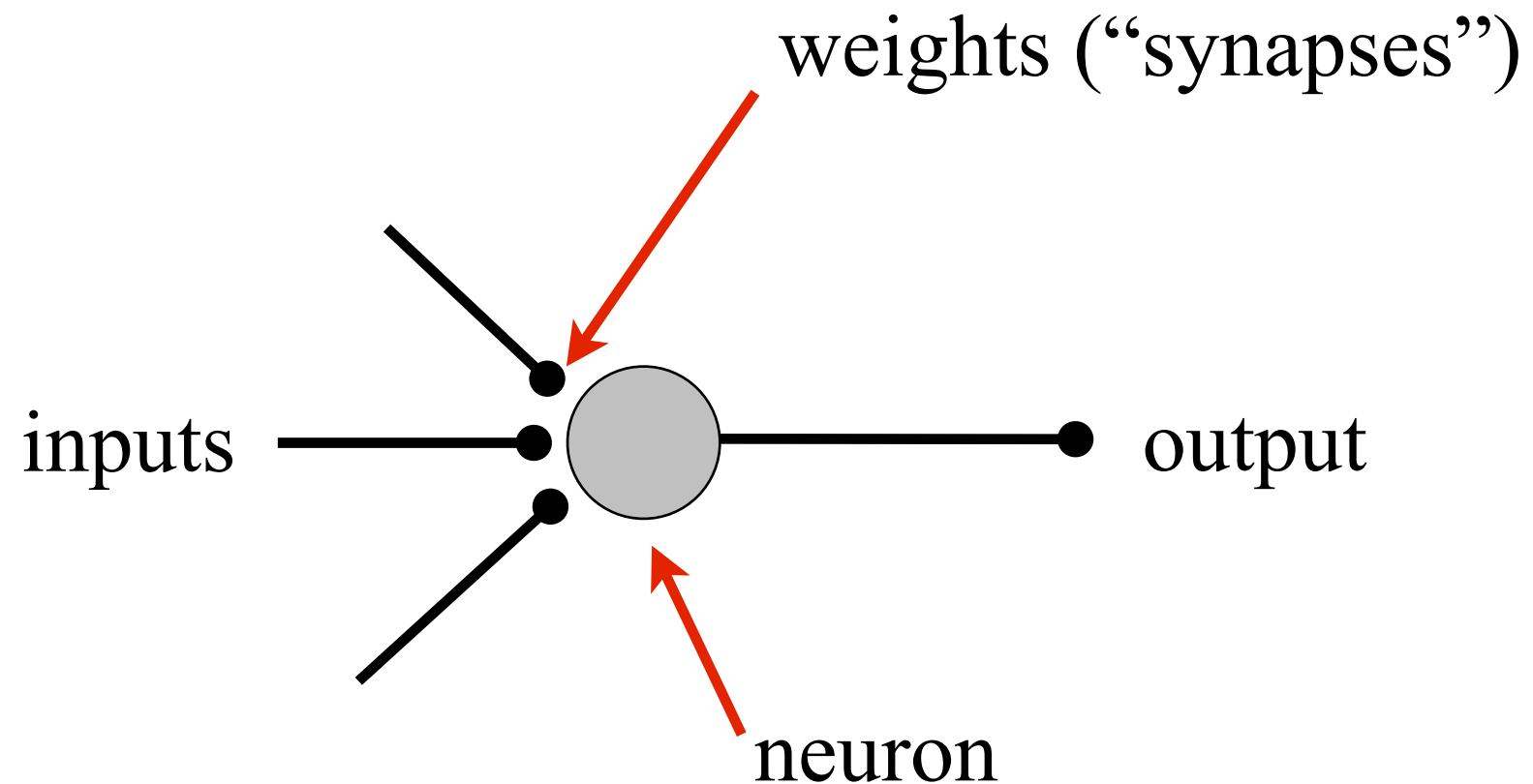
# Today

We'll study learning in a “simple” neural network:

- Backpropagation

# Remember, the Perceptron

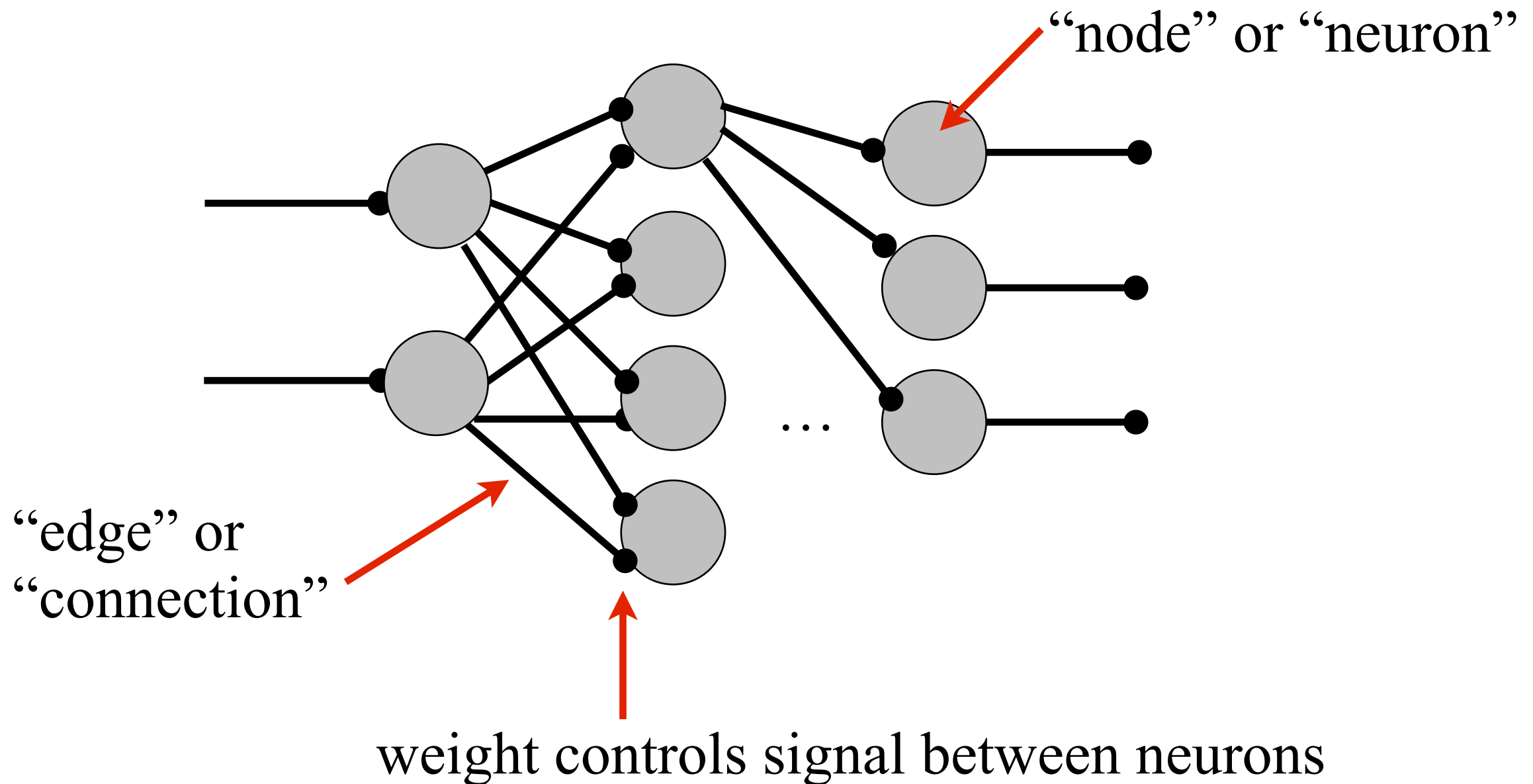
## Cartoon & Cast of Characters



Note: there's only one neuron.

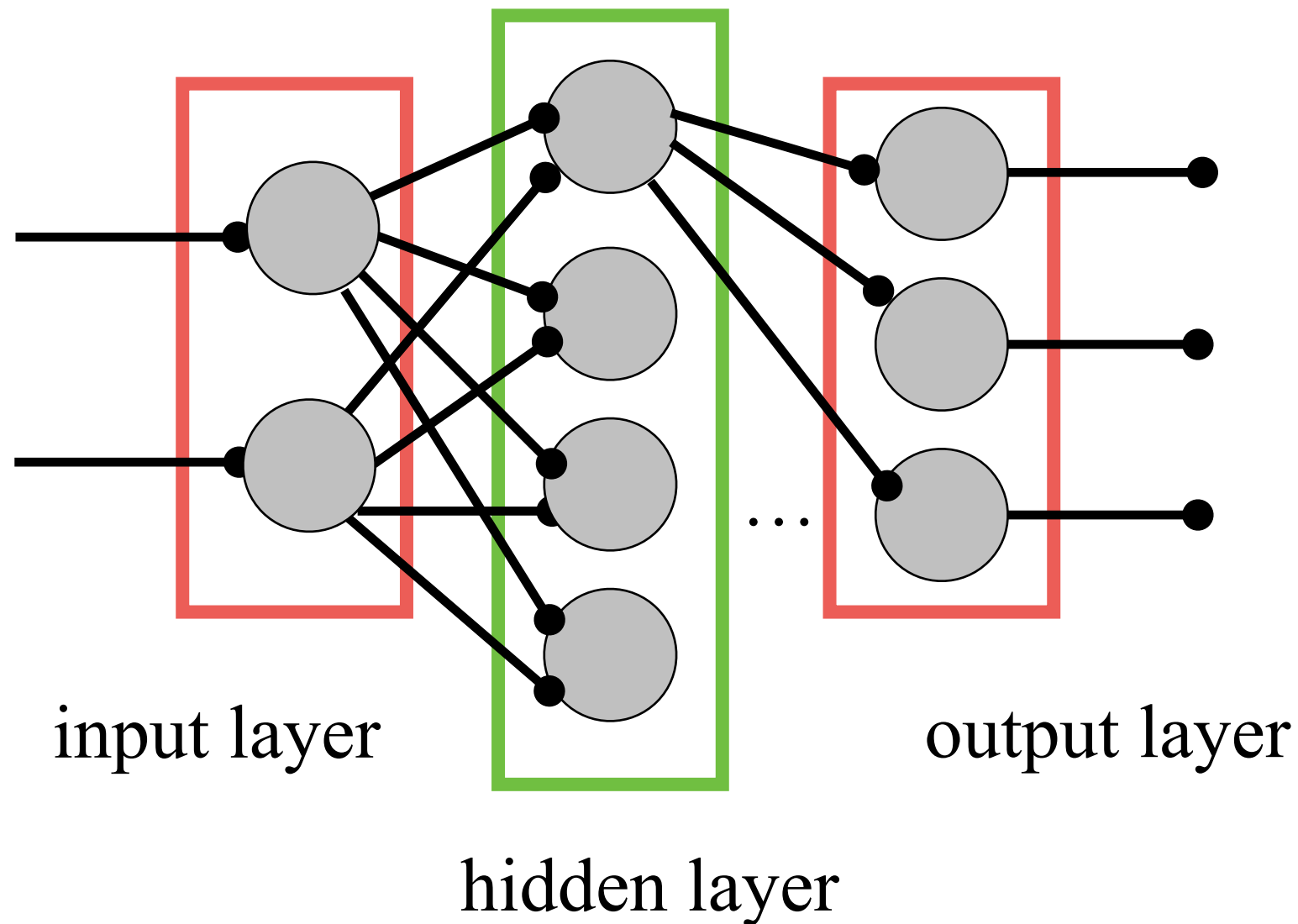
# Today, a neural network

Today, **neural networks**: collections of abstracted neurons connected to each other through weighted connections (simplified “synapses”).



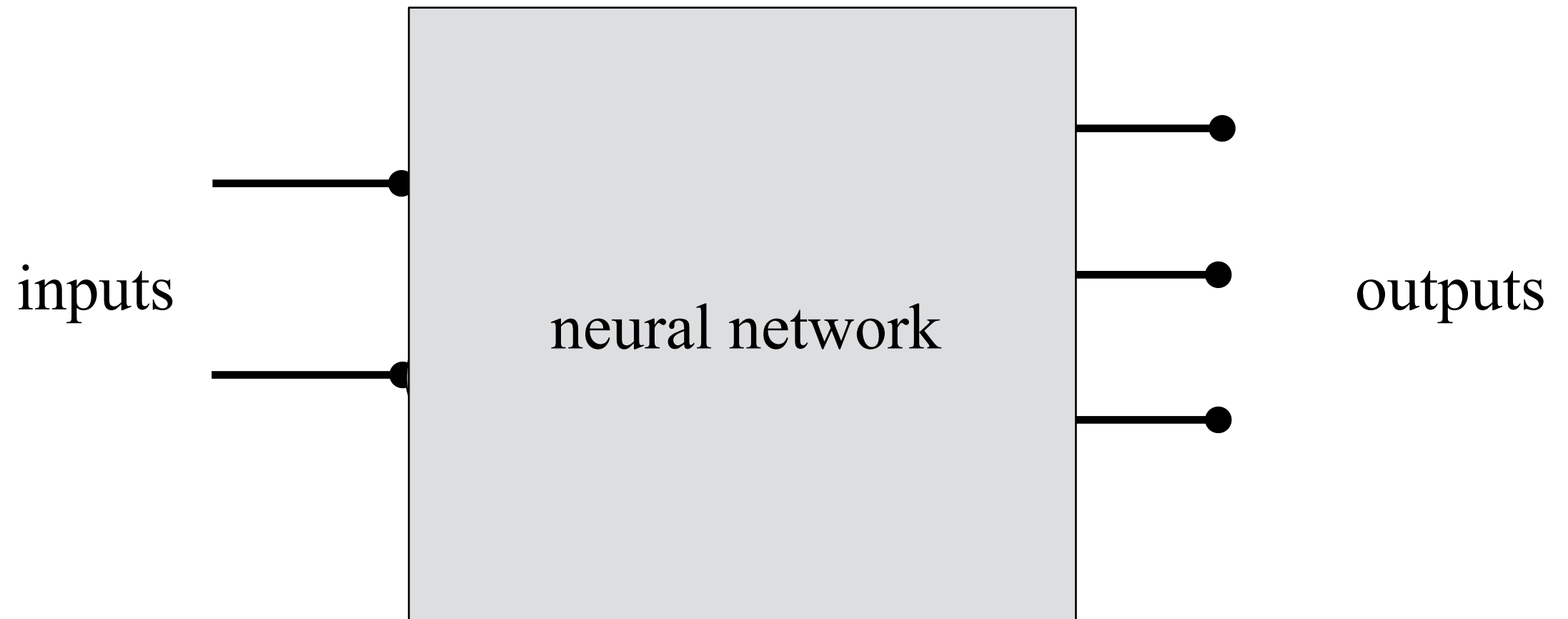
# Today, a neural network

Today, **neural networks**: collections of abstracted neurons connected to each other through weighted connections (simplified “synapses”).



# Information processed through the network

Abstractly:



Neural networks can exhibit rich behavior.

Cool example: [playground.tensorflow.org](https://playground.tensorflow.org)

# Neural networks can learn

Neural networks are:

- **adaptive**

- internal structure changes based on information flowing through the network.

- To do so, **adjust weights**.

- Idea:

- When network outputs are “good”, preserve the weights.
- When network output are “bad”, changes the weights.
  - When the network makes errors, adapt.

We trained a perceptron ...

Now, we'll train a neural network to do what we want ...

# Neural networks can learn

Some terminology:

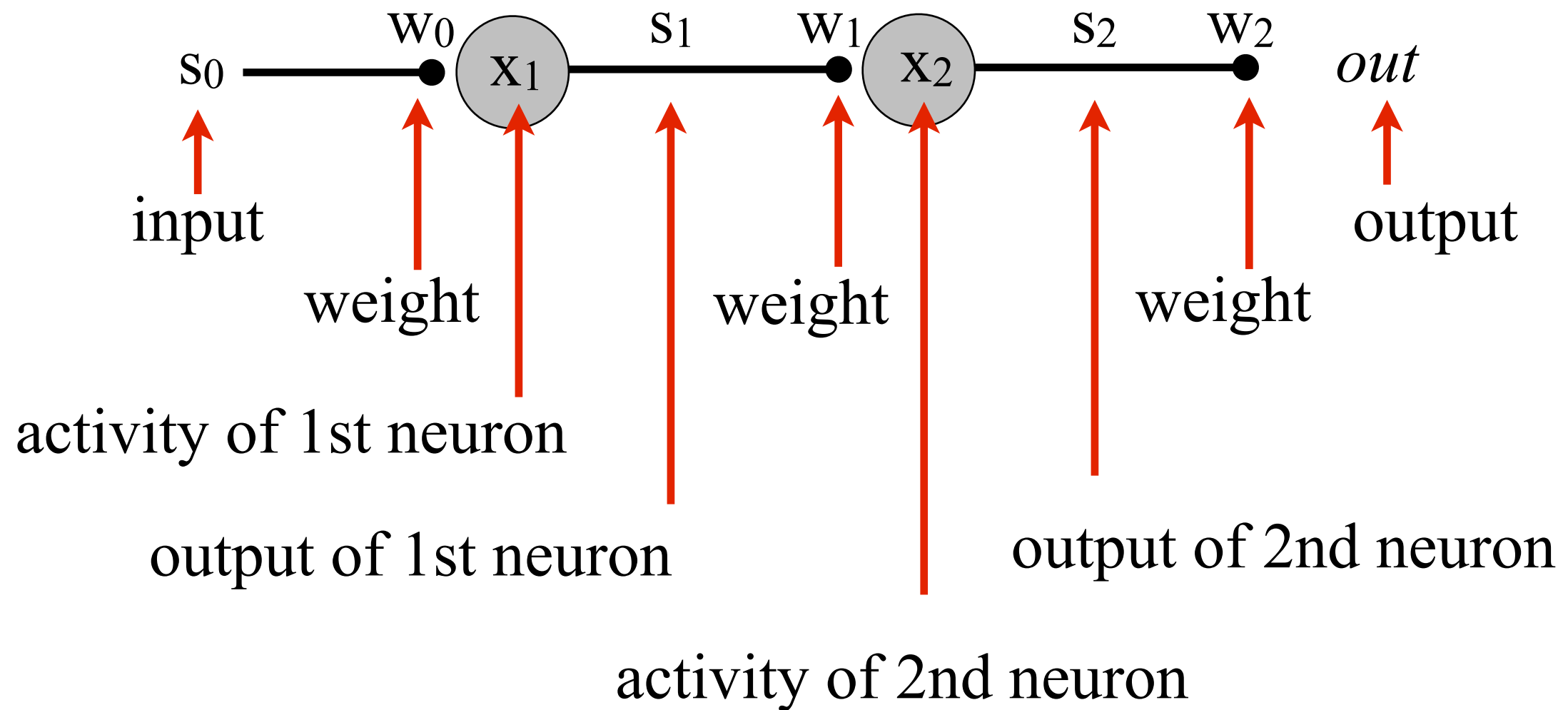
- “*training a neural network*”  
calibrate weights to get output we want.
- **Forward propagation**  
For a set of weights & input, calculate output.
- **Backpropagation**  
Determine error in output, and adjust weights to decrease error.

Let’s train a “simple” neural network to do our bidding ...



# A “simple” neural network

Start with perceptron ...      add a node ...      and label everything.



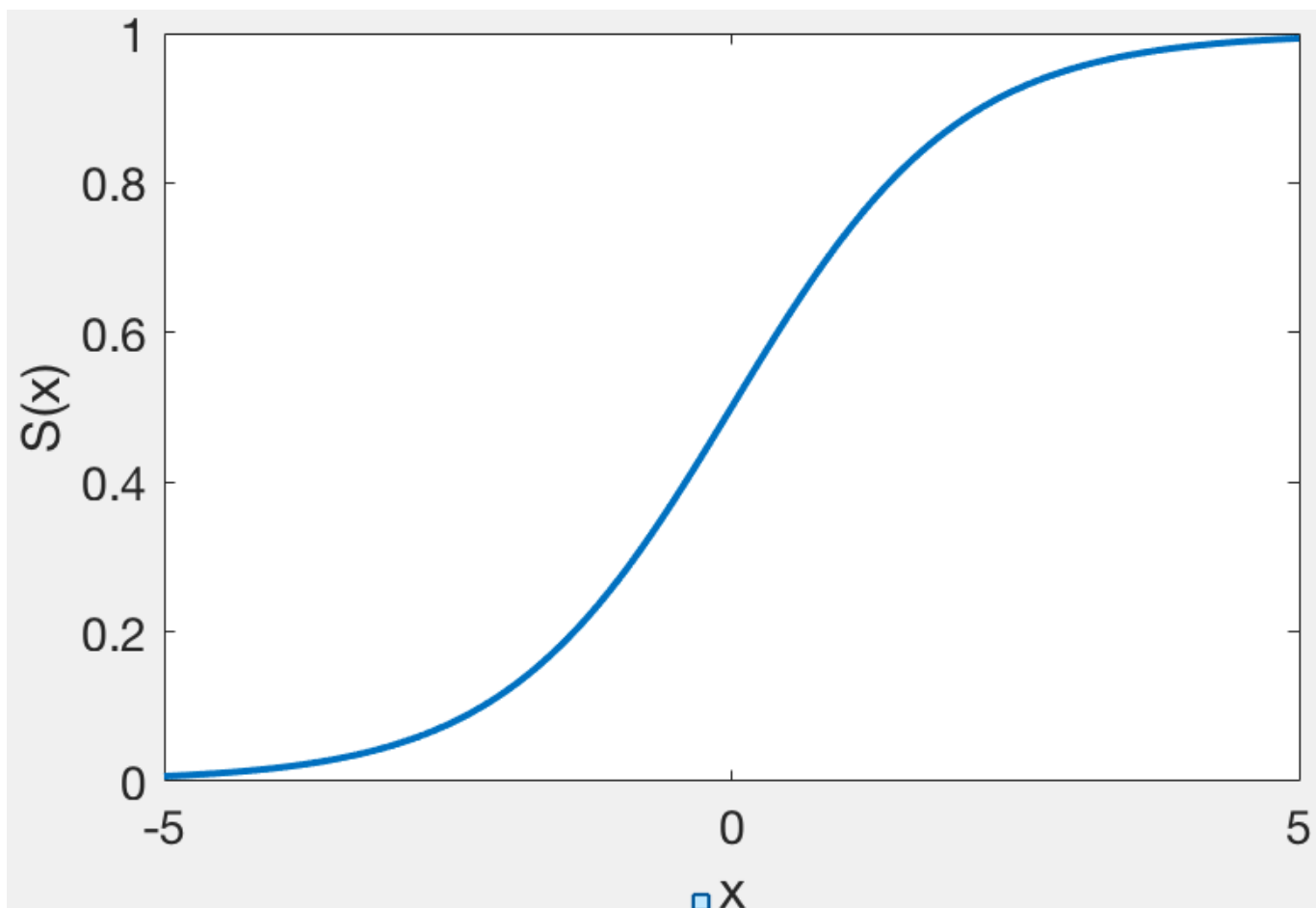
# Activation function

Remember, the activation function:

$x$  (activity)  $\xrightarrow{\text{activation function}}$  output

Here we'll use a **sigmoid** activation function:

$$S(x) = 1 / ( 1 + e^{-x} )$$



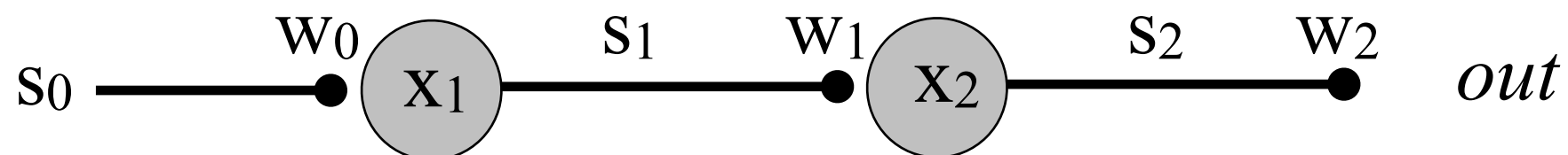
NOTE: It's like a  
“smoothed” binary  
threshold.

# The challenge

We're given a list of inputs ( $s_0$ ) and outputs ( $out$ ):

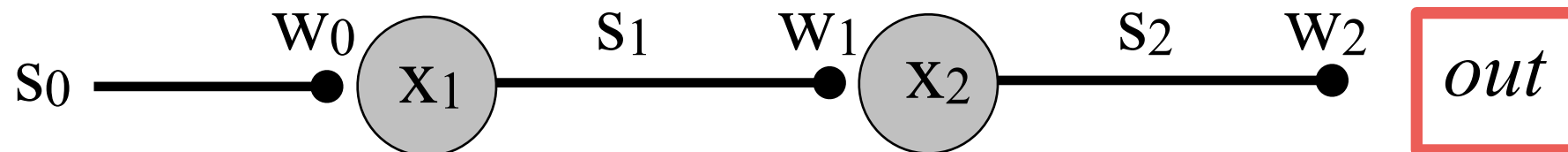
$s_0$	$out$
[ [ 0.32625098	0.71507894]
[-0.65595776	1.0065779 ]
[-0.9664146	1.09676408]
...	
[ 1.53148642	0.48558244]
[-1.07312405	1.12584863]
[ 1.32680081	0.51212226]]

**Q:** What weights ( $w_0$   $w_1$   $w_2$ ) produced these data?



# A “simple” neural network

We want our network to **learn** ...



so that when then input  $s_0=2$ ,

the input  $out=0.7$

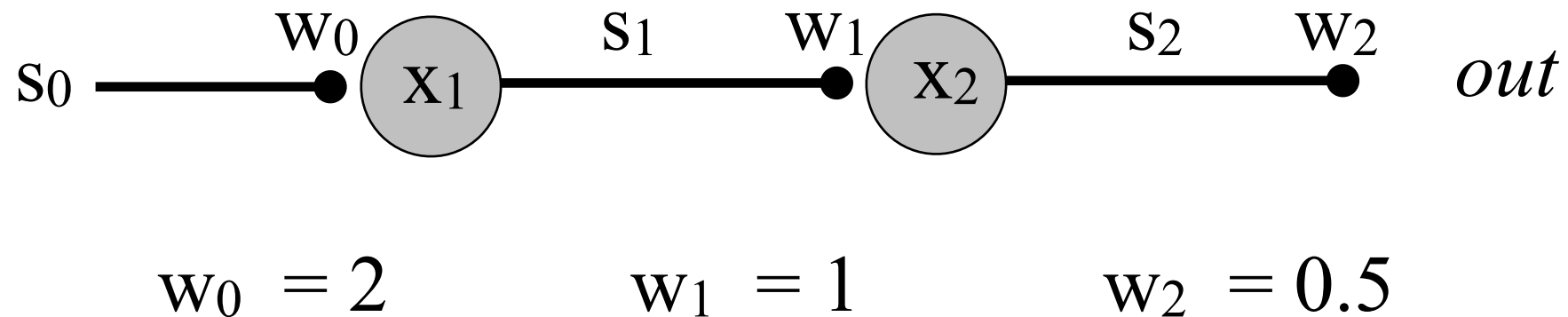
**Q:** How do we do it?

**A:** We need to choose the right weights:  $w_0$   $w_1$   $w_2$

So, how do we find the right weights?

# What are the right model weights?

Let's guess:

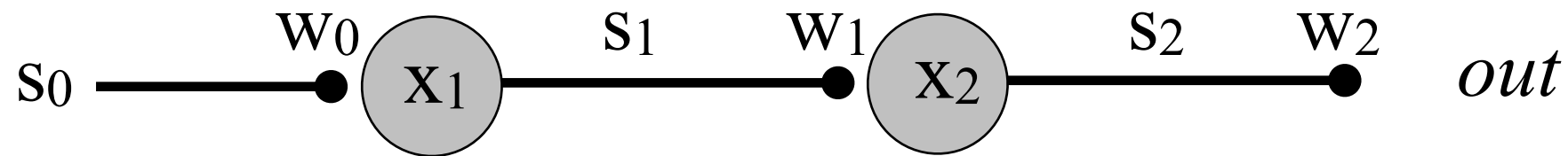


**Q:** How did I choose these?

**Q:** Do they work?

**A:** Let's check ...      **forward propagation**

# Forward propagation



$$s_0 = 2 \quad w_0 = 2 \quad w_1 = 1 \quad w_2 = 0.5$$

target:  
 $out = 0.7$

Let's do it.

$$x_1 = w_0 s_0 = 2 * 2 = 4$$

$$s_1 = S(x_1) = S(4) = 0.982$$

$$x_2 = w_1 s_1 = 1 * 0.982 = 0.982$$

$$s_2 = S(x_2) = S(0.982) = 0.7275$$

$$out = w_2 s_2 = 0.5 * 0.7275 = 0.3638$$

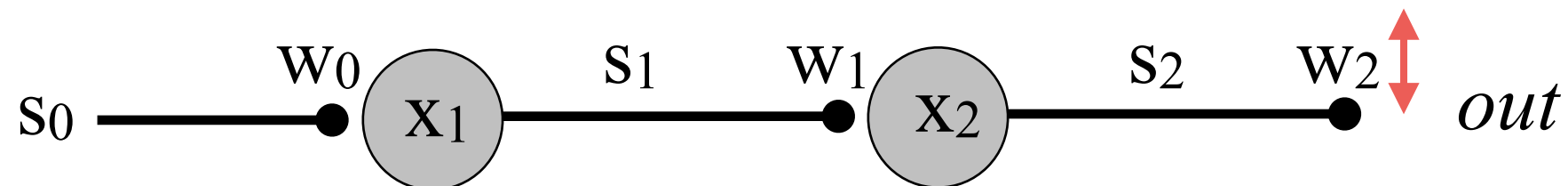
Match?

**NO**

# How does a change in weight $w_2$ impact output?

**Q:** So now what?

(intermediate) Goal: get output (*out*) closer to target (0.7)



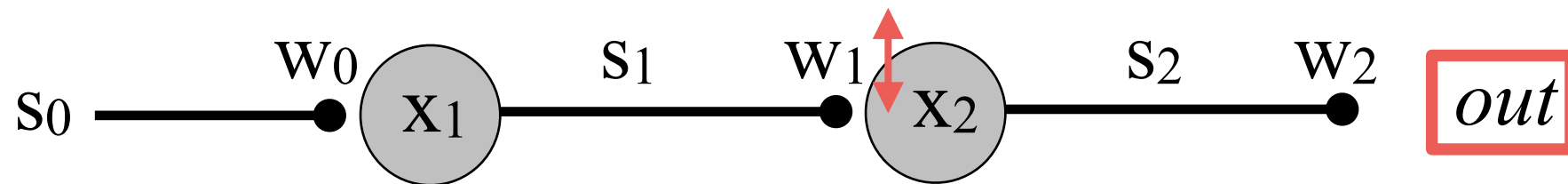
**Q:** How does a change in weight  $w_2$  impact the output?

Idea: wiggle  $w_2$       how does *out* change?       $out = s_2 w_2$

Mathematically, 
$$\frac{d \text{ out}}{d w_2} = \frac{d (s_2 w_2)}{d w_2} = s_2$$

# How does a change in weight $w_1$ impact output?

Let's keep going, working backwards.



**Q:** How does a change in weight  $w_1$  impact the output?

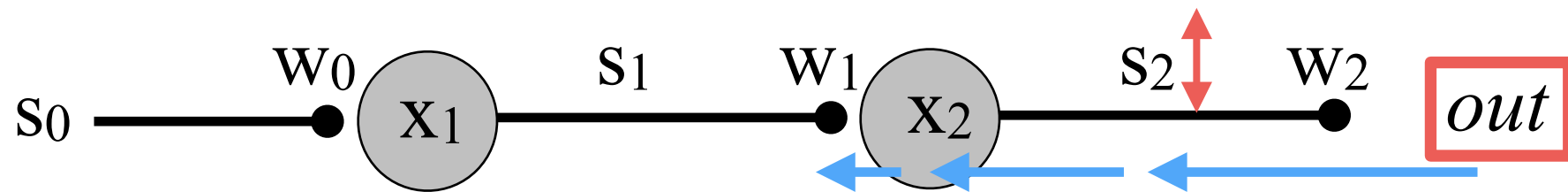
Idea: wiggle  $w_1$       how does *out* change?

Mathematically,  $\frac{d \text{out}}{d w_1} =$  Hmm ...

*out* does not depend directly on  $w_1$



# How does a change in weight $w_1$ impact output?



$out$  does depend on  $s_2$       and  $s_2$  depend on  $x_2$       and  $x_2$  depend on  $w_1$

Mathematically ... the **chain rule**

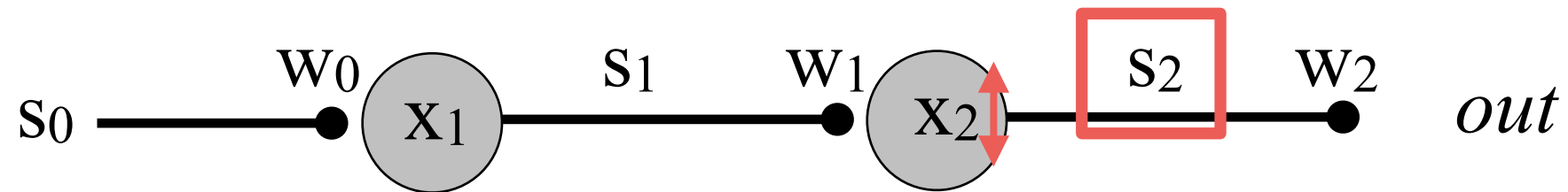
$$\frac{d \text{ out}}{d w_1} = \boxed{\frac{d \text{ out}}{d s_2}} \frac{d s_2}{d x_2} \frac{d x_2}{d w_1}$$

wiggle  $s_2$  and  $out$  changes

Remember:  $out = s_2 w_2$

$$\frac{d \text{ out}}{d s_2} = \frac{d (s_2 w_2)}{d s_2} \boxed{= w_2}$$

# How does a change in weight $w_1$ impact output?



Continue the **chain rule**

$$\frac{d \text{out}}{d w_1} = \frac{d \text{out}}{d s_2} \boxed{\frac{d s_2}{d x_2}} \frac{d x_2}{d w_1}$$

wiggle  $x_2$  and  $s_2$  changes

$$s_2 = S(x_2) = \left( \frac{1}{1 + e^{-x_2}} \right)$$

$$\text{so } \frac{d s_2}{d x_2} = \frac{d}{d x_2} \left( \frac{1}{1 + e^{-x_2}} \right)$$

# A complicated derivative

We need to compute:

$$\frac{d}{dx_2} \left( \frac{1}{1 + e^{-x_2}} \right) = \text{Hmm ... Quotient Rule}$$

$$= \frac{(1 + e^{-x_2}) \frac{d(1)}{dx_2} - (1) \frac{d(1 + e^{-x_2})}{dx_2}}{(1 + e^{-x_2})^2}$$

**Q:** What is  $\frac{d(1 + e^{-x_2})}{dx_2}$  ?

$$= \frac{d(1)}{dx_2} + \frac{d(e^{-x_2})}{dx_2} = (e^{-x_2}) \frac{d(-x_2)}{dx_2} = -e^{-x_2}$$

## A complicated derivative

So,

$$\begin{aligned}\frac{d}{dx_2} \left( \frac{1}{1 + e^{-x_2}} \right) &= \frac{0 - (1)(-e^{-x_2})}{(1 + e^{-x_2})^2} \\ &= \frac{e^{-x_2}}{(1 + e^{-x_2})^2}\end{aligned}$$

**Q:** Can we simplify this expression?

**A:** Yes, but requires faith ...

- Split up denominator: 
$$\frac{e^{-x_2}}{(1 + e^{-x_2})^2} = \left( \frac{1}{1 + e^{-x_2}} \right) \left( \frac{e^{-x_2}}{1 + e^{-x_2}} \right)$$

# A complicated derivative

- Add 0 to the second term:

$$\left(\frac{1}{1+e^{-x_2}}\right)\left(\frac{e^{-x_2}}{1+e^{-x_2}}\right) = \left(\frac{1}{1+e^{-x_2}}\right)\left(\frac{\boxed{1} + e^{-x_2}\boxed{-1}}{1+e^{-x_2}}\right)$$

Sum is 0

**Q:** Why?

**A:** Let's organize terms ...

$$= \underbrace{\left(\frac{1}{1+e^{-x_2}}\right)}_{= s_2} \left( \underbrace{\left(\frac{1+e^{-x_2}}{1+e^{-x_2}}\right)}_{= 1} - \underbrace{\left(\frac{1}{1+e^{-x_2}}\right)}_{= s_2} \right)$$

$$= (s_2) (1 - s_2)$$

Remember:

$$s_2 = \left(\frac{1}{1+e^{-x_2}}\right)$$

## A complicated derivative

So, for our chain rule calculation:

$$\frac{d \text{ out}}{d w_1} = \frac{d \text{ out}}{d s_2} \boxed{\frac{d s_2}{d x_2}} \frac{d x_2}{d w_1}$$

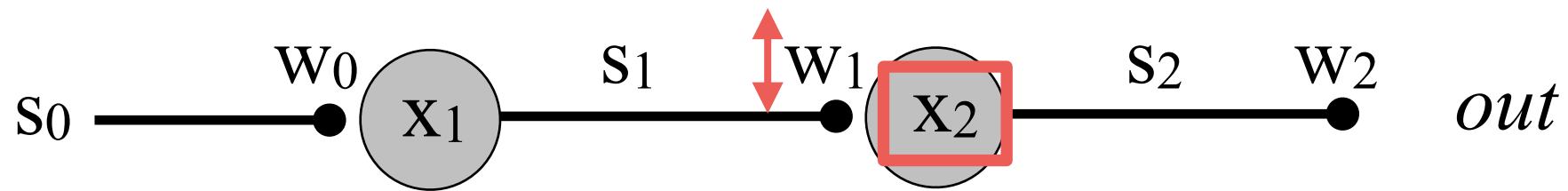
And we found:

$$\frac{d s_2}{d x_2} = \frac{d}{dx_2} \left( \boxed{\frac{1}{1 + e^{-x_2}}} \right) = \dots \text{many steps} \dots \boxed{= s_2 (1 - s_2)}$$

$\boxed{\frac{1}{1 + e^{-x_2}}} = s_2$

To complete the chain rule, one more derivative ...

# How does a change in weight $w_1$ impact output?



Continue the **chain rule**:

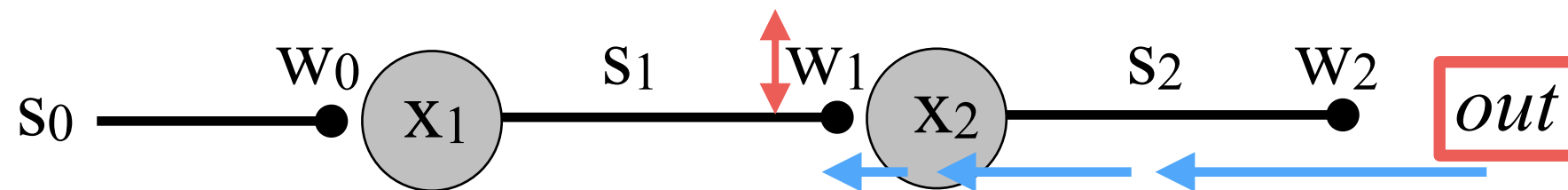
$$\frac{d \text{out}}{d w_1} = \frac{d \text{out}}{d s_2} \frac{d s_2}{d x_2} \boxed{\frac{d x_2}{d w_1}}$$

wiggle  $w_1$  and  $x_2$  changes

Remember:  $x_2 = s_1 w_1$

$$\frac{d x_2}{d w_1} = \frac{d (s_1 w_1)}{d w_1} = \boxed{s_1}$$

Back to our original question:



**Q:** How does a change in weight  $w_1$  impact the output?

Mathematically ... the **chain rule**

$$\frac{d \text{out}}{d w_1} = \frac{d \text{out}}{d s_2} \frac{d s_2}{d x_2} \frac{d x_2}{d w_1}$$

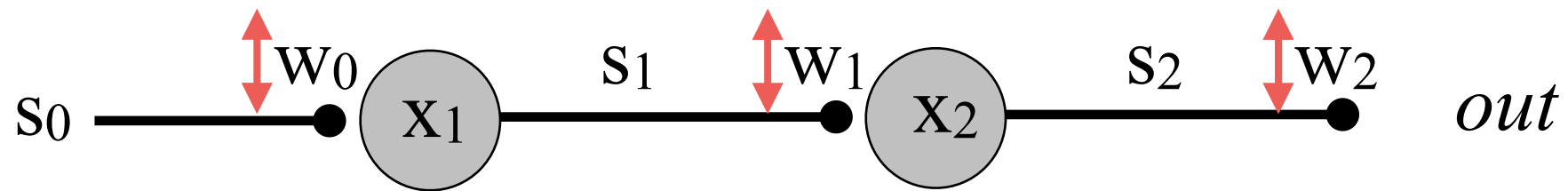
$$\frac{d \text{out}}{d w_1} = w_2 \quad s_2 (1 - s_2) \quad s_1$$

*Slide 16    Slide 21    Slide 22*



# How does a change in weight $w_0$ impact output?

We're almost there ...



$$\frac{d \text{out}}{d w_2} = s_2$$

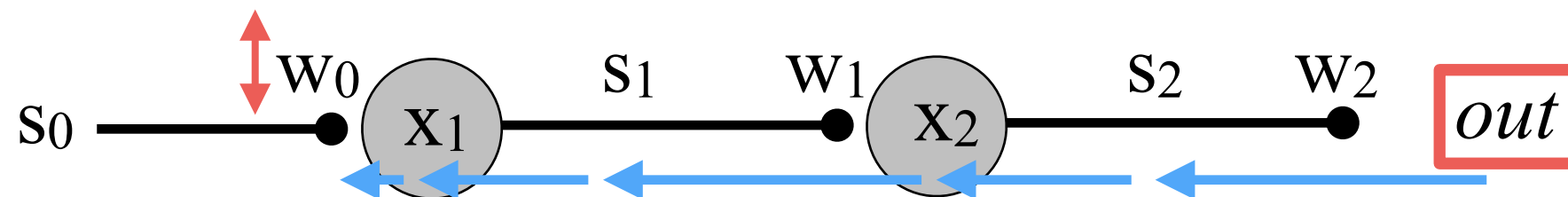
$$\frac{d \text{out}}{d w_1} = w_2 s_2 (1 - s_2) s_1$$

**Q:** How does a change in weight  $w_0$  impact the output?

**A:** Chain rule ...

# How does a change in weight $w_0$ impact output?

**Q:** How does a change in weight  $w_0$  impact the output?



$$\frac{d \text{out}}{d w_0} = \frac{d \text{out}}{d s_2} \frac{d s_2}{d x_2} \boxed{\frac{d x_2}{d s_1} \frac{d s_1}{d x_1} \frac{d x_1}{d w_0}} \quad \text{Ugh ...}$$

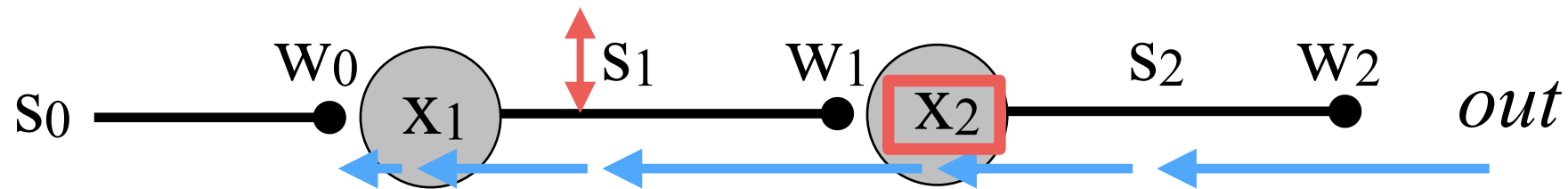
Luckily, we've already calculated two of these.

$$w_2 \quad s_2 (1 - s_2)$$

Let's compute the last 3 terms ...

# How does a change in weight $w_0$ impact output?

3rd term:



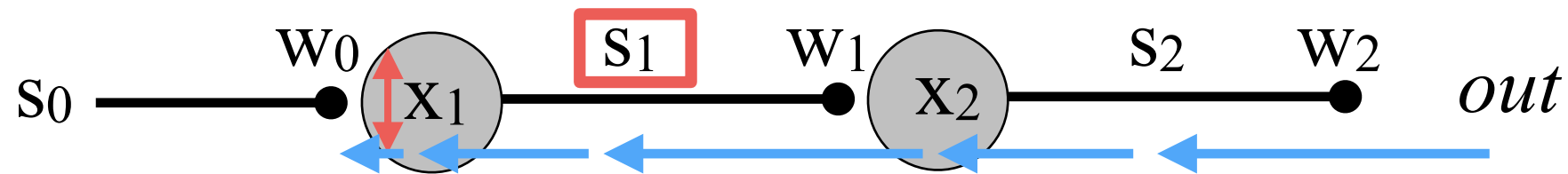
$$\frac{d \text{out}}{d w_0} = \frac{d \text{out}}{d s_2} \frac{d s_2}{d x_2} \boxed{\frac{d x_2}{d s_1}} \frac{d s_1}{d x_1} \frac{d x_1}{d w_0}$$

Remember:  $x_2 = s_1 w_1$

$$\frac{d x_2}{d s_1} = \frac{d (s_1 w_1)}{d s_1} = \boxed{w_1}$$

# How does a change in weight $w_0$ impact output?

4th term:



$$\frac{d \text{out}}{d w_0} = \frac{d \text{out}}{d s_2} \frac{d s_2}{d x_2} \frac{d x_2}{d s_1} \boxed{\frac{d s_1}{d x_1}} \frac{d x_1}{d w_0}$$

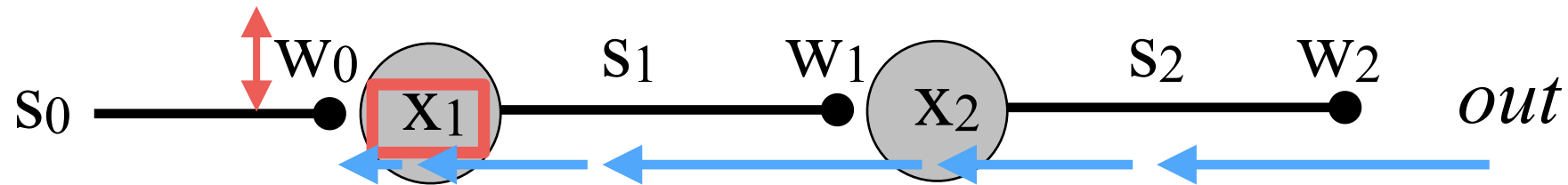
We found earlier that:

$$\frac{d s_2}{d x_2} = s_2 (1 - s_2) \quad \dots \text{so} \dots \quad \frac{d s_1}{d x_1} = \boxed{s_1 (1 - s_1)}$$

This involved many steps!

# How does a change in weight $w_0$ impact output?

5th term:



$$\frac{d \text{out}}{d w_0} = \frac{d \text{out}}{d s_2} \frac{d s_2}{d x_2} \frac{d x_2}{d s_1} \frac{d s_1}{d x_1} \boxed{\frac{d x_1}{d w_0}}$$

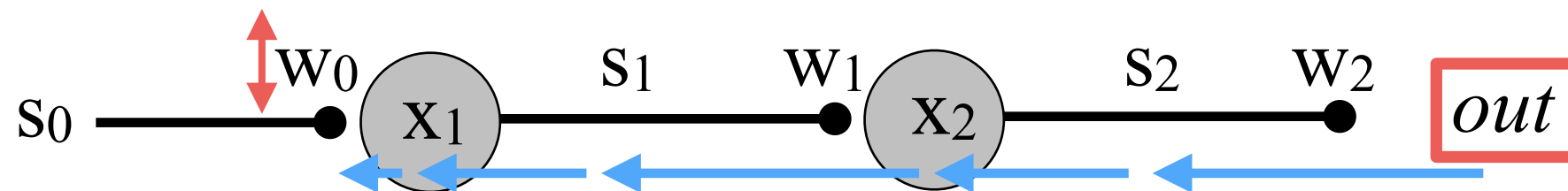
Remember:  $x_1 = s_0 w_0$

$$\frac{d x_1}{d w_0} = \frac{d (s_0 w_0)}{d w_0} = \boxed{s_0}$$

# How does a change in weight $w_0$ impact output?

We now have the pieces to answer:

**Q:** How does a change in weight  $w_0$  impact the output?



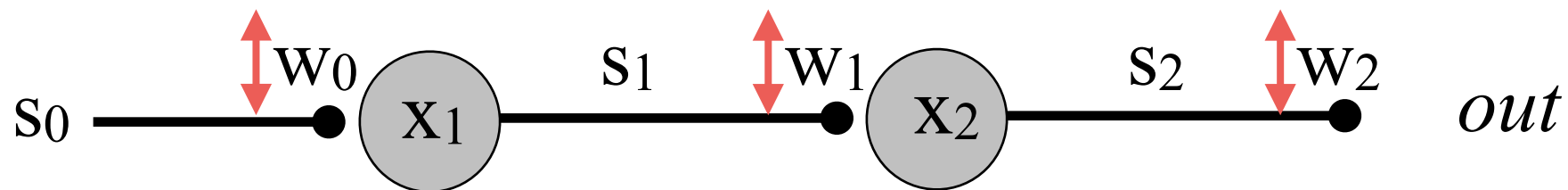
$$\frac{d \text{out}}{d w_0} = \frac{d \text{out}}{d s_2} \frac{d s_2}{d x_2} \frac{d x_2}{d s_1} \frac{d s_1}{d x_1} \frac{d x_1}{d w_0}$$
$$= w_2 \quad s_2 (1 - s_2) \quad w_1 \quad s_1 (1 - s_1) \quad s_0$$

*Slide 16*    *Slide 21*    *Slide 26*    *Slide 27*    *Slide 28*

# How does a change in weight impact output?

To summarize:

- We've found how changes in model weights impact output.



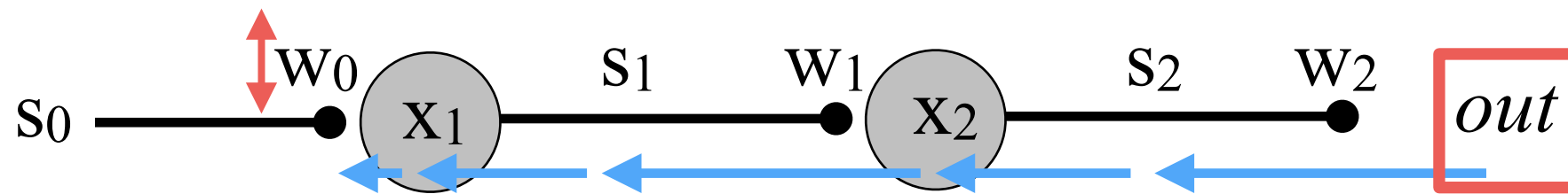
$$\frac{d \text{out}}{d w_2} = s_2$$

$$\frac{d \text{out}}{d w_1} = w_2 s_2 (1 - s_2) s_1$$

$$\frac{d \text{out}}{d w_0} = w_2 s_2 (1 - s_2) w_1 s_1 (1 - s_1) s_0$$

So, how does a change in weight impact output? **backpropagation!**

# How does a change in weight impact output?



## Backpropagation:

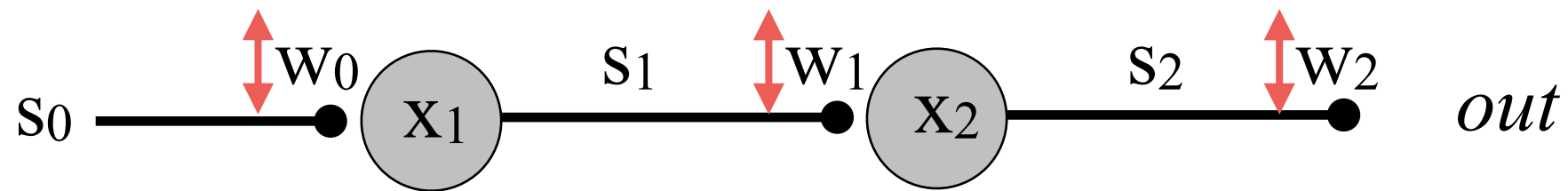
Work “**backwards**” from output to weight, computing derivatives along the way

**Q:** How do these derivatives help us update weights and obtain desired output?



# Define our goal

We want:



$$out = target$$

rearrange  $out - target = 0$  **GOAL**

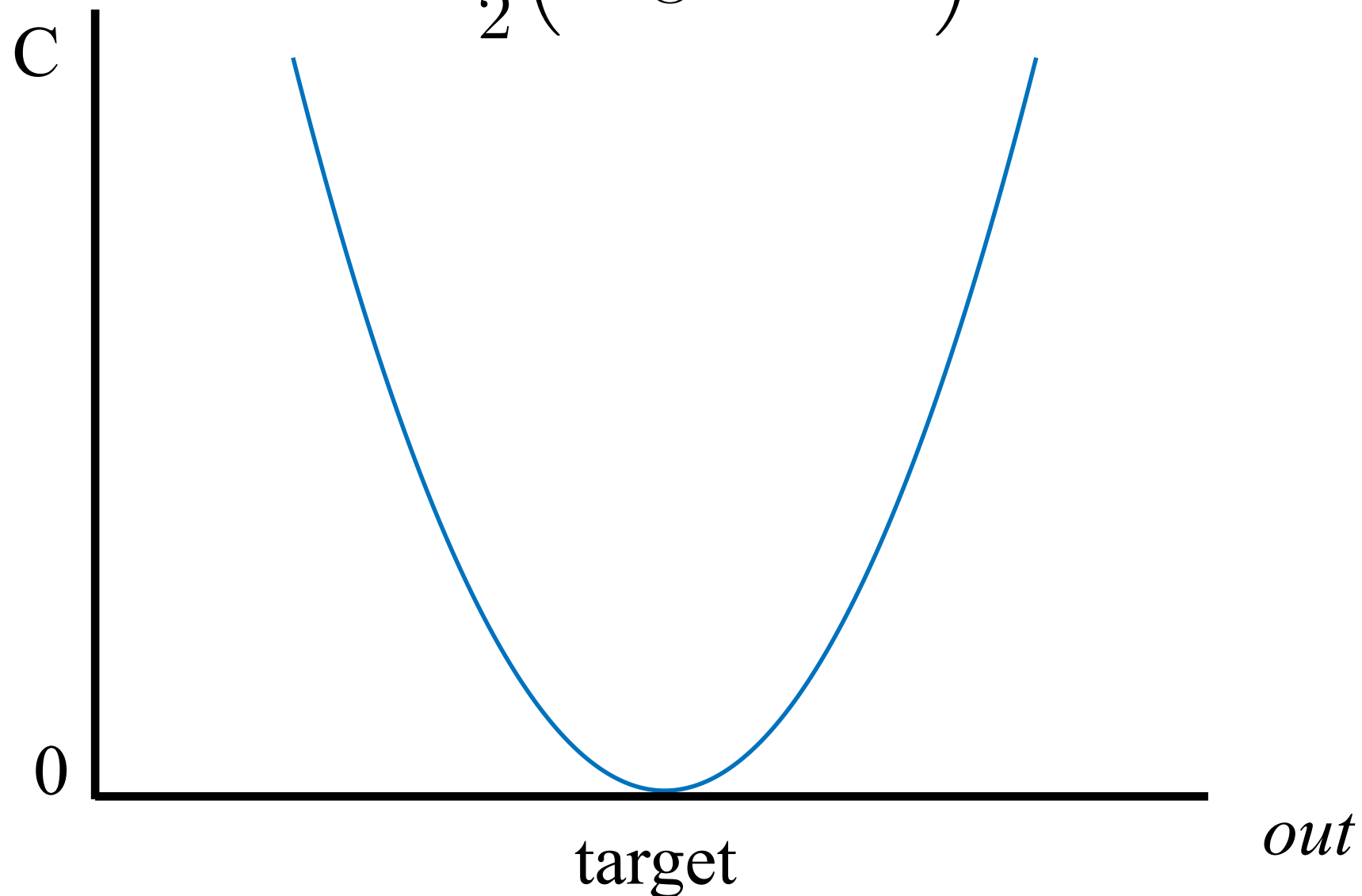
Let's use this to define a **cost function** ...

# Create a cost function

Define the cost function:

$$C = \frac{1}{2} (\text{target} - out)^2$$

Plot it:



**Q:** Where is the cost zero?

**A:** When  $out = \text{target}$ .

# Create a cost function

**Q:** Why this cost function?

$$C = \frac{1}{2} \left( \text{target} - out \right)^2$$

- Minimum (the “lowest cost”) when  $out = \text{target}$ .
- It’s convenient (a quadratic).
- It steadily increases as  $out$  deviates from target.
- It’s “easy” to compute derivatives.

# Create a cost function

**Q:** How does the cost function change due to changes in *out*?

**A:** We need to compute a derivative ...

$$\frac{dC}{dout} = \frac{d}{out} \left[ \frac{1}{2} \left( \text{target} - out \right)^2 \right] = ?$$

# Chain rule ...

$$\frac{dC}{dout} = 2 \cdot \frac{1}{2} (\text{target} - out) \cdot \left( \frac{d(-out)}{dout} \right)$$

$$\frac{dC}{dout} = -(\text{target} - out)$$

$$\frac{dC}{dout} = out - target$$

# Create a cost function

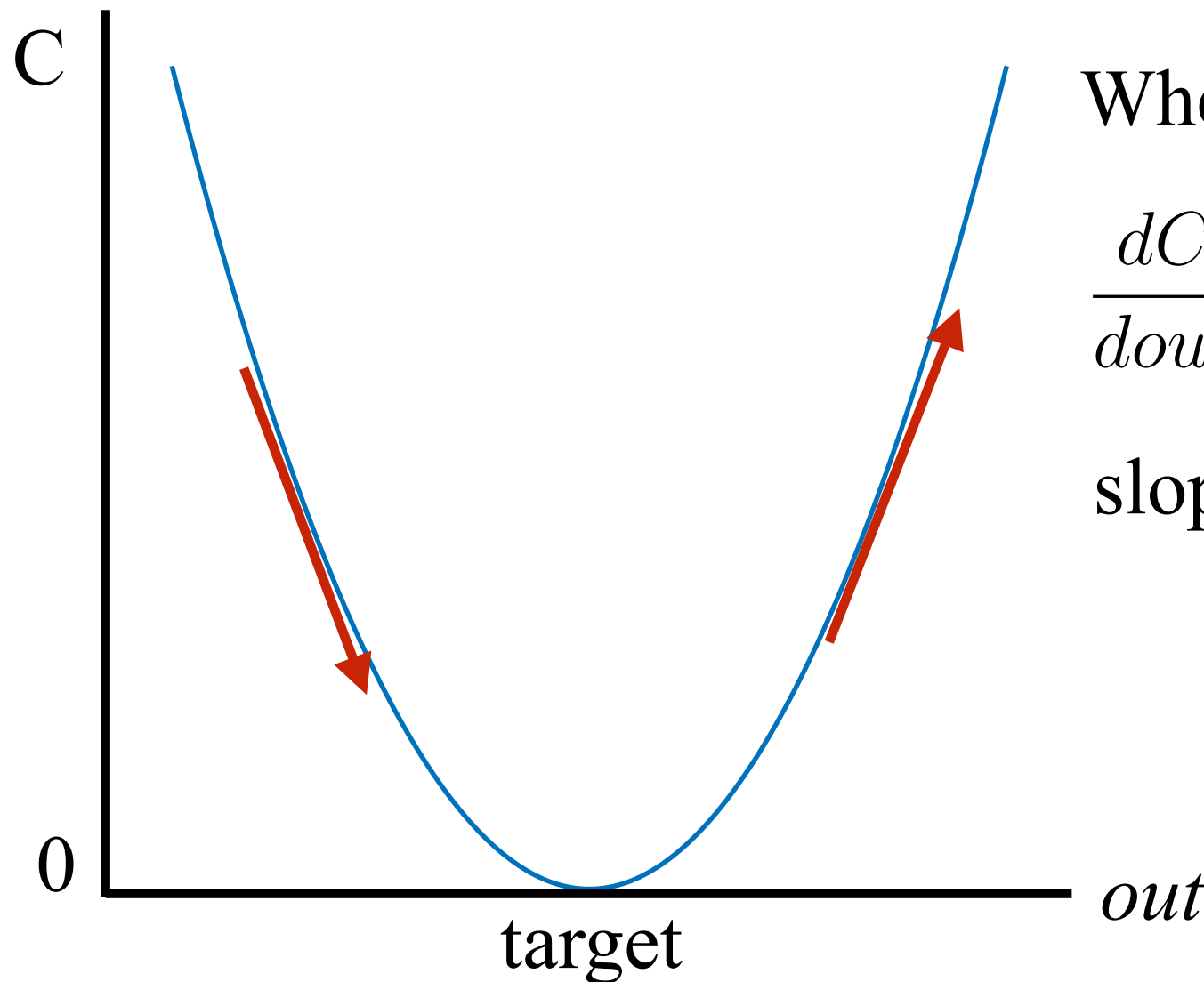
**Q:** Does this derivative make sense?

$$\frac{dC}{dout} = out - target$$

When  $out < target$

$$\frac{dC}{dout} < 0$$

slope  $< 0$



When  $out > target$

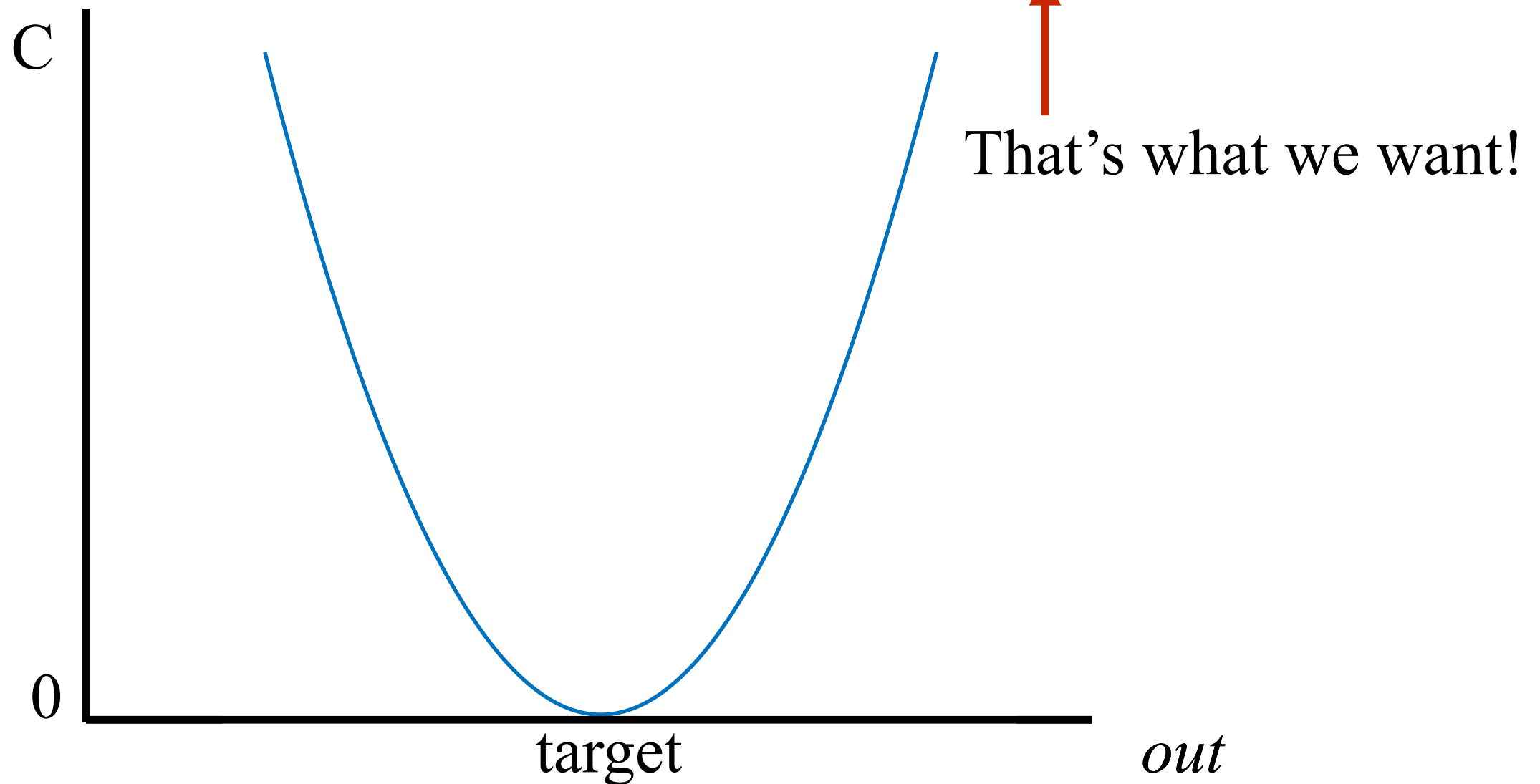
$$\frac{dC}{dout} > 0$$

slope  $> 0$

# Create a cost function

Now, our goal: Choose weights to minimize the cost function

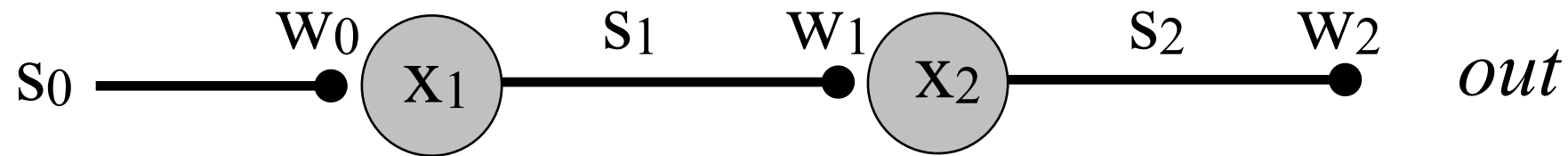
Remember, cost minimized when  $out = target$



Here, we plot  $C$  versus  $out$ . But  $out$  depends on weights ...

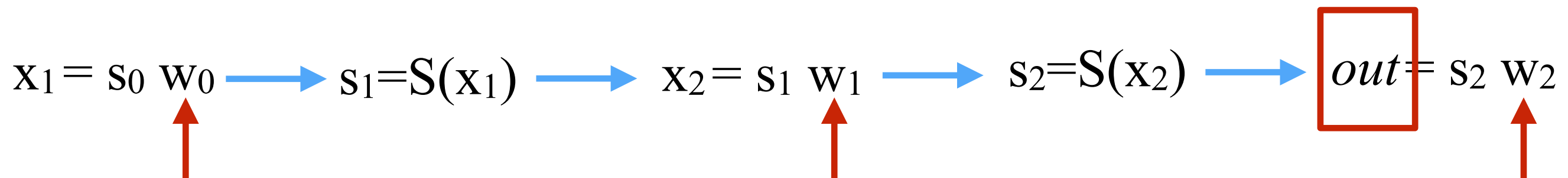
# Create a cost function

**Q:** How does *out* depend on weights?



**A:** It's complicated.

Consider feedforward solution ...

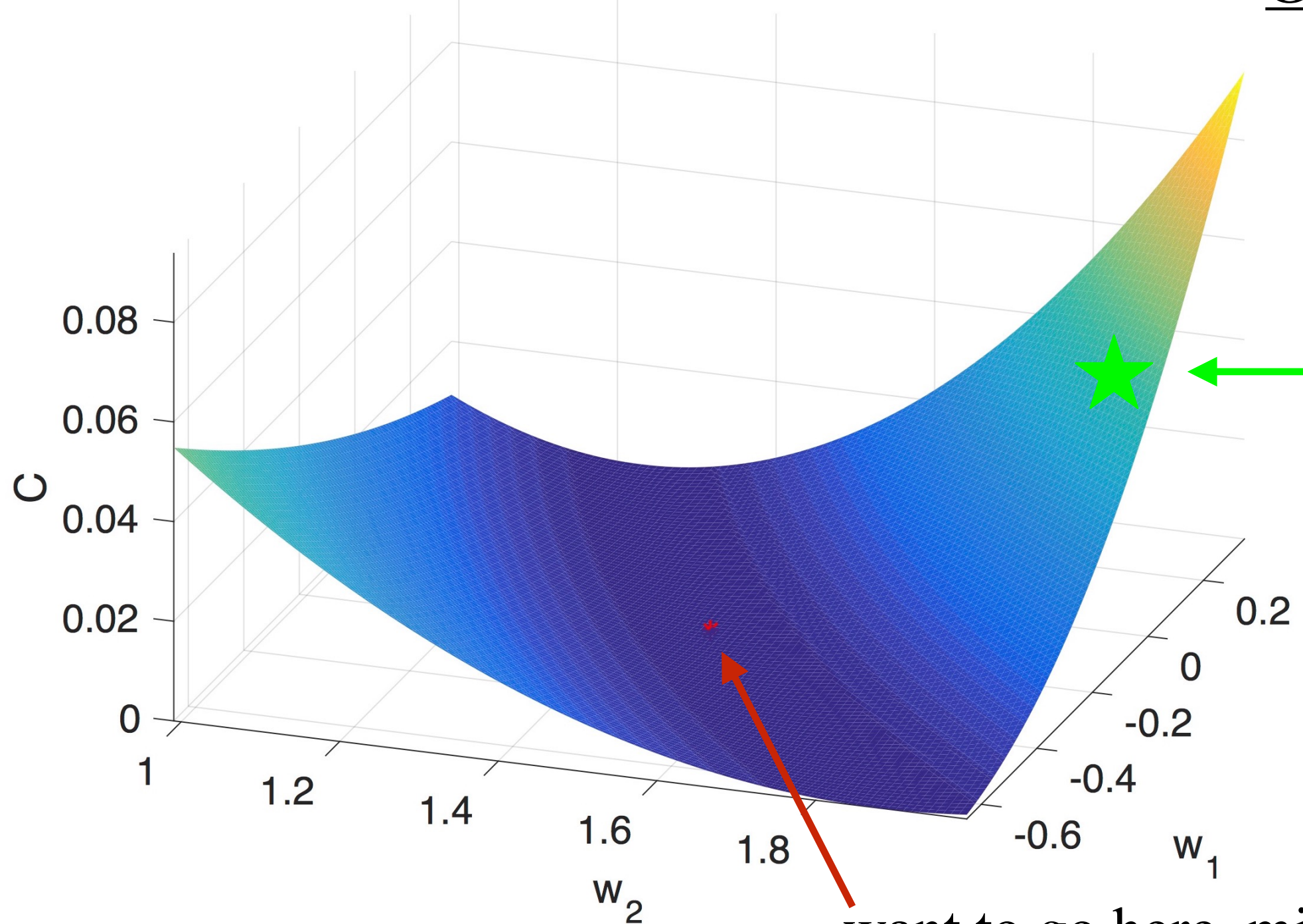


If *out* depends on weights, so does the cost ...  $C(w_0, w_1, w_2) = ?$

# Create a cost function

Plot cost  $C$  versus  $w_1$  and  $w_2$ :

Goal: minimize cost.



But, we have  
our initial,  
randomly  
chosen weights

want to go here, minimize cost.

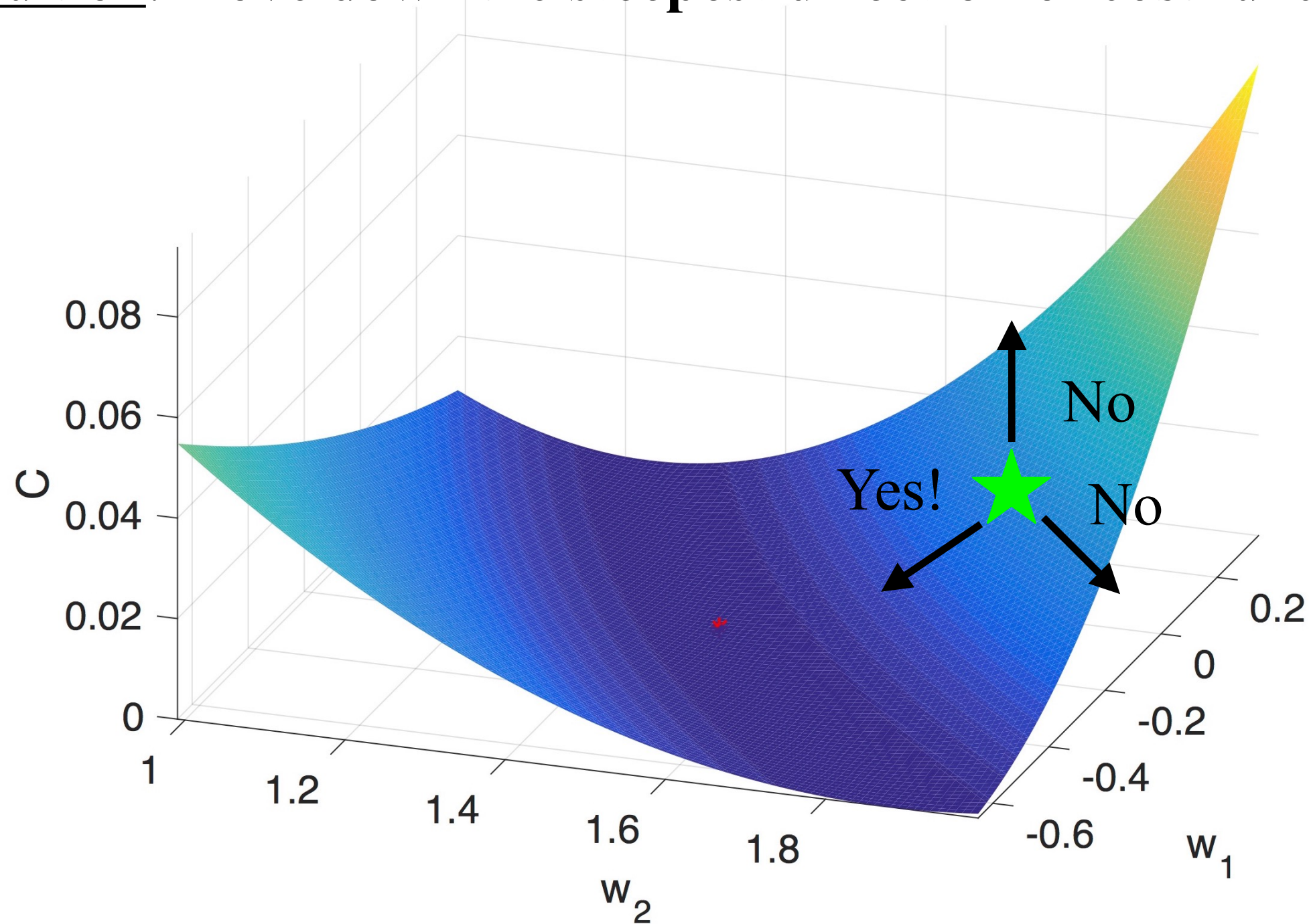
**Q:** How do we adjust weights to reach the minimum cost?

**A:** Move “downhill” ...



# Follow the cost function

Intuition: move down the **steepest direction** of cost function



Imagine placing a marble ... where does it roll? To the minimum.

**Q:** How do we find the steepest direction? **A:** Compute the gradient

# Follow the cost function

Gradient of the cost function.

How  $C$  changes due to small changes in  $w_0, w_1, w_2$ .

We need to compute:  $\frac{dC}{dw_0}$        $\frac{dC}{dw_1}$        $\frac{dC}{dw_2}$

Then, update the weights in steps proportional to the negative gradient.

The diagram illustrates the weight update step for  $w_1$ . It shows three update equations:  $w_0 \leftarrow w_0 - \alpha \frac{dC}{dw_0}$ ,  $w_1 \leftarrow w_1 - \alpha \frac{dC}{dw_1}$ , and  $w_2 \leftarrow w_2 - \alpha \frac{dC}{dw_2}$ . The term  $\alpha \frac{dC}{dw_1}$  in the middle equation is enclosed in a red box and labeled "update" in red text. An upward arrow from the text "new" weight points to the  $w_1$  on the left side of the middle equation. Another upward arrow from the text "original" weight points to the  $w_1$  on the right side of the middle equation. A diagonal arrow points from the text "(negative) steepest direction of C" to the negative sign between the  $w_1$  terms in the middle equation.

$$w_0 \leftarrow w_0 - \alpha \frac{dC}{dw_0} \quad w_1 \leftarrow w_1 - \alpha \frac{dC}{dw_1} \quad w_2 \leftarrow w_2 - \alpha \frac{dC}{dw_2}$$

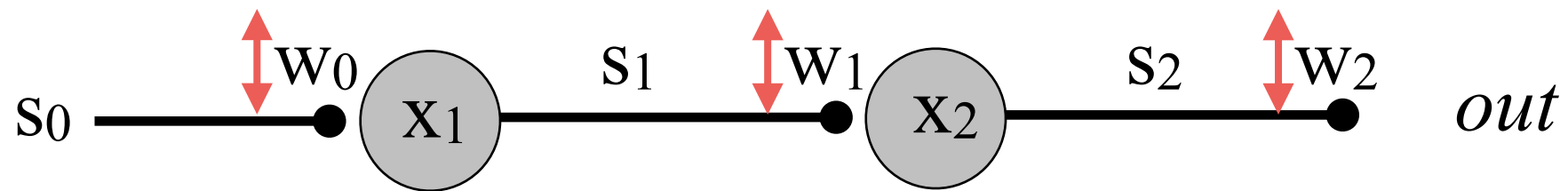
“new” weight      “original” weight      (negative) steepest direction of C

Procedure: gradient descent

$\alpha$  = learning rate

# Follow the cost function

**Q:** How does the cost function change due to changes in weights?



Consider:

$$\frac{dC}{dw_2} = ???$$

We know how  $C$  depends on  $out$ :  $C = \frac{1}{2} \left( \text{target} - \text{out} \right)^2$

And we know how  $out$  depends on  $w_2$ :  $out = s_2 w_2$

To compute the derivative, use the **chain rule** ...

# Follow the cost function

Our goal:

$$\frac{dC}{dw_2} = ???$$

We know  $C$  depends on  $out$ , and  $out$  depends on  $w_2$  ...

$$\frac{dC}{dw_2} = \frac{dC}{dout} \frac{dout}{dw_2}$$

We've already solved the first derivative!

$$\frac{dC}{dout} = out - target \quad \textit{Slide 35}$$

Let's compute the next derivative ...

## Follow the cost function

$$\frac{d \text{out}}{d w_2} = ???$$

We know:  $\text{out} = s_2 w_2$

$$\text{So, } \frac{d \text{out}}{d w_2} = \frac{d (s_2 w_2)}{d w_2} = s_2 \frac{d w_2}{d w_2} = \boxed{s_2}$$

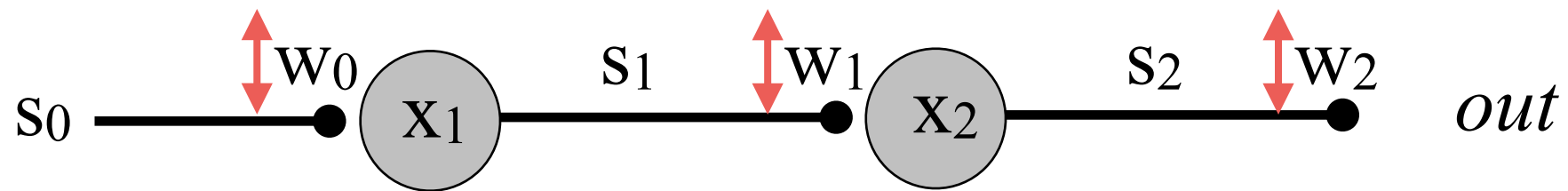
Then,

$$\frac{d C}{d w_2} = \frac{d C}{d \text{out}} \frac{d \text{out}}{d w_2}$$

$$\boxed{\frac{d C}{d w_2} = (\text{out} - \text{target}) s_2}$$

# Follow the cost function

**Q:** How does the cost function change due to changes in weights?



We found,  $\frac{dC}{dw_2} =$   $(out - target)$

How bad we're doing.

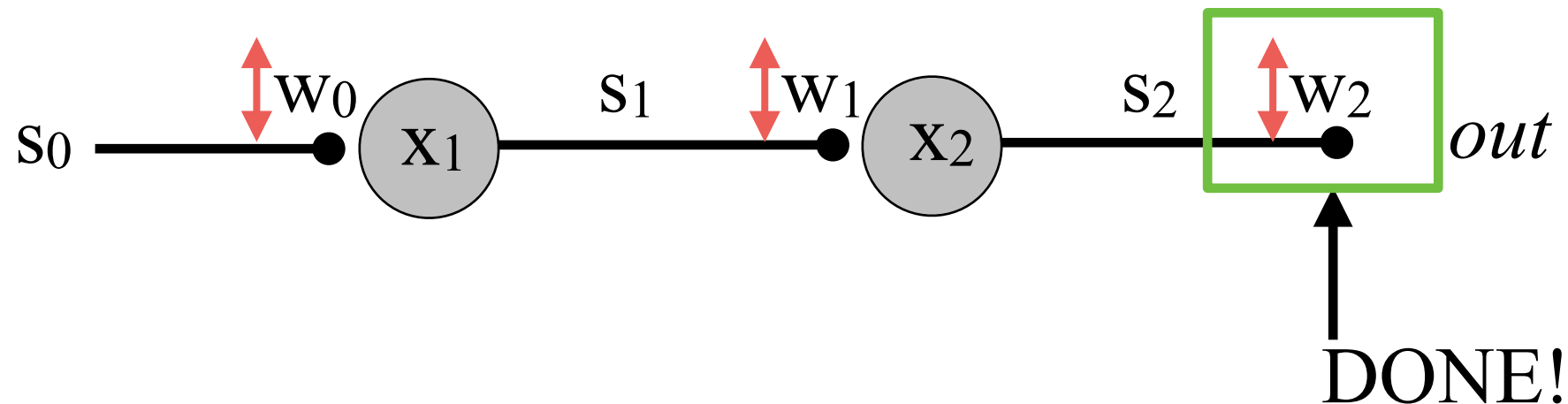
$s_2$   
↑  
output from  $x_2$

Update the weight  $w_2$ :

$$w_2 \leftarrow w_2 - \alpha \frac{dC}{dw_2} \quad \text{becomes} \quad \text{ $w_2 \leftarrow w_2 - \alpha(out - target)s_2$ }$$

## Follow the cost function

So, we've now found an equation to update one of the weights  $w_2$  that acts to minimize the cost function.



$$w_2 \text{ update: } w_2 \leftarrow w_2 - \alpha(out - \text{target})s_2$$

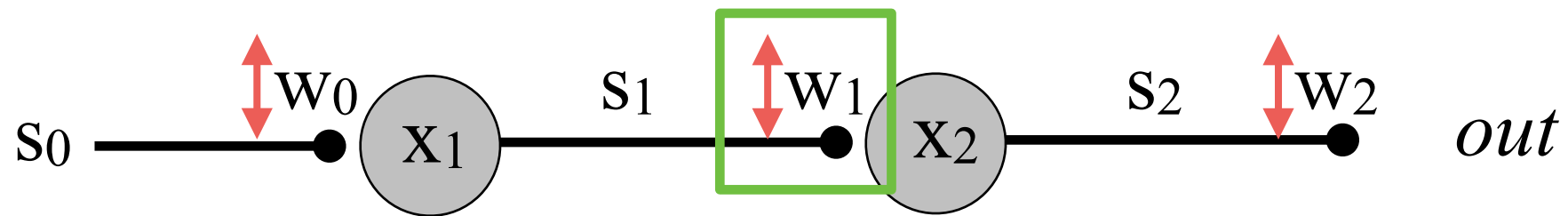
**Q:** Can we find equations to update  $w_1$  and  $w_0$  ?

**A:** Yes we can.

It'll seem difficult, but we've already done the hard work ...

## Follow the cost function

**Q:** How does the cost function change due to change in  $w_1$ ?



$$\frac{dC}{dw_1} = ???$$

We don't know this ...  
but can write it using  
things we do know.

We need the **chain rule**:

$$\frac{dC}{dw_1} = \frac{dC}{dout} \frac{dout}{dw_1}$$



## Follow the cost function

**Q:** How does the cost function change due to change in  $w_1$ ?

$$\frac{dC}{dw_1} = \frac{dC}{dout} \frac{dout}{dw_1}$$

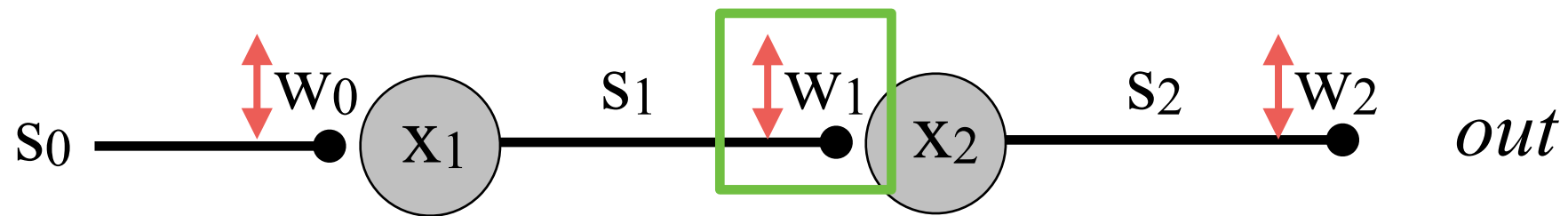
We already found:

$$\frac{dC}{dout} = out - target \quad (Slide\ 35)$$

$$\frac{dout}{dw_1} = w_2 s_2 (1 - s_2) s_1 \quad (Slide\ 23)$$

# Follow the cost function

**Q:** How does the cost function change due to change in  $w_1$ ?



We conclude:

$$\frac{dC}{dw_1} = \frac{dC}{dout} \frac{dout}{dw_1}$$

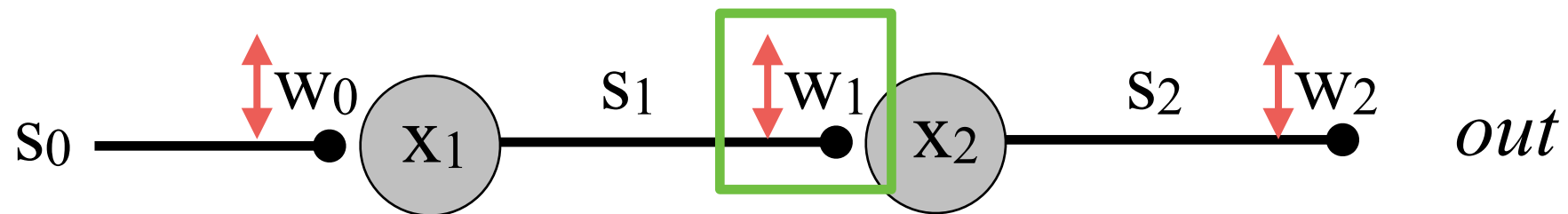
$$\frac{dC}{dw_1} = \boxed{(out - target)} w_2 s_2 (1 - s_2) s_1$$

How bad we're doing.

↑  
complicated expression  
of outputs and weight

## Follow the cost function

**Q:** How does the cost function change due to change in  $w_1$ ?



Update the weight  $w_1$ :

$$w_1 \leftarrow w_1 - \alpha \frac{dC}{dw_1} \quad \text{substitute in for this!}$$

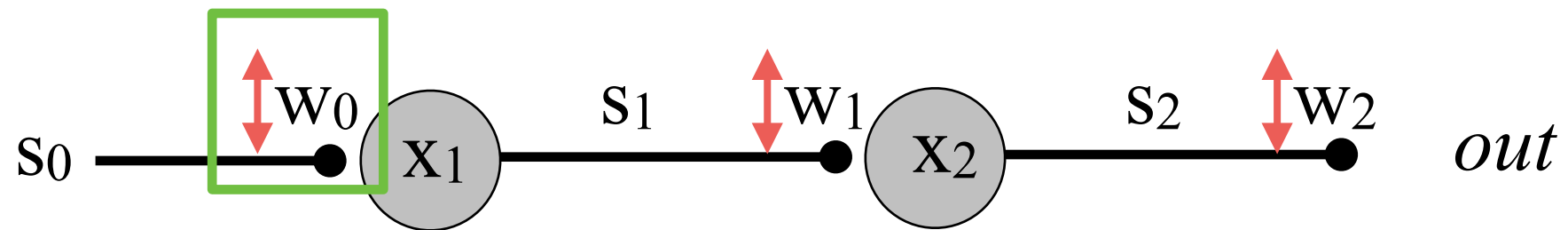
becomes

$$w_1 \leftarrow w_1 - \alpha (out - target) w_2 s_2 (1 - s_2) s_1$$

**Q:** What happens when  $out = target$ ?

# Follow the cost function

**Q:** How does the cost function change due to change in  $w_0$ ?

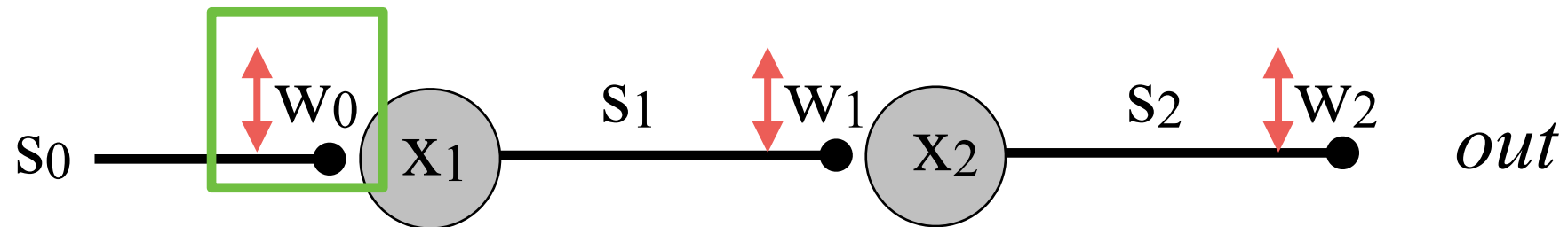


$$\frac{dC}{dw_0} = ???$$

Try it ...

## Follow the cost function

**Q:** How does the cost function change due to change in  $w_0$ ?



We conclude:

$$\frac{dC}{dw_0} = (out - target) w_2 s_2 (1 - s_2) w_1 s_1 (1 - s_1) s_0$$

and

$$w_0 \leftarrow w_0 - \alpha(out - target)w_2s_2(1 - s_2)w_1s_1(1 - s_1)s_0$$

Impressive expression!

# Put it all together

Prescription to find the weights that minimize cost function  
(so that *out* is near target).

1. Choose random initial weights.

$$w_0 = 2 \qquad w_1 = 1 \qquad w_2 = 0.5$$

2. Fix input at desire value, and calculate *out*.



# Put it all together

## Prescription (continued)

### 3. Update the weights

$$w_2 \leftarrow w_2 - \alpha(out - target)s_2$$

$$w_1 \leftarrow w_1 - \alpha(out - target)w_2s_2(1 - s_2)s_1$$

$$w_0 \leftarrow w_0 - \alpha(out - target)w_2s_2(1 - s_2)w_1s_1(1 - s_1)s_0$$

Note: We know all of the values required

$\alpha$  = learning rate, we choose this.

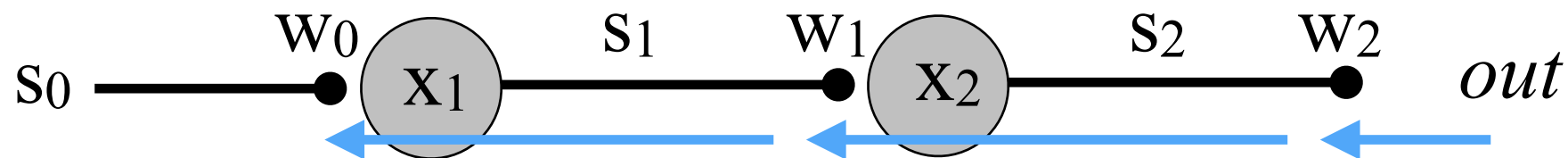
$s_0, s_1, s_2$  = calculated during forward propagation

# Put it all together

## Prescription (continued)

4. Repeat Steps 2 & 3 until error is small enough.  
or *out* is close enough to target.

This procedure is called **backpropagation**



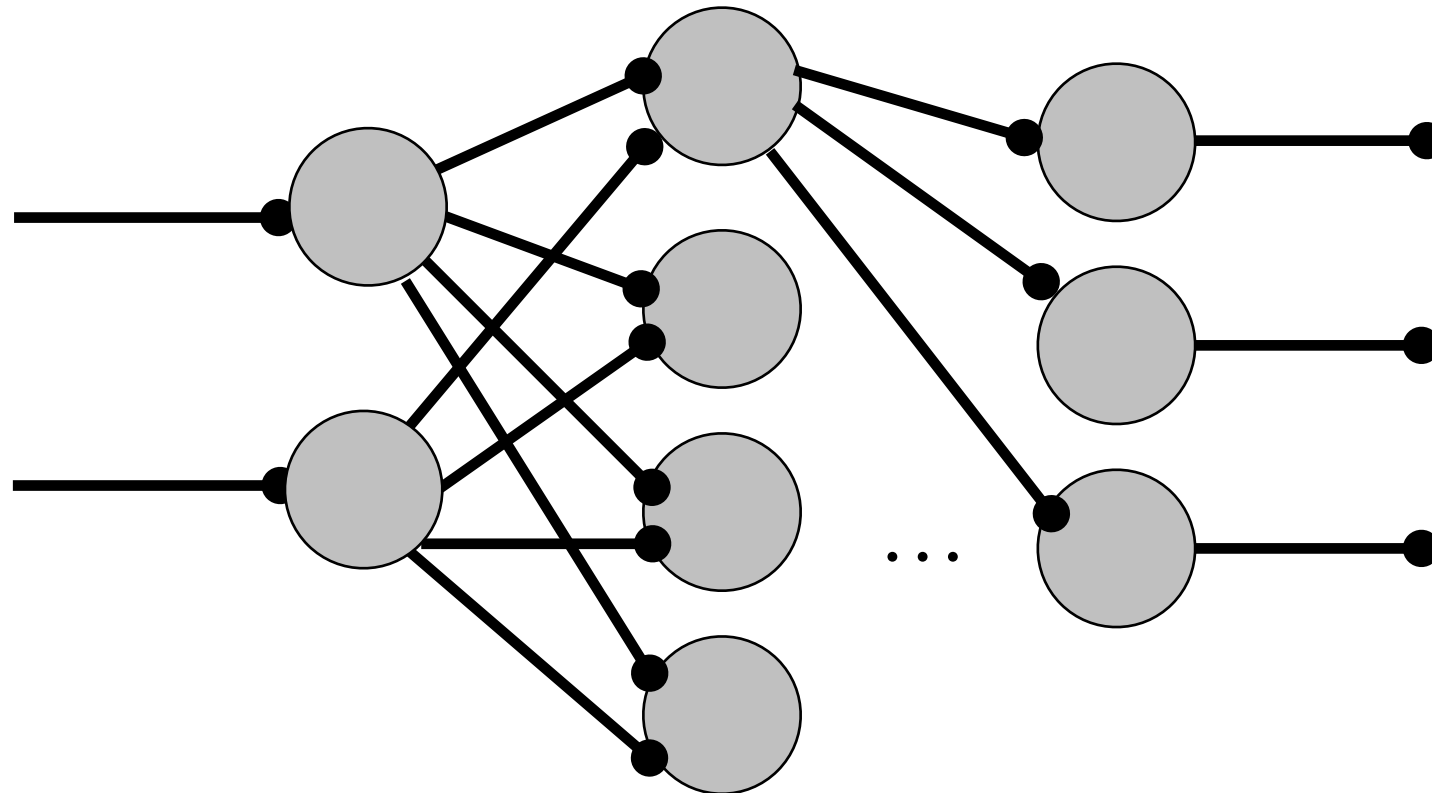
we work “backwards” through our neural network

from *out* to changes in  $w_2$  to changes in  $w_1$  to changes in  $w_0$



# Backpropagation

Can evaluate more complicated neural networks



Same ideas apply, but algebra is intense.

Cool example: [playground.tensorflow.org](https://playground.tensorflow.org)

## Next ...

Implement backpropagation in Python

$S_0$	$out$
[ [ 0.32625098	0.71507894]
[-0.65595776	1.0065779 ]
[-0.9664146	1.09676408]
...	
[ 1.53148642	0.48558244]
[-1.07312405	1.12584863]
[ 1.32680081	0.51212226]]

**Q:** What weights ( $w_0$   $w_1$   $w_2$ ) produced these data?

