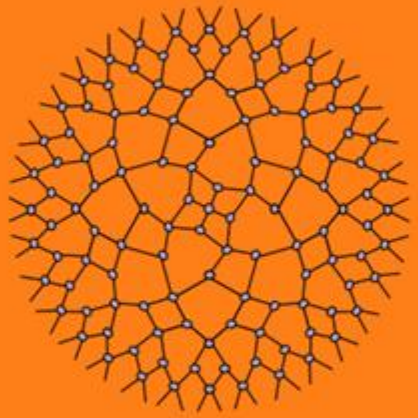
ML Algorithms

NEURAL NETWORKS



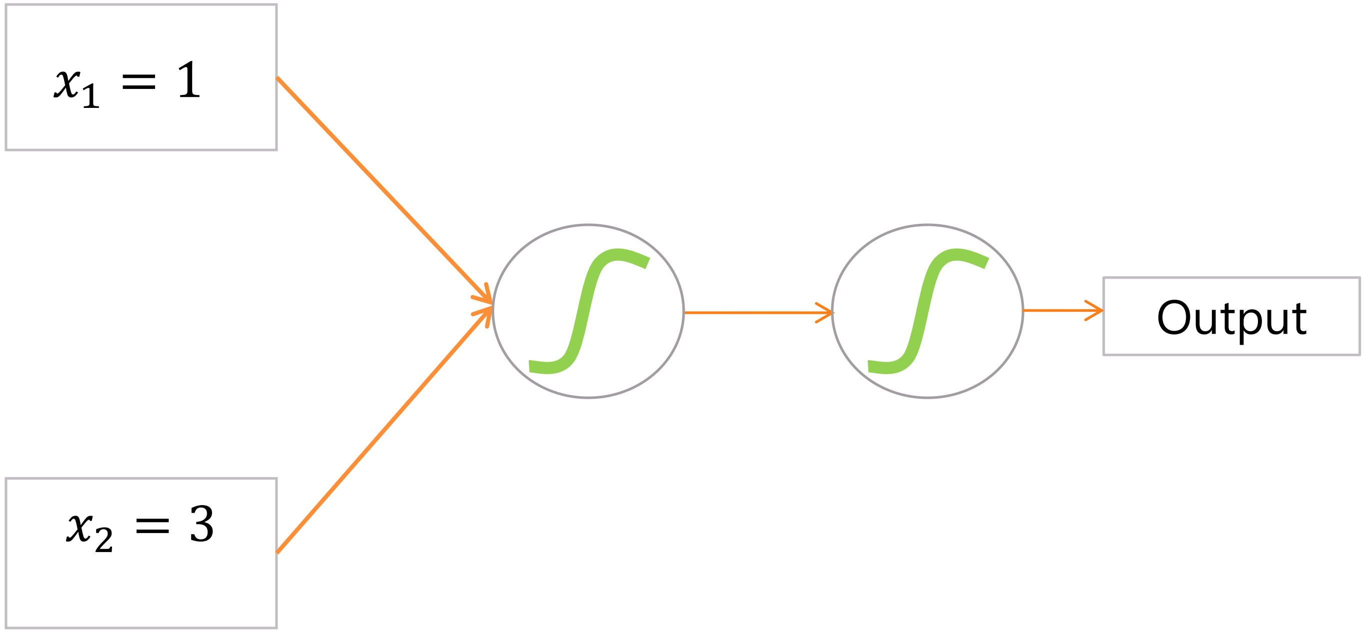
Class

A Detailed Look At Neural Networks

Topic

The Backpropagation Algorithm;
Deltas For The Output Layer;
The General Case;
Evaluating The Performance Of A Binary Classifier

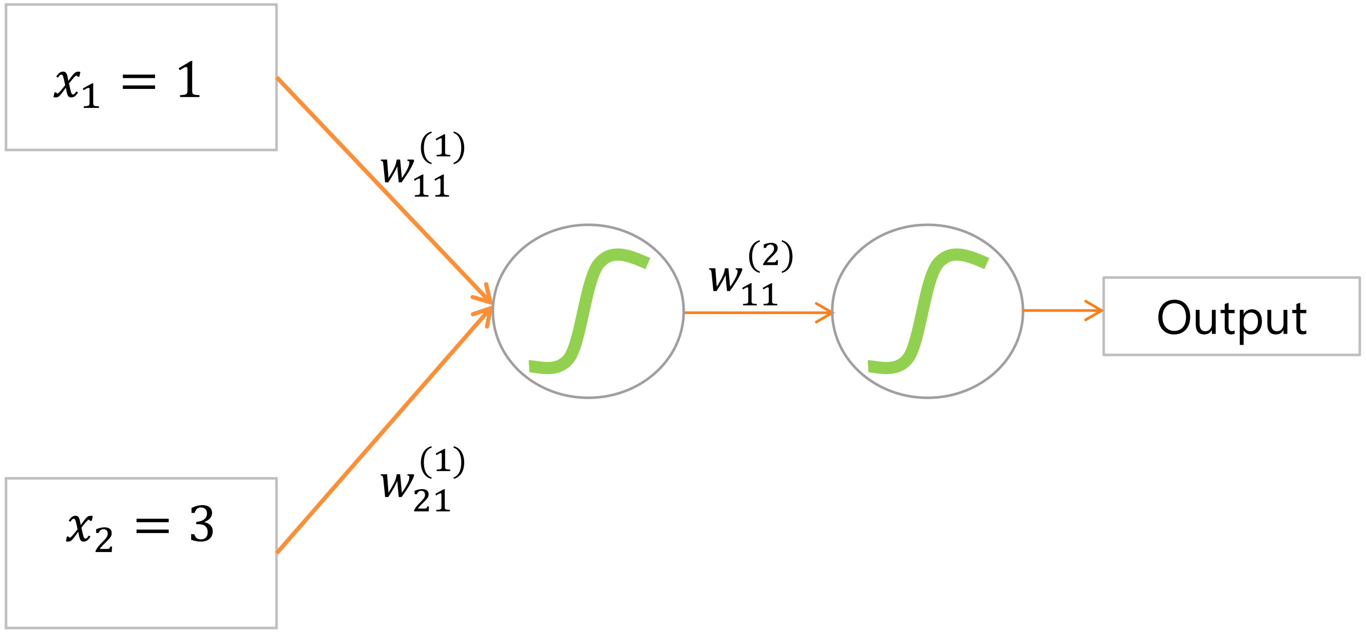
Backpropagation



Backpropagation

- A new, recursive approach, to computing cost function gradient
- It's discovery has reduced computation time

The Backpropagation Algorithm



Observed Response $\boldsymbol{y} = \mathbf{1}$

The Backpropagation Algorithm

Step 1: Compute the signal going into the node in the hidden layer:

- $s_1^{(1)} = w_{11}^{(1)} + 3w_{21}^{(1)}$

Step 2: Compute the output coming out of the node in the hidden layer:

- $x_1^{(1)} = h\left(s_1^{(1)}\right) = h\left(w_{11}^{(1)} + 3w_{21}^{(1)}\right)$

Step 3: Compute the signal going into the node in the output layer:

- $s_1^{(2)} = w_{11}^{(2)} x_1^{(1)} = w_{11}^{(2)} h(w_{11}^{(1)} + 3w_{21}^{(1)})$

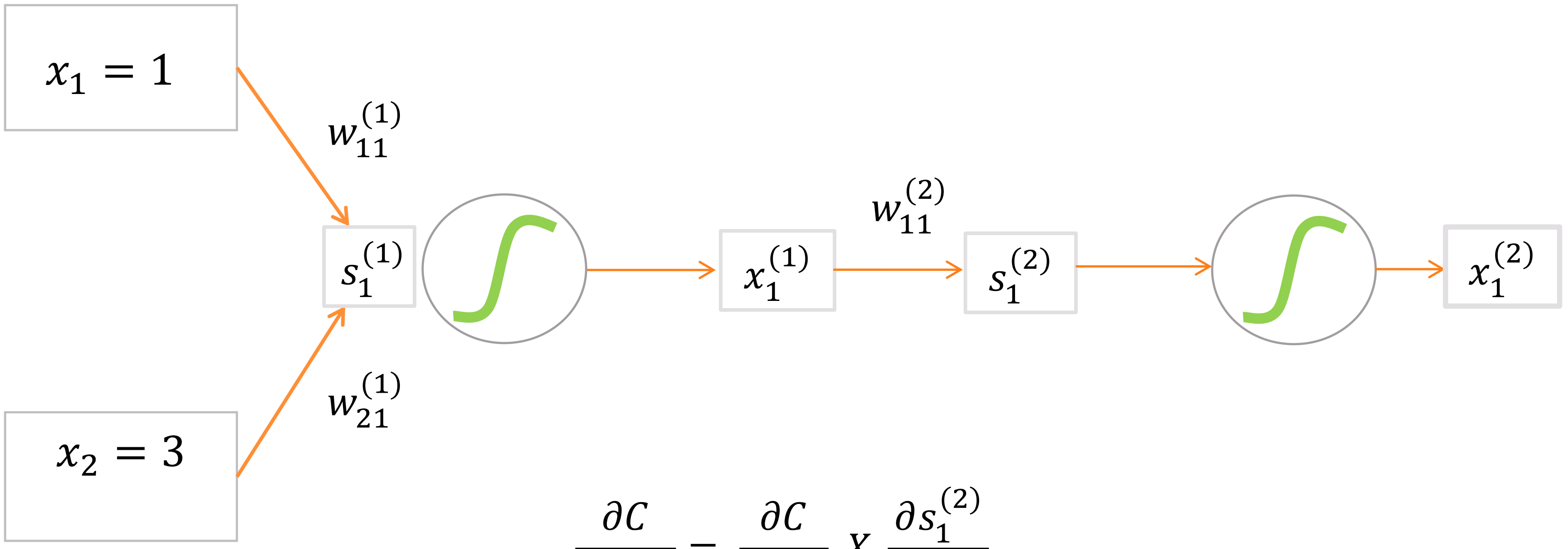
Step 4: Compute the output coming out from the node in the output layer :

- $x_1^{(2)} = h\left(s_1^{(2)}\right) = h\left(w_{11}^{(2)}\, h(s_1^{(1)})\right) = h\left(w_{11}^{(2)} h(w_{11}^{(1)} + 3w_{21}^{(1)})\right)$

$$\boldsymbol{x_1^{(2)} = h(ah(bh\,(.)))}$$

The Backpropagation Algorithm

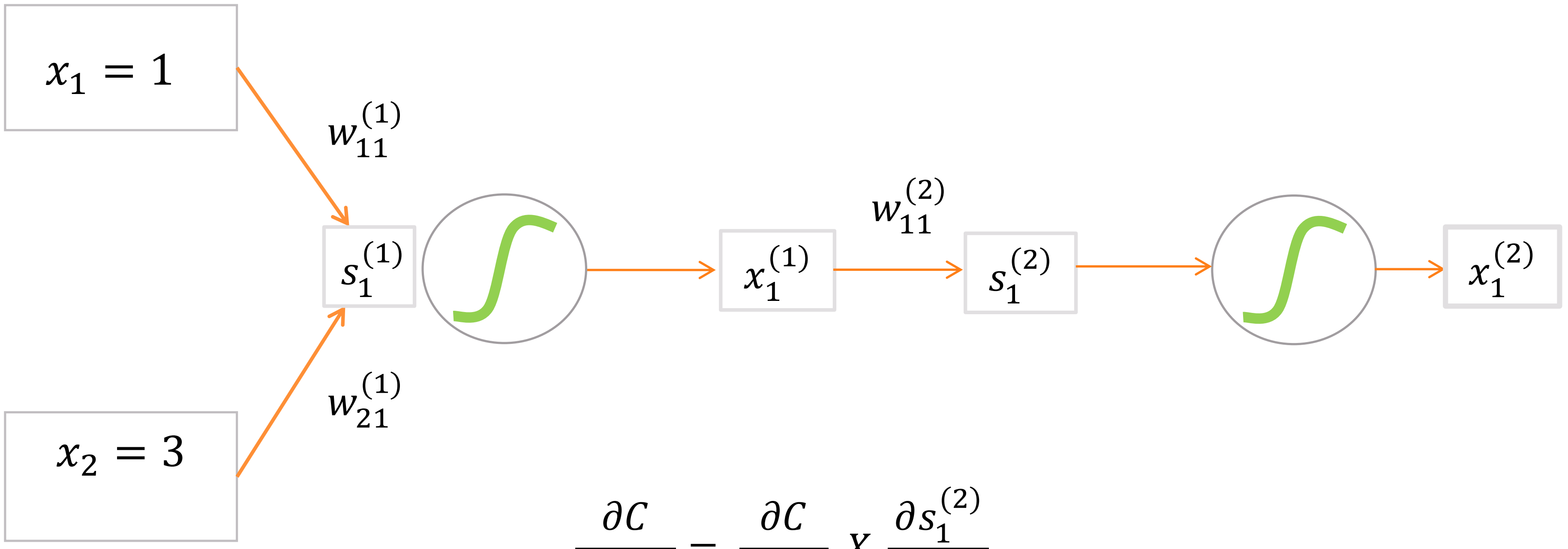
Three parameters require 3 partial derivatives to calculate the gradient



$$\frac{\partial C}{\partial w_{11}^{(2)}} = \frac{\partial C}{\partial s_1^{(2)}} X \frac{\partial s_1^{(2)}}{\partial w_{11}^{(2)}}$$

The Backpropagation Algorithm

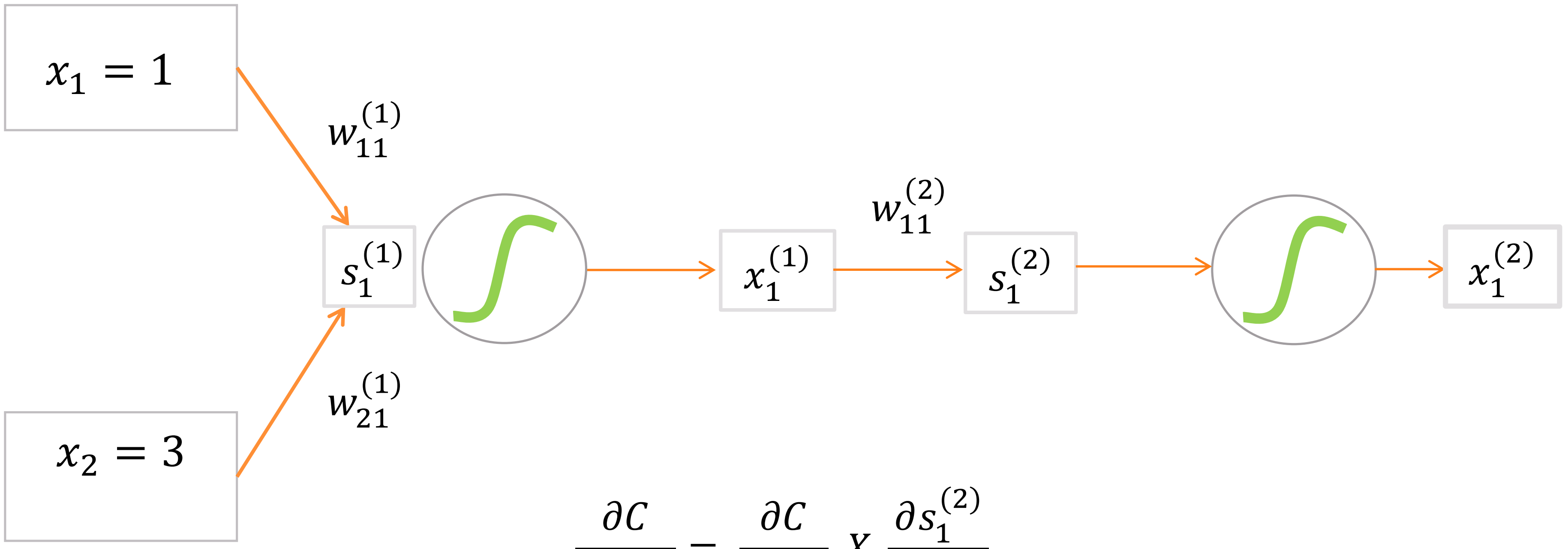
The cost function does not directly depend upon the weight



$$\frac{\partial C}{\partial w_{11}^{(2)}} = \frac{\partial C}{\partial s_1^{(2)}} X \frac{\partial s_1^{(2)}}{\partial w_{11}^{(2)}}$$

The Backpropagation Algorithm

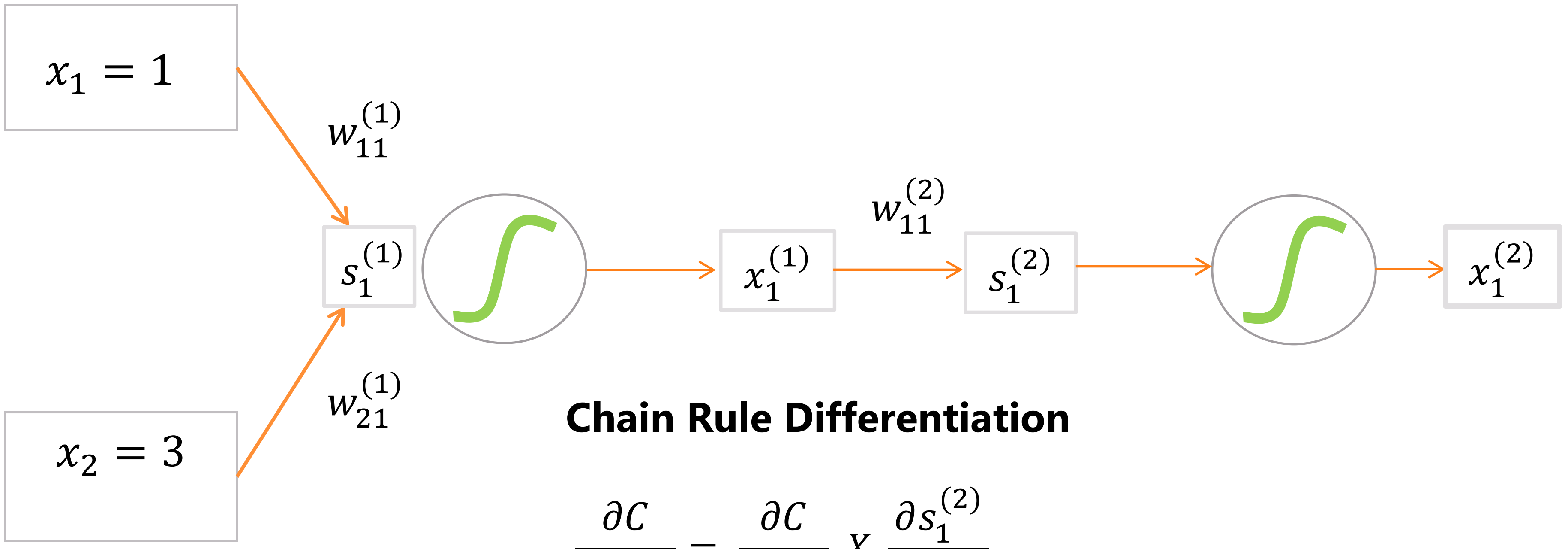
C depends on the signal, which depends upon the weights



$$\frac{\partial C}{\partial w_{11}^{(2)}} = \frac{\partial C}{\partial s_1^{(2)}} X \frac{\partial s_1^{(2)}}{\partial w_{11}^{(2)}}$$

The Backpropagation Algorithm

The chain rule of differentiation decomposes the original derivative into 2 parts

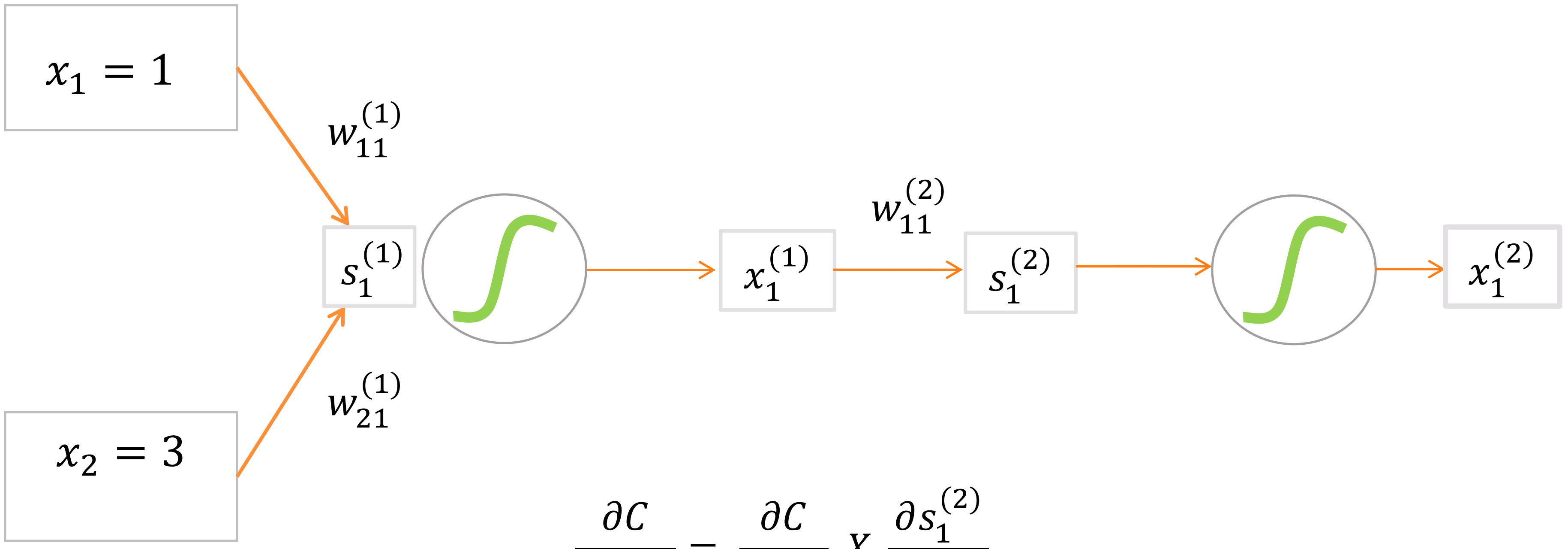


Chain Rule Differentiation

$$\frac{\partial C}{\partial w_{11}^{(2)}} = \frac{\partial C}{\partial s_1^{(2)}} X \frac{\partial s_1^{(2)}}{\partial w_{11}^{(2)}}$$

The Backpropagation Algorithm

1st term: The partial derivative of C with respect to the signal from the hidden neuron to the output neuron

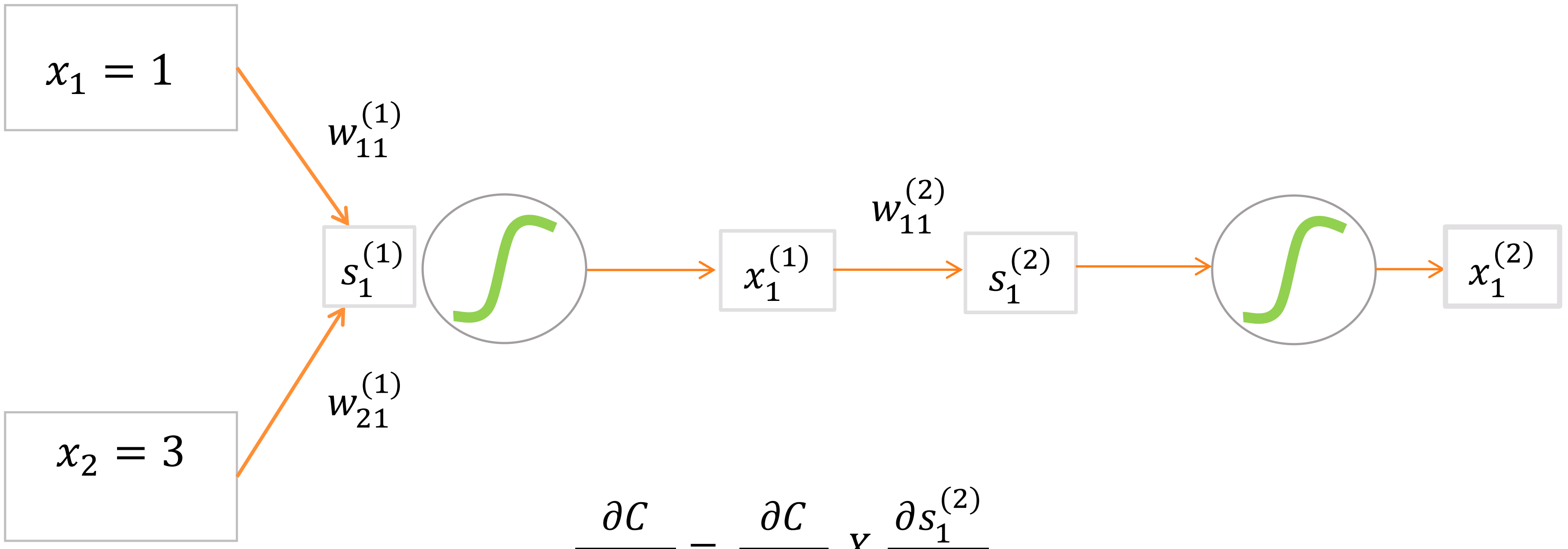


$$\frac{\partial C}{\partial w_{11}^{(2)}} = \frac{\partial C}{\partial s_1^{(2)}} \; X \; \frac{\partial s_1^{(2)}}{\partial w_{11}^{(2)}}$$

Partial derivative of C $\partial_1^{(2)}$

The Backpropagation Algorithm

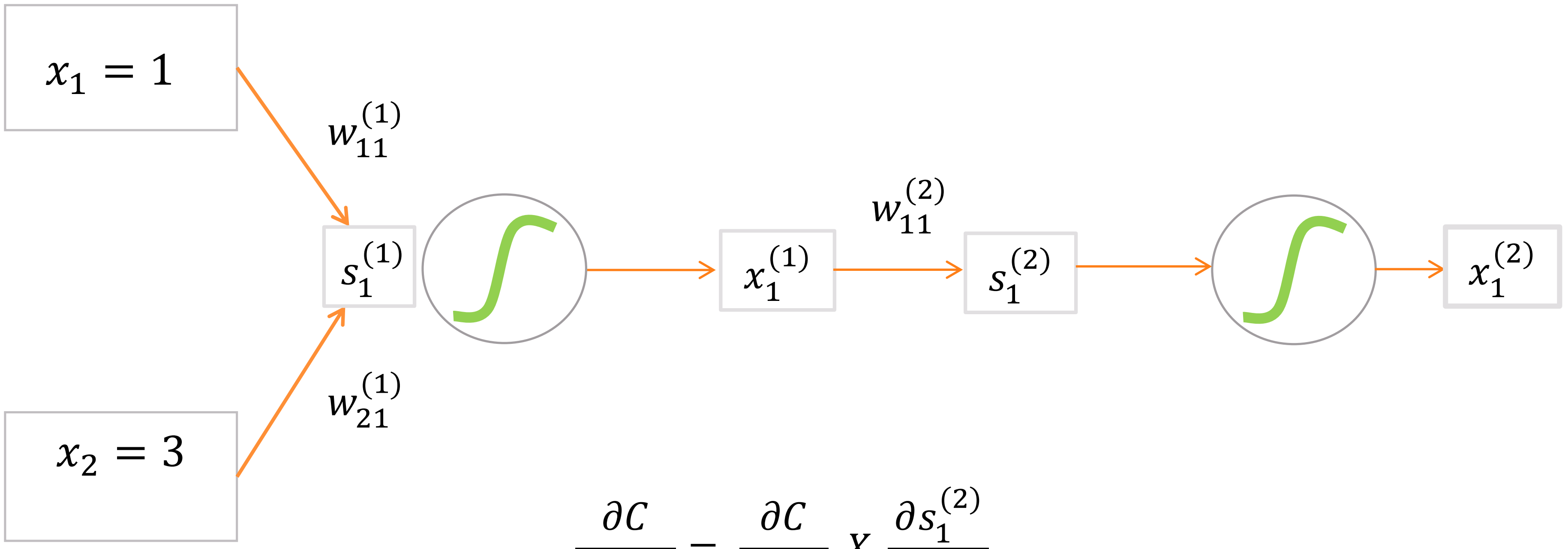
This tells us how much the cost would vary if the signal were to vary a little bit



$$\frac{\partial C}{\partial w_{11}^{(2)}} = \underbrace{\frac{\partial C}{\partial s_1^{(2)}}}_{\boxed{\partial_1^{(2)}}} X \frac{\partial s_1^{(2)}}{\partial w_{11}^{(2)}}$$

The Backpropagation Algorithm

2nd term: The partial derivative of the signal from the neuron in the hidden layer to the neuron in the output layer

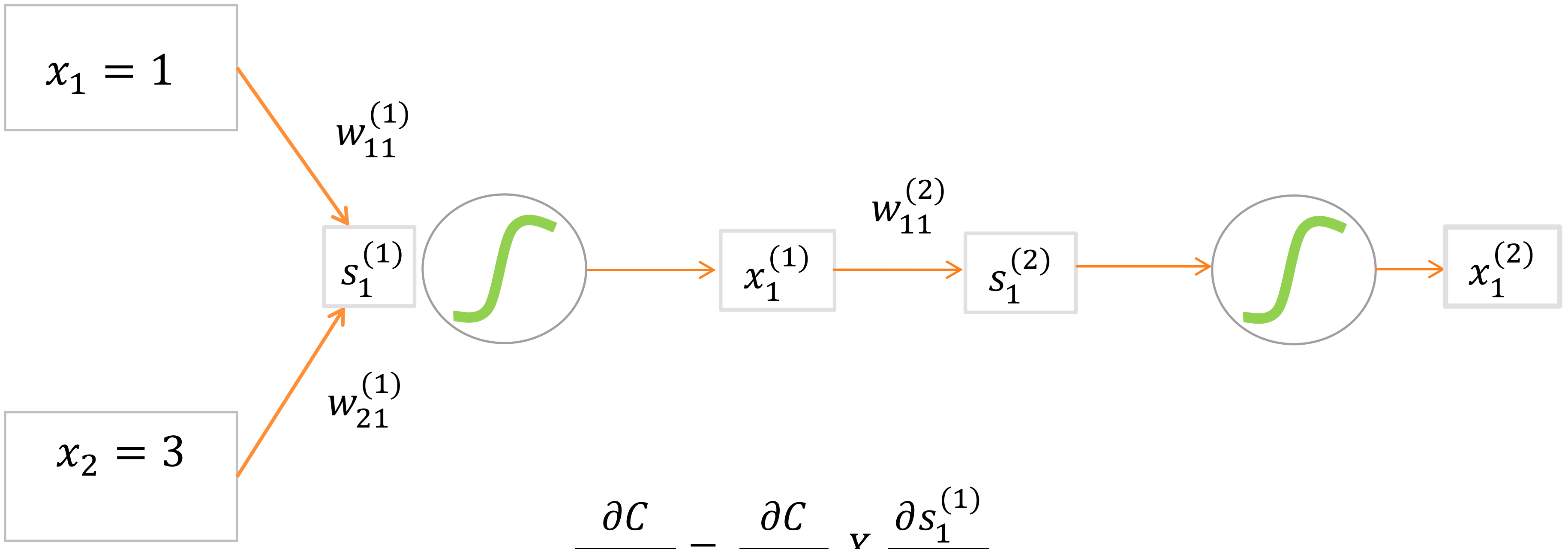


$$\frac{\partial C}{\partial w_{11}^{(2)}} = \frac{\partial C}{\partial s_1^{(2)}} \times \frac{\partial s_1^{(2)}}{\partial w_{11}^{(2)}}$$

$$\frac{\partial s_1^{(2)}}{\partial w_{11}^{(2)}} \rightarrow x_1^{(1)}$$

Partial derivative of neuronal signal

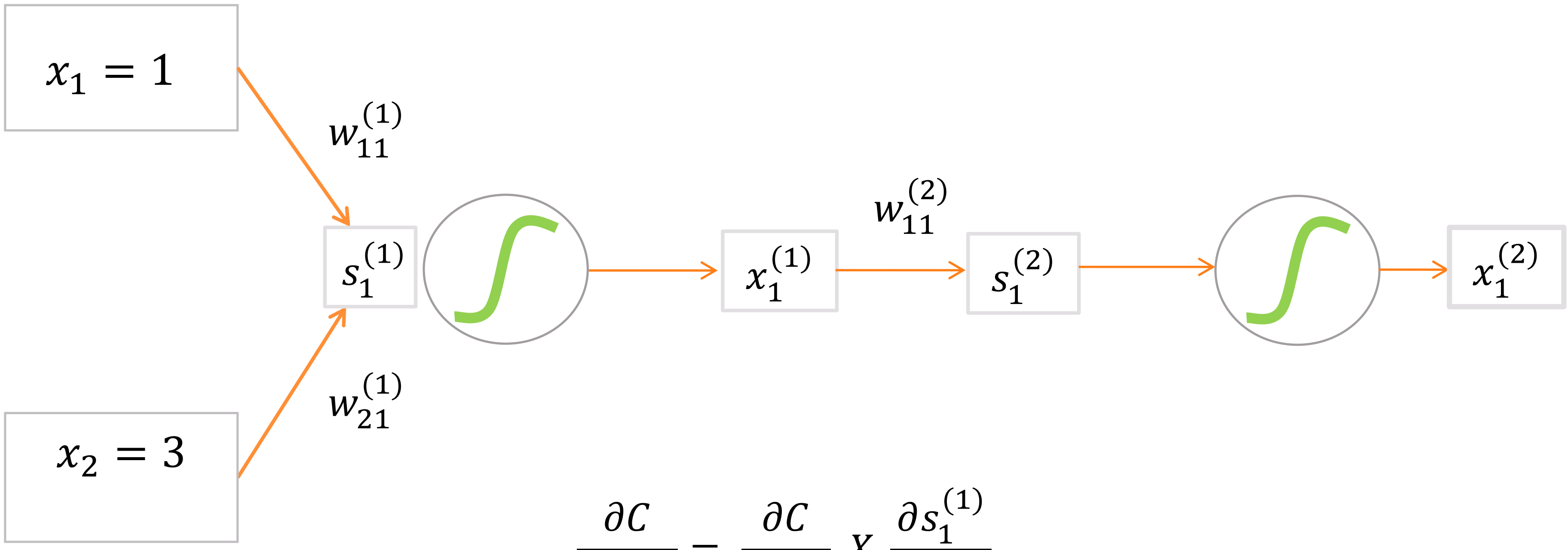
The Backpropagation Algorithm



$$\frac{\partial C}{\partial w_{11}^{(1)}} = \frac{\partial C}{\partial s_1^{(1)}} \; X \; \underbrace{\frac{\partial s_1^{(1)}}{\partial w_{11}^{(1)}}}_{1}$$

$$\boldsymbol{\nabla C} = \left[\frac{\boldsymbol{\partial C}}{\boldsymbol{\partial w_{ij}^{(l)}}}\right]$$

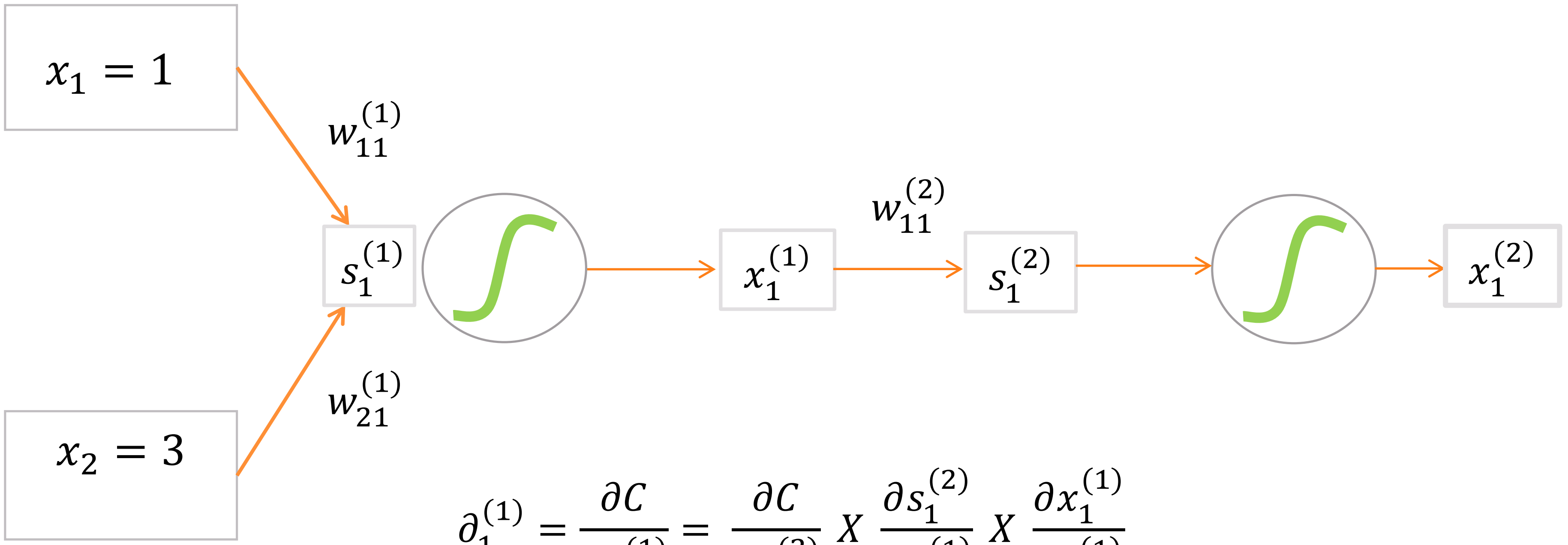
The Backpropagation Algorithm



$$\frac{\partial C}{\partial w_{21}^{(1)}} = \frac{\partial C}{\partial s_1^{(1)}} \; X \; \frac{\partial s_1^{(1)}}{\partial w_{21}^{(1)}}$$

$$\partial_1^{(1)} \qquad\qquad 3$$

The Backpropagation Algorithm

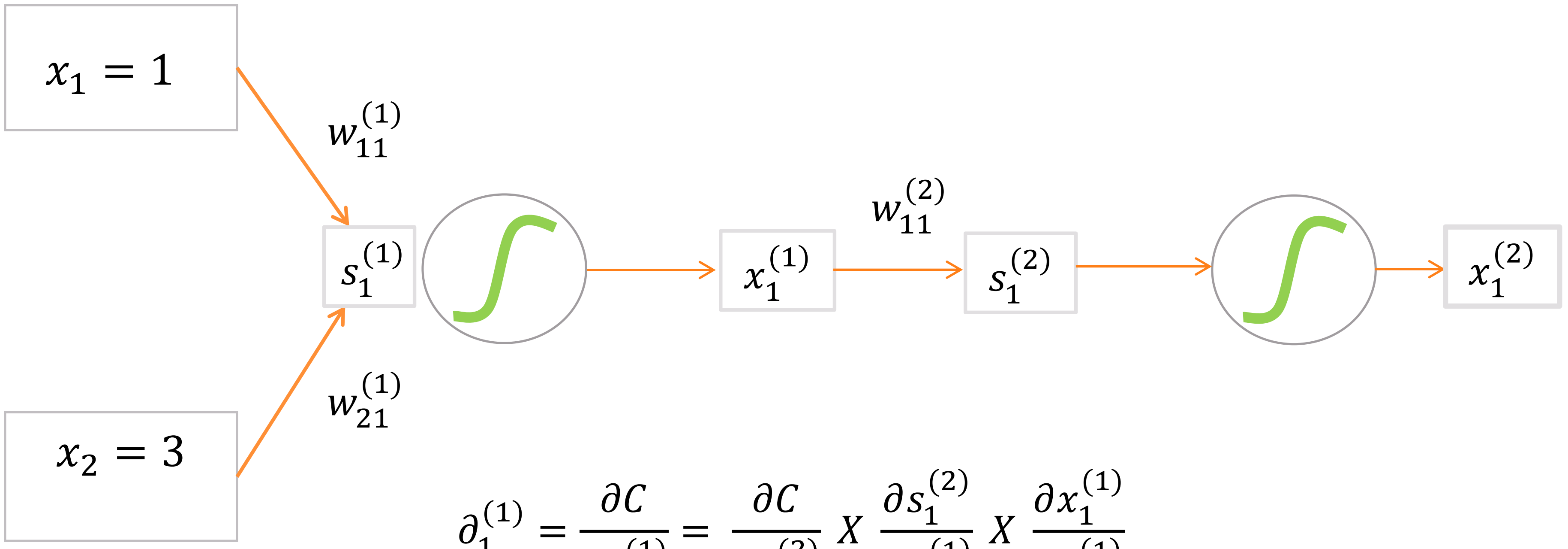


$$\partial_1^{(1)} = \frac{\partial C}{\partial s_1^{(1)}} = \frac{\partial C}{\partial s_1^{(2)}} \; X \; \frac{\partial s_1^{(2)}}{\partial x_1^{(1)}} \; X \; \frac{\partial x_1^{(1)}}{\partial s_1^{(1)}}$$

- Perturbing the signal follows a path before it affects cost function
- If you change the signal, output from hidden neuron changes a little bit
- In turn affects signal that goes into output neuron
- Affects the cost function

The Backpropagation Algorithm

Delta in the hidden layer is determined by delta in output layer



$$\partial_1^{(1)} = \frac{\partial C}{\partial s_1^{(1)}} = \underbrace{\frac{\partial C}{\partial s_1^{(2)}}}_{\boxed{\partial_1^{(2)}}} \, X \, \underbrace{\frac{\partial s_1^{(2)}}{\partial x_1^{(1)}}}_{\boxed{w_{11}^{(2)}}} \, X \, \underbrace{\frac{\partial x_1^{(1)}}{\partial s_1^{(1)}}}_{\boxed{h'(s_1^{(1)})}}$$

The Backpropagation Algorithm: Summary

In general neural networks

- The deltas in each layer are determined by the deltas in previous layers
- Backpropagation can trace previous deltas all the way back to the first hidden layer

Deltas for the Output Layer

Determining the delta in the final layer

- $\partial_1^{(2)} = \frac{\partial C}{\partial s_1^{(2)}}$
- $C = y \log\left(h\left(s_1^{(2)}\right)\right) + (1-y)\log\left(1 - h\left(s_1^{(2)}\right)\right) = \log\left(h\left(s_1^{(2)}\right)\right)$

Output from neural network

y=1

$$\frac{\partial C}{\partial s_1^{(2)}} = \frac{\partial \log h\,(s_1^{(2)})}{\partial s_1^{(1)}} = \frac{1}{h(s_1^{(2)})} h'(s_1^{(2)})$$

The General Case

- Compute delta in the output layer: $\partial_1^{(L)}$
- Propagate them backwards starting from the output layer: $\partial_i^{(l-1)} = \sum_{j=1}^{d^{(l)}} \partial_j^{(l)} \; X \; \frac{\partial s_j^{(l)}}{\partial x_i^{(l-1)}} \; X \; \frac{\partial x_i^{(l)}}{\partial s_i^{(l-1)}}$
- Apply gradient descent algorithm to compute optimal parameter values of neural network

Evaluating the Performance of a Binary Classifier

Classification algorithms and machine learning

Gauge performance once a classifier has been trained

Performance determines the quality of the classifier

A classifier that performs badly, must be discarded

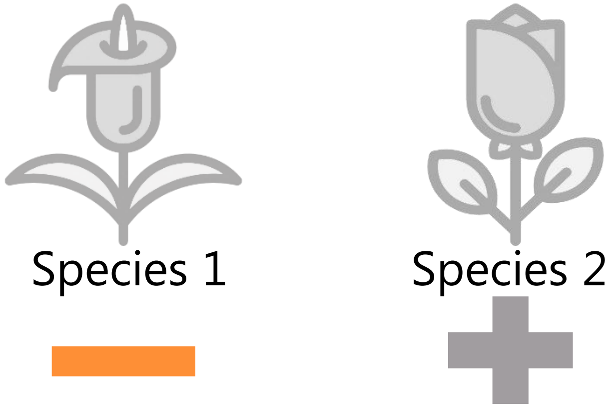
If it performs reasonably well, it can be retained as a final candidate

Evaluating the Performance of a Binary Classifier

Classification algorithms and machine learning

Positive & Negative Classes in binary Classification: E.g. cancer detection

Positive/negative classifications aren't always applicable



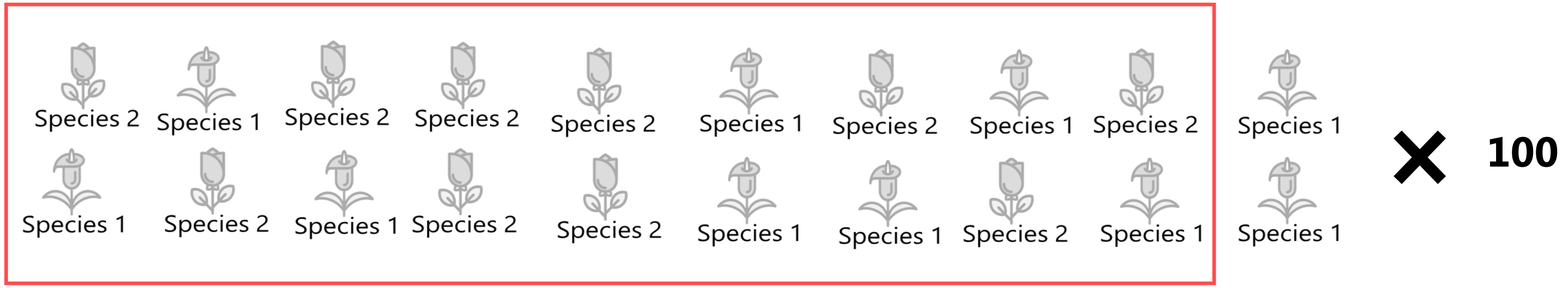
Evaluating the Performance of a Binary Classifier

Classification algorithms and machine learning

Positive & Negative Classes in binary Classification: E.g. cancer detection

Positive/negative classifications aren't always applicable

Accuracy: The first measure of a binary classifier's performance is accuracy, i.e., the number of observations a classifier has predicted correctly



Accuracy is 80%

Evaluating the Performance of a Binary Classifier

Classification algorithms and machine learning

Positive & Negative Classes in binary Classification: E.g. cancer detection

Positive/negative classifications aren't always applicable

Accuracy: The first measure of a binary classifier's performance is accuracy, i.e., the number of observations a classifier has predicted correctly

Sensitivity: Measures actual Positives identified as Positives, e.g., the % of cancer cases that were correctly identified in being cancerous

Specificity: Measures actual Negatives identified as Negatives, e.g., the % healthy cases that were correctly identified as being healthy

Recap

- Backpropagation
- The backpropagation algorithm
- The backpropagation algorithm: Summary
- Deltas for the output layer
- The general case
- Evaluating the performance of a binary classifier

JIGSAW ACADEMY

THE ONLINE SCHOOL OF ANALYTICS