Neural Networks

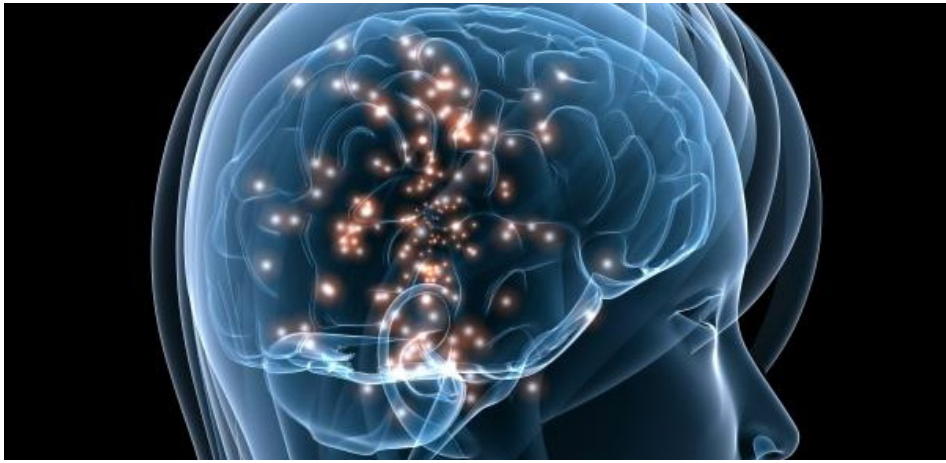
Instructor: Gary Overett

Biological Inspiration

- “Hey, brains are smart! Let’s copy what they do!”
- “But we don’t really know what they do!”
- “Mmm, well it looks a bit like … mumble, mumble, mumble … let’s try that.”
- “Whooho it works! … sort of … sometimes really well … sometimes not so much”

Biological Neural Nets

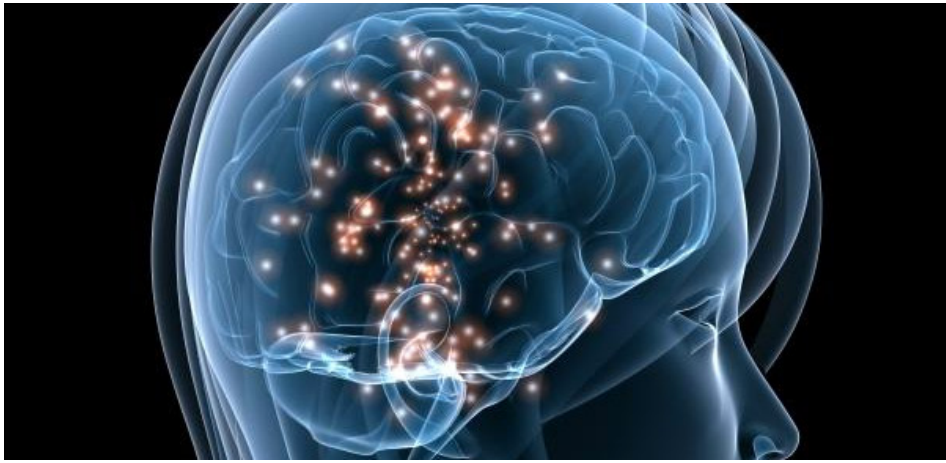
- Highly connected (10^4 connections)
- Neurons fire in response to input firings from connected neurons
- Certain specific structures to learn from e.g. visual cortex
- Somehow learn



Biological (Human) Neural Nets

- ~10^{10} neurons
- 0.001s switching time
- ~10^{4} connections per neuron
- Scene recognition in about 0.1 seconds. i.e. 100 inference* steps!
- Is this enough time?
- Massively parallel!

*may include fairly complex intra neuron inference



Biological Inspiration

Albatross

- Wings
- Body
- Tail

Airbus A380

- Wings
- Body
- Tail

Biological Inspiration

Albatross

- Kidney
- Liver
- Feathers
- Flaps wings

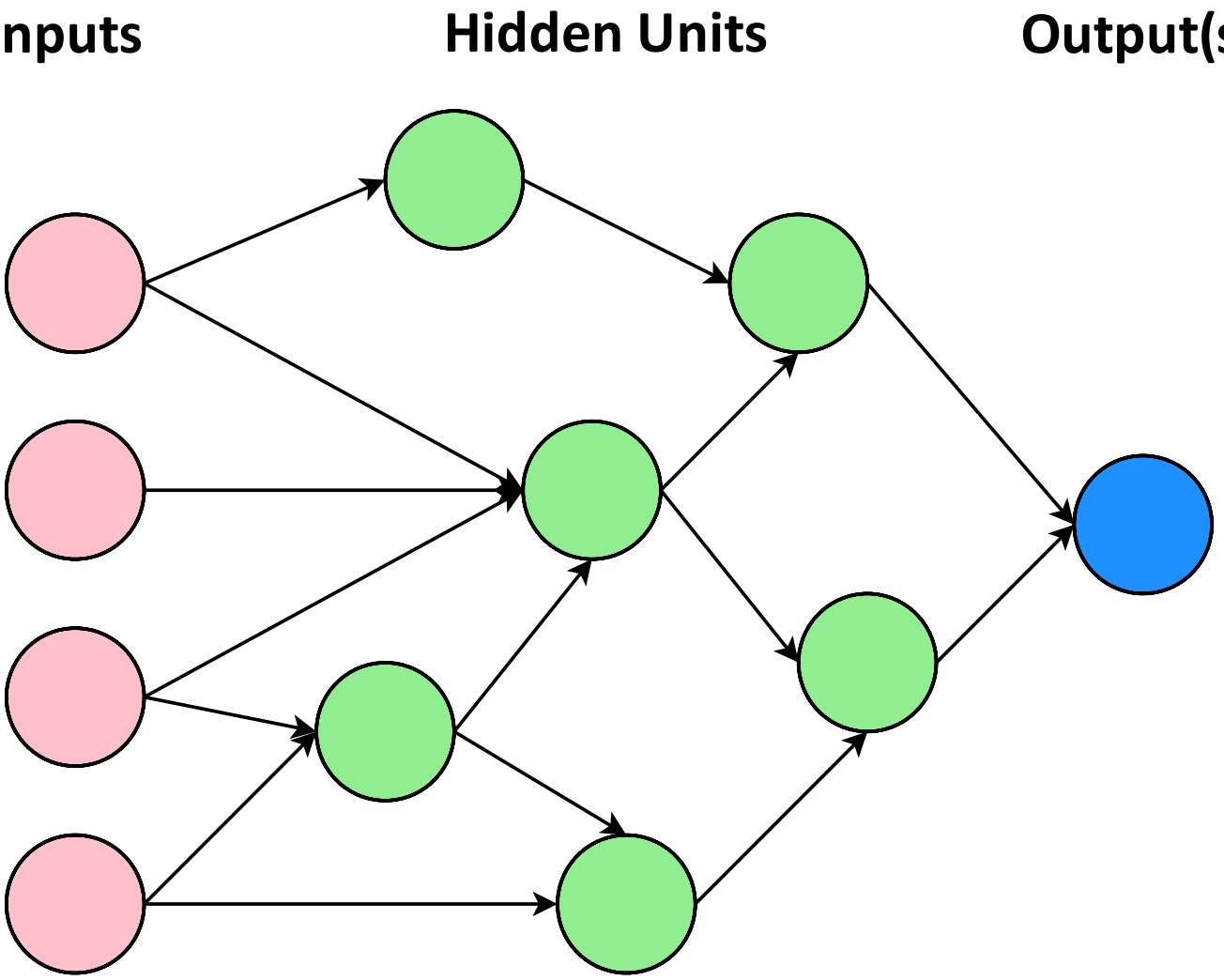
Airbus A380

- Rolls-Royce Trent Engine
- Jet Fuel
- Hydraulics
- Has flaps on wings!

Biological Inspiration of ANN's

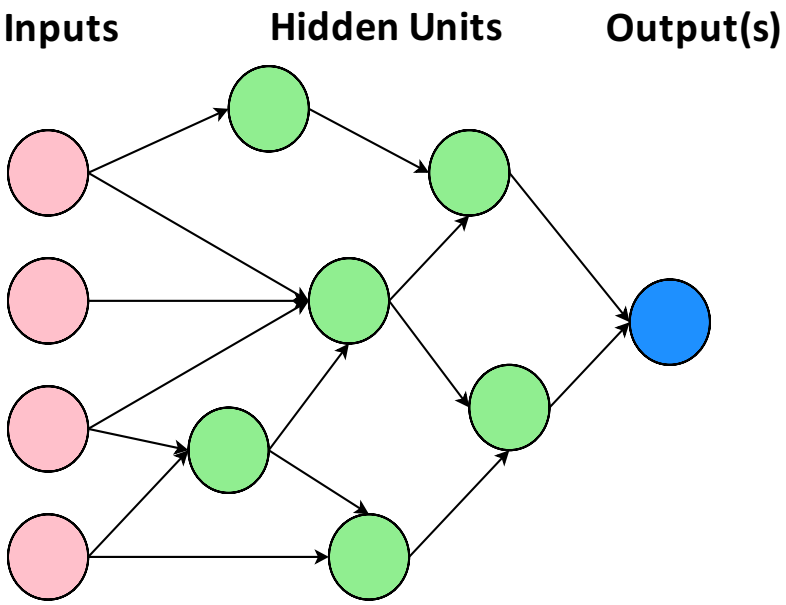
- Many parts of the Artificial Neural Network models were invented prior to the use of the biological analogy (perceptron's, logistic regression, …).
- Much of the standard methods used in ANN's is NOT actually as biologically plausible as first thought.
- But they do work very well (state-of-the-art) in many problems!
- As neuroscience develops we may yet get a significant biologically inspired "win"…

Neural Network Structure

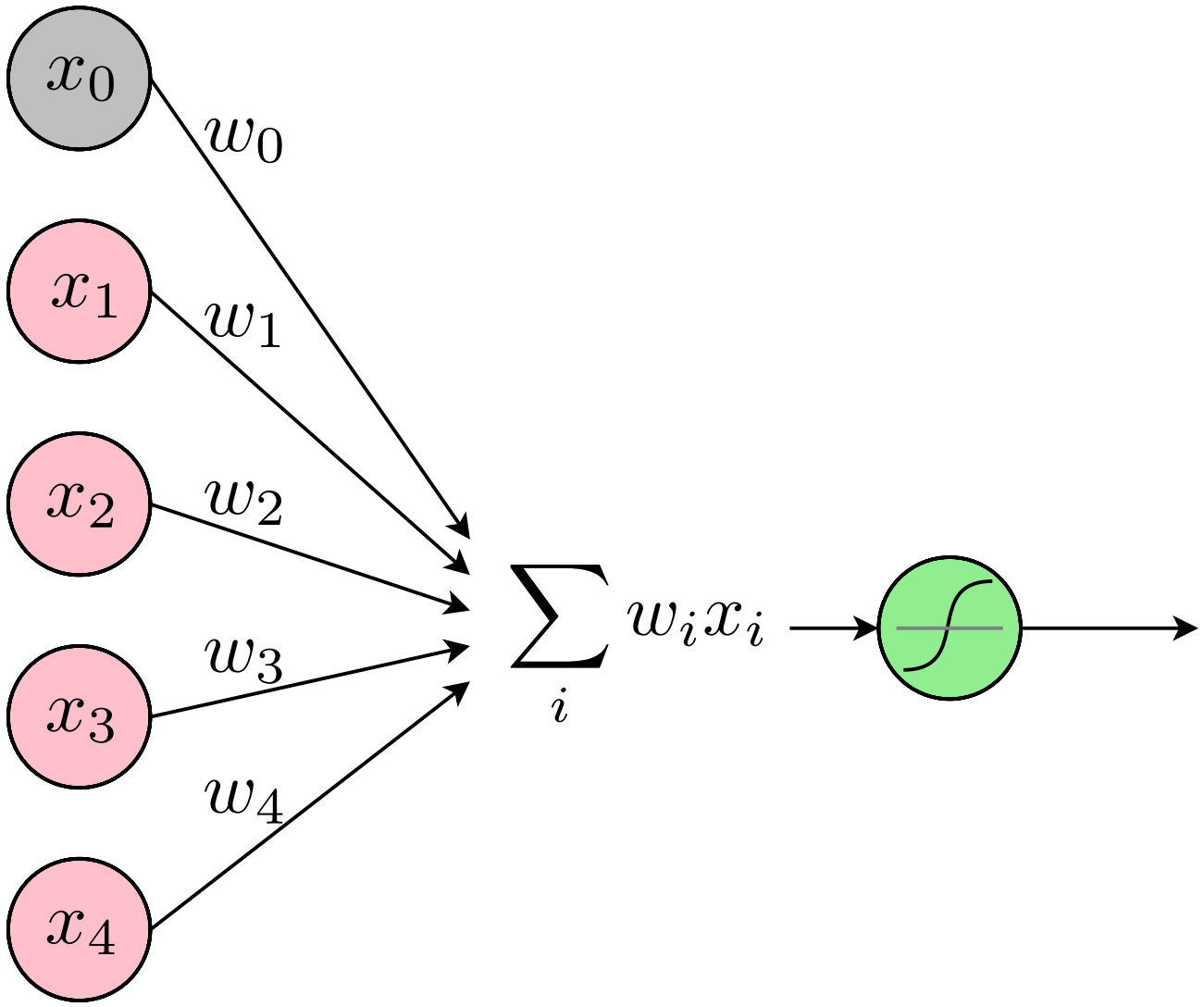


Neural Network Questions

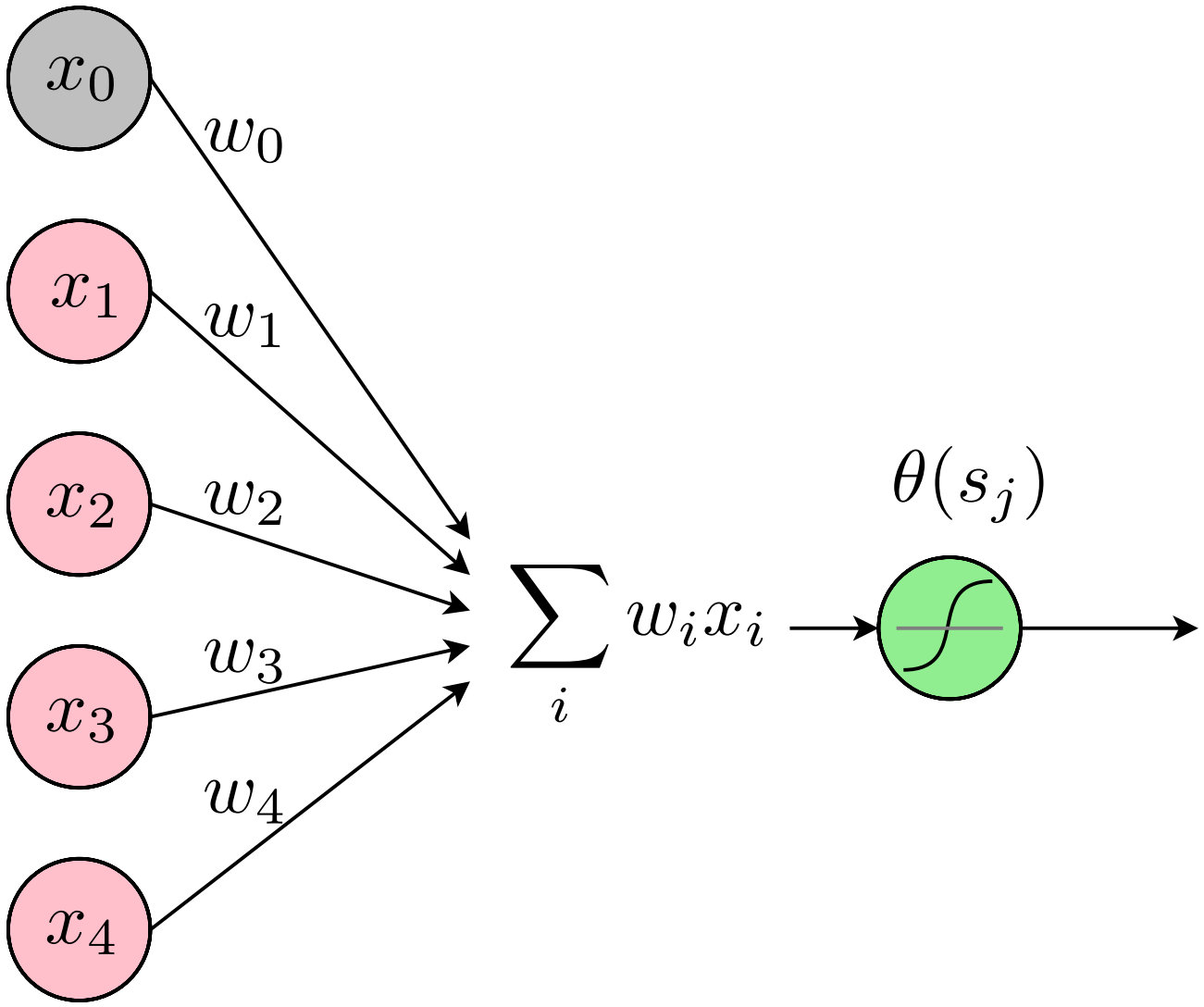
- What structure to use?
- What non-linearity's to adopt?
- Size/Depth of the network
- HOW TO DETERMINE (LEARN) THE WEIGHTS???
- How to use and arrange data?



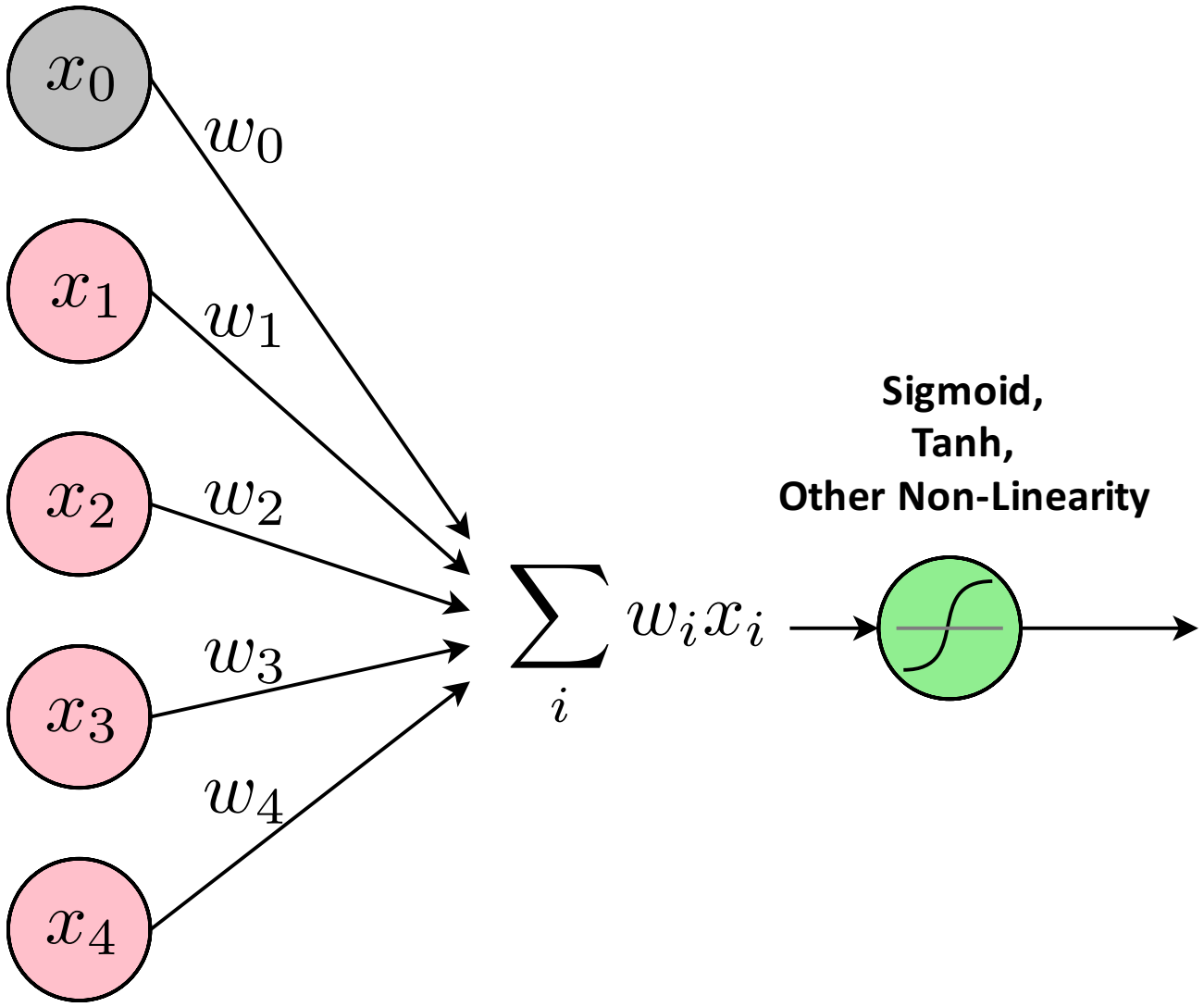
A Neuron



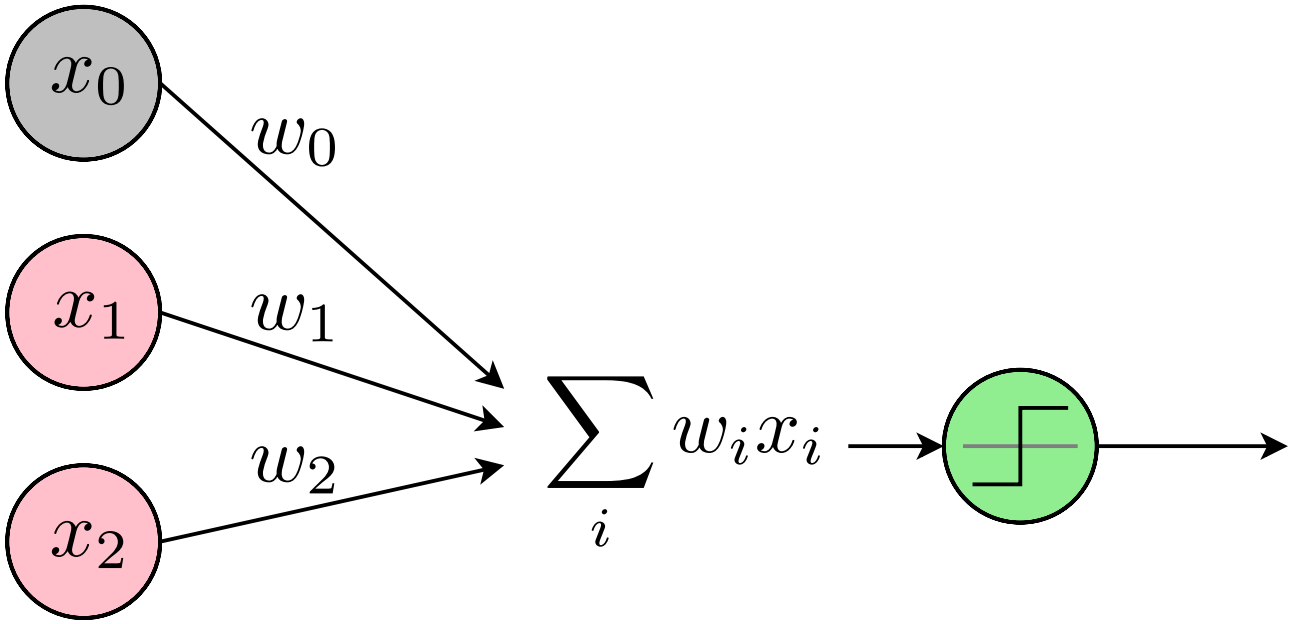
A Neuron



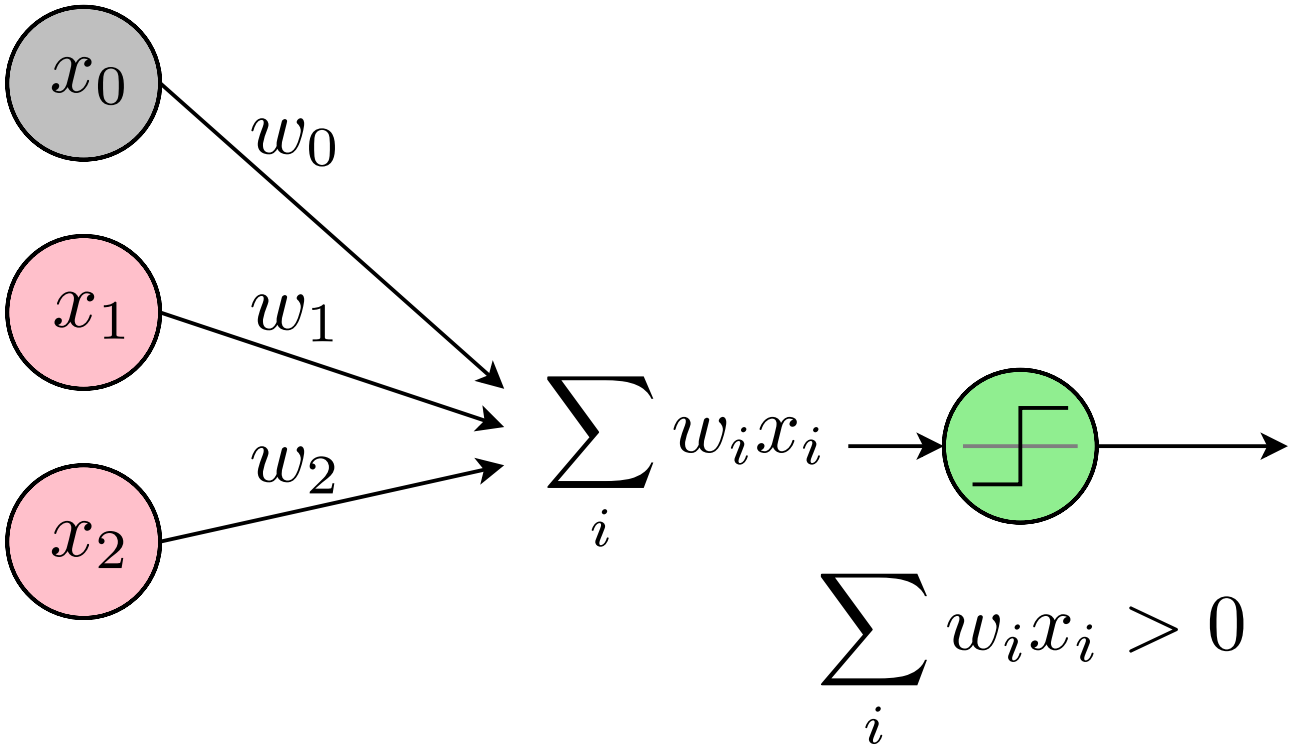
A Neuron


x_0
x_1
x_2
x_3
x_4
w_0
w_1
w_2
w_3
w_4
$\sum_i w_i x_i$
Sigmoid,
Tanh,
Other Non-Linearity

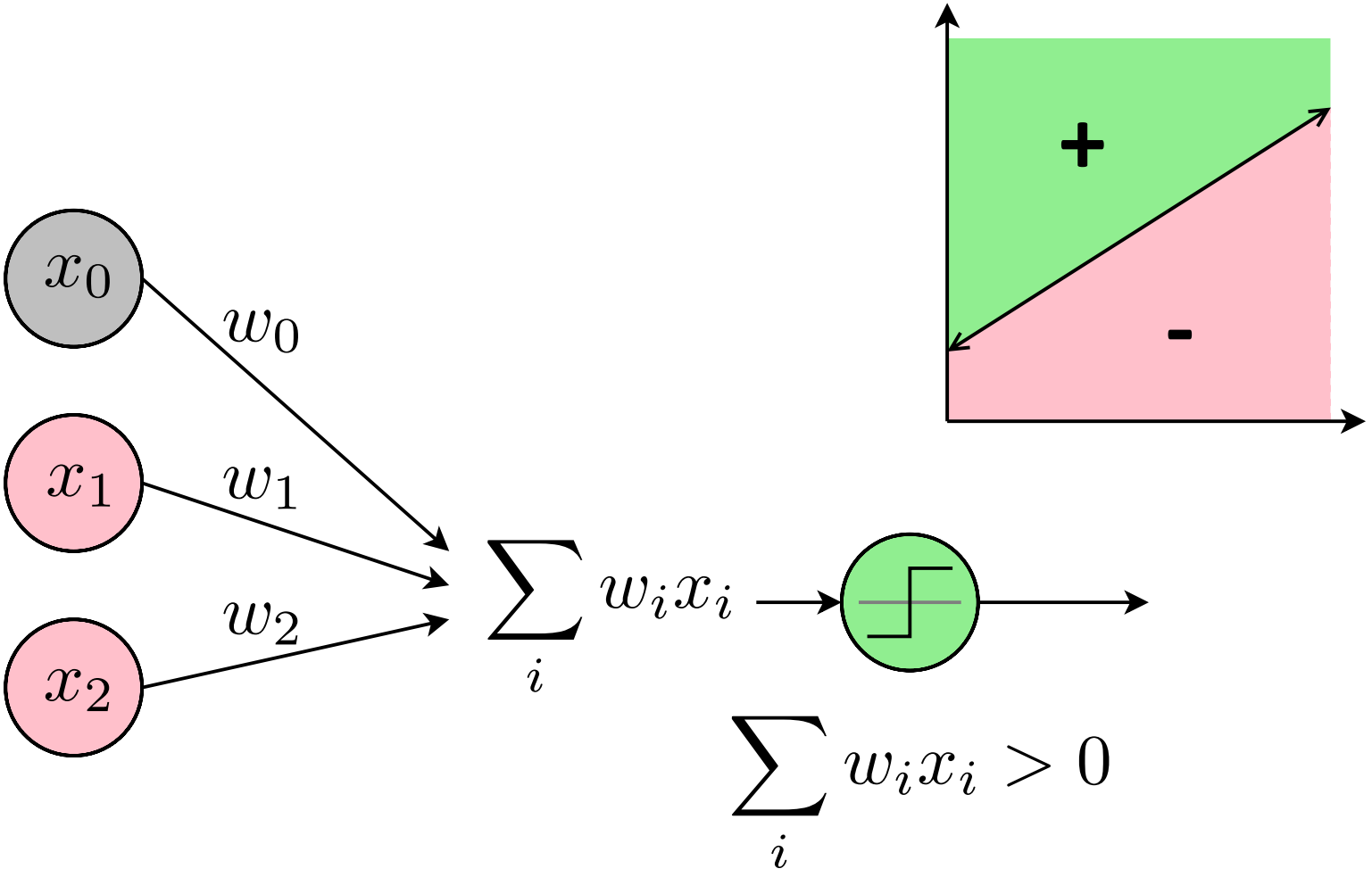
Perceptron : An Early Idea (1950)



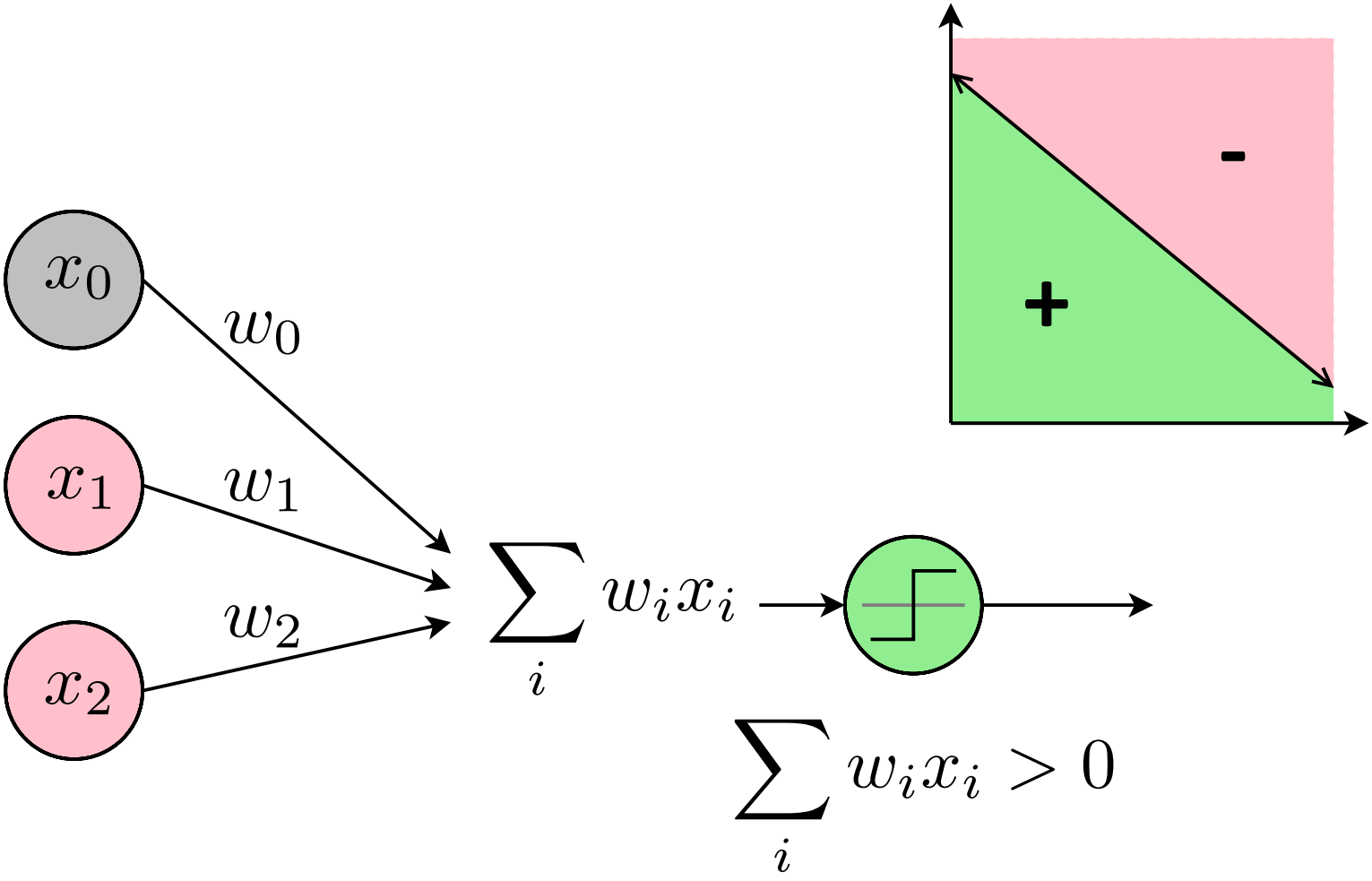
Perceptron : An Early Idea (1950)



Perceptron : An Early Idea (1950)

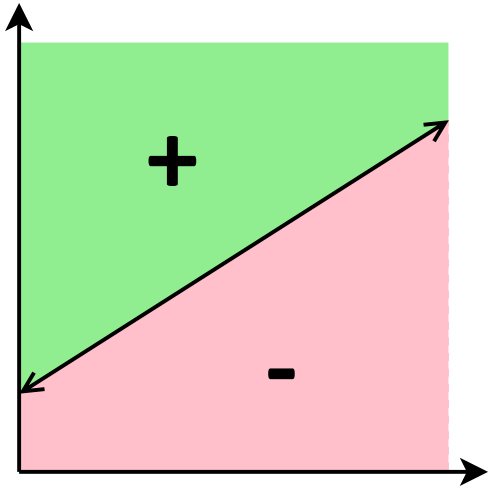


Another Perceptron

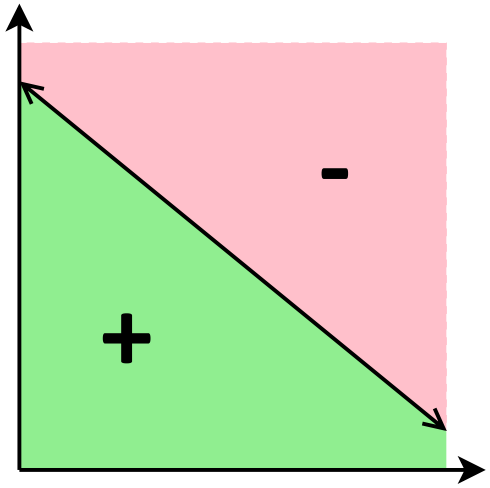


More Complex Decision Boundary?

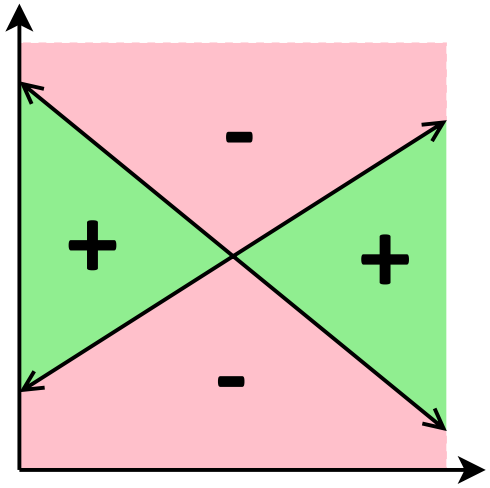
Yes



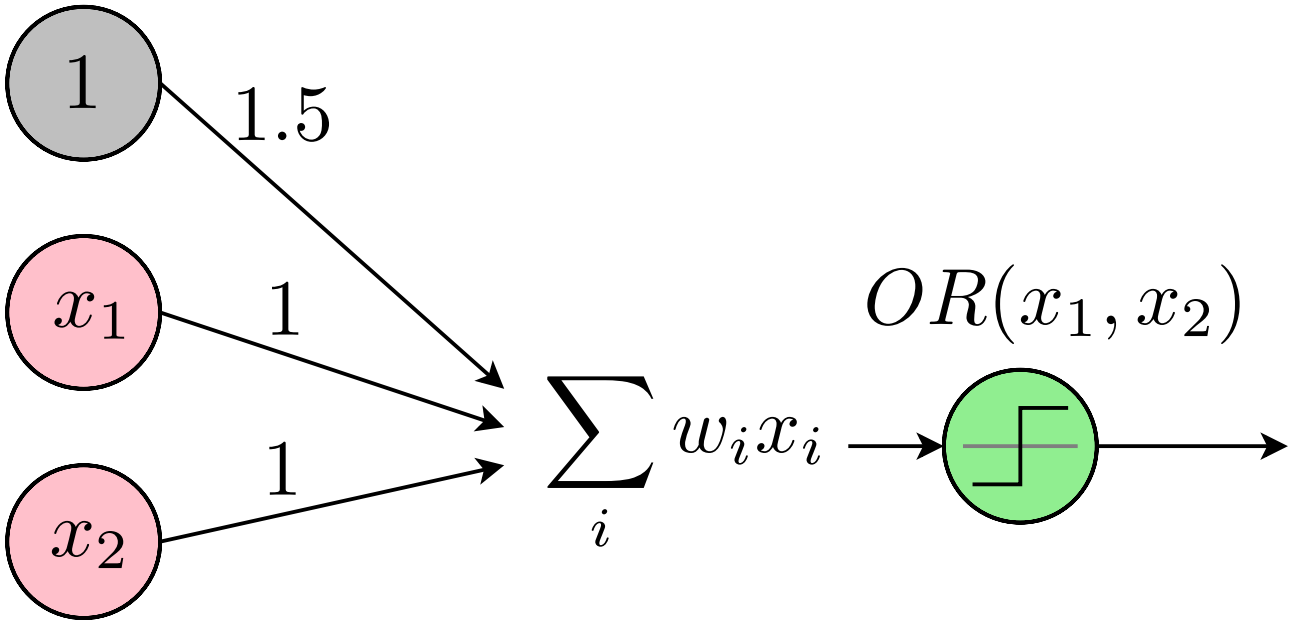
Yes



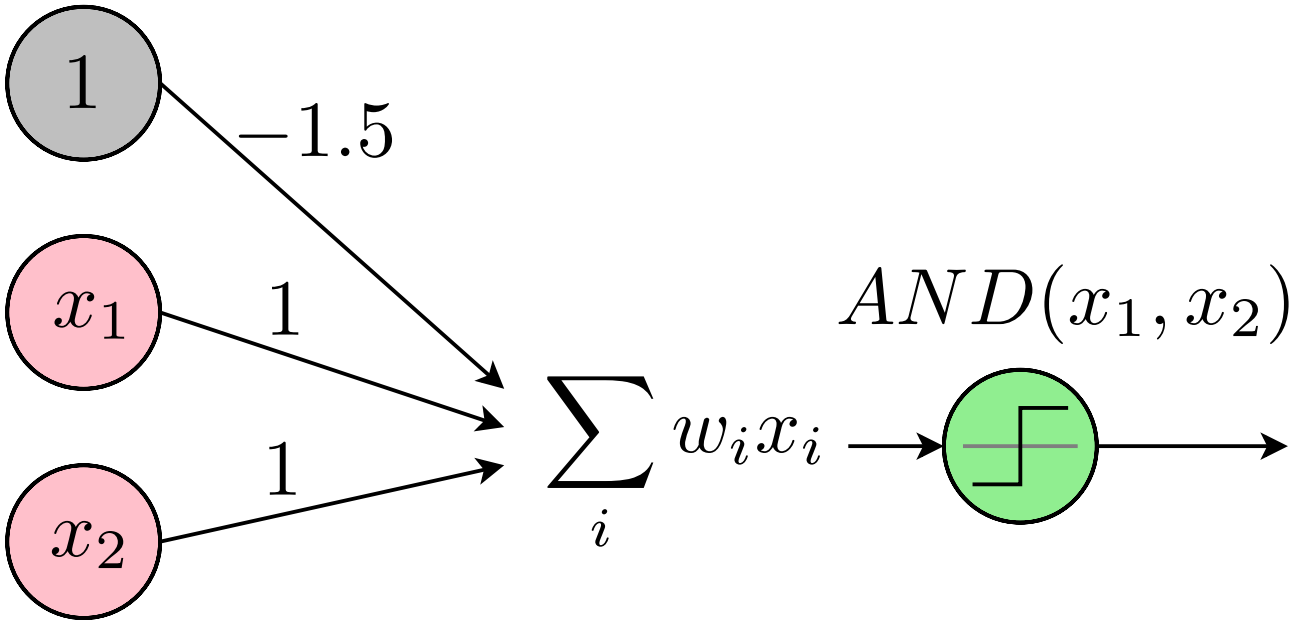
???



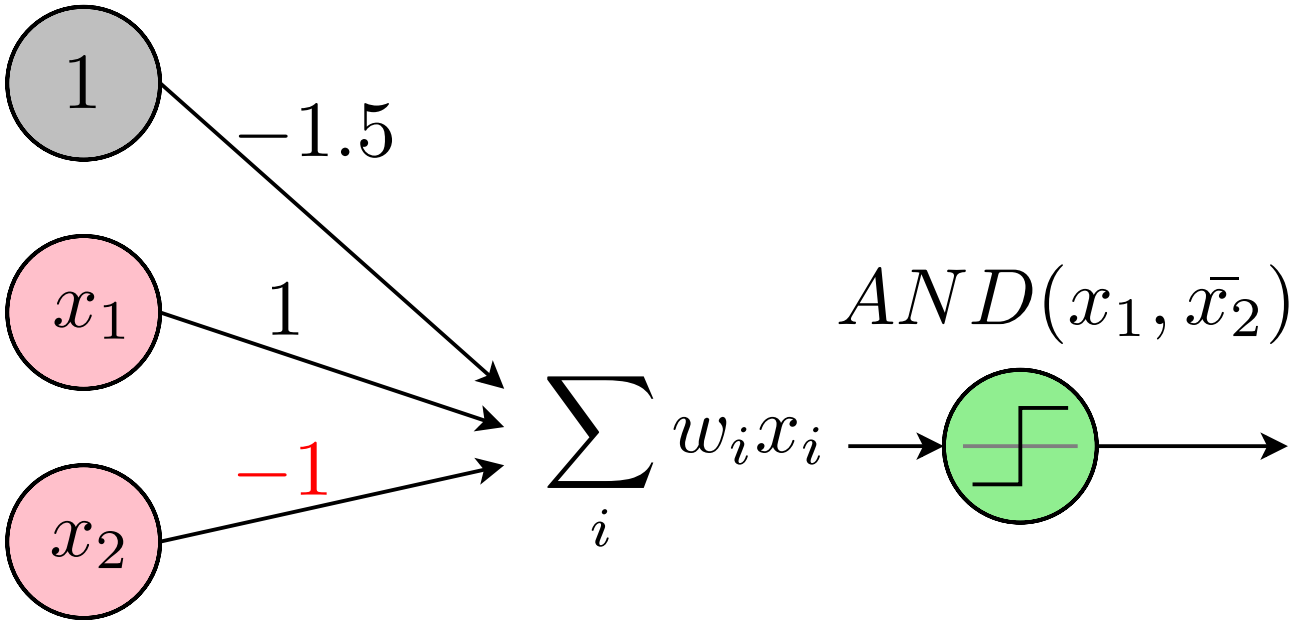
Perceptron : OR(x1,x2)



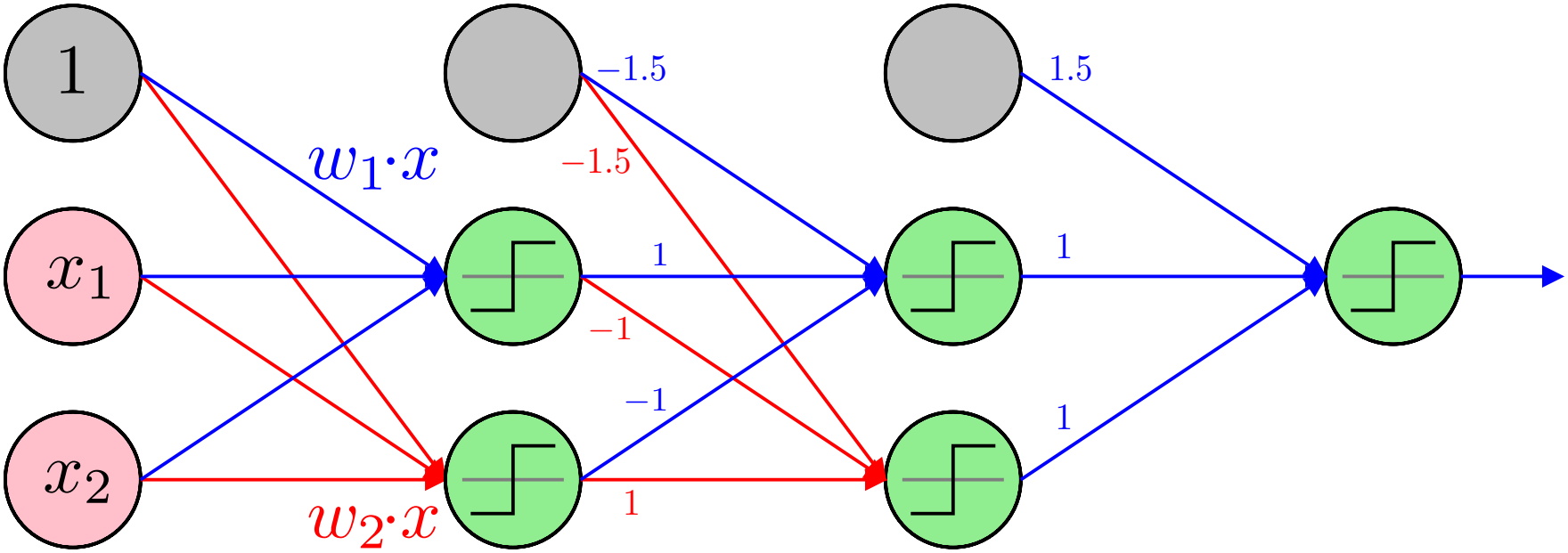
Perceptron : AND(x1,x2)



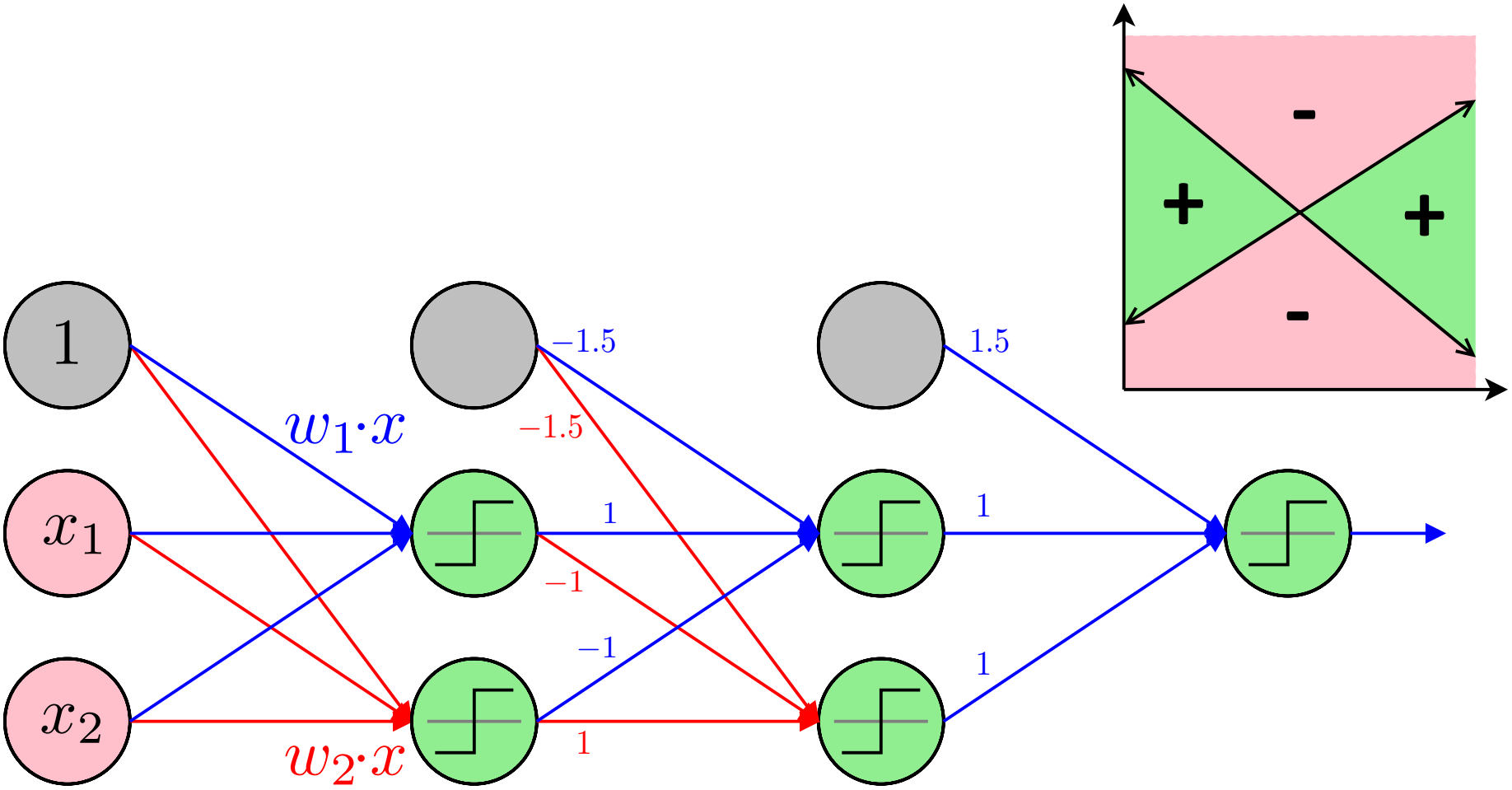
Perceptron : AND(x1,~~x2~~)



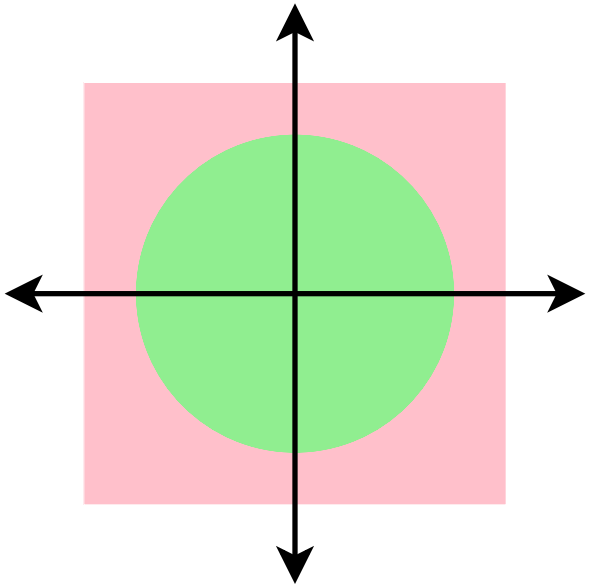
Multi Layer Perceptron



Multi Layer Perceptron

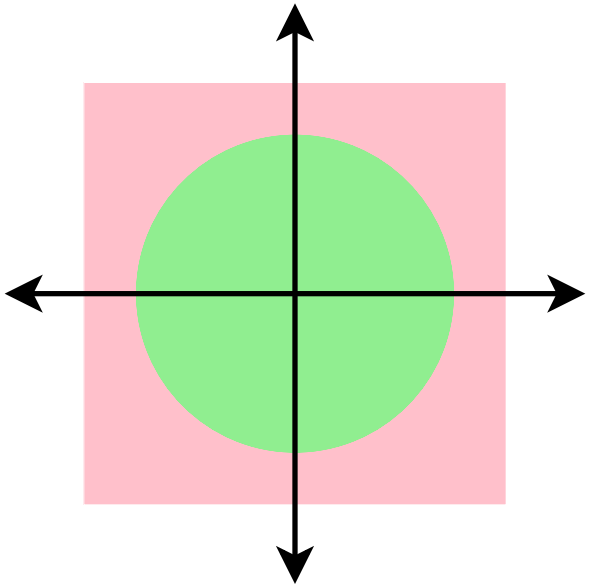


A Powerful Model

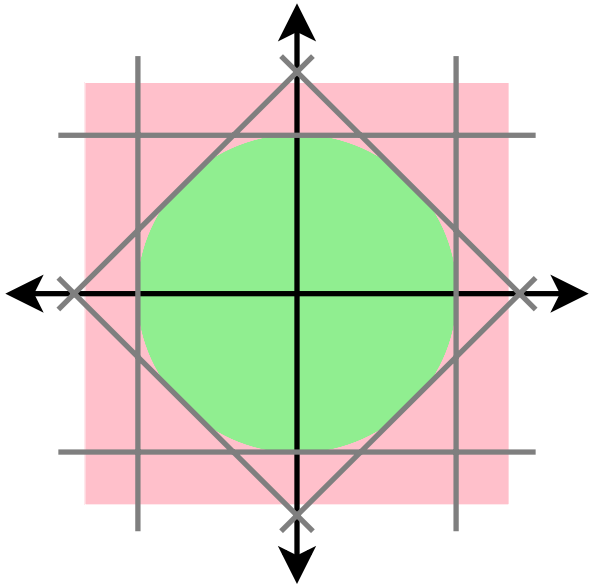


Goal

A Powerful Model

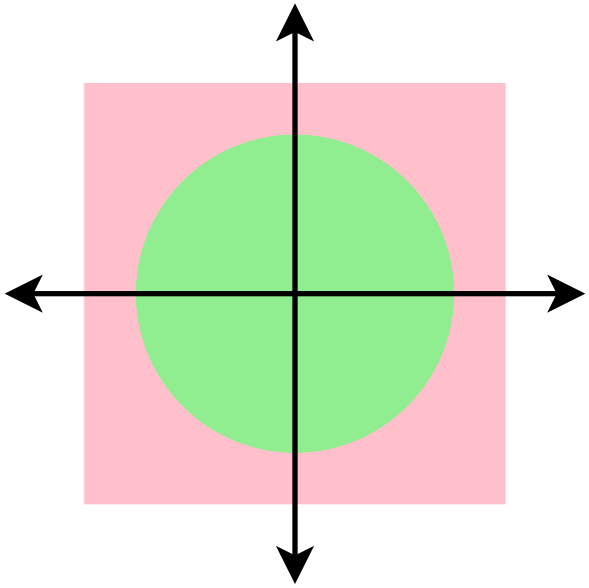


Goal

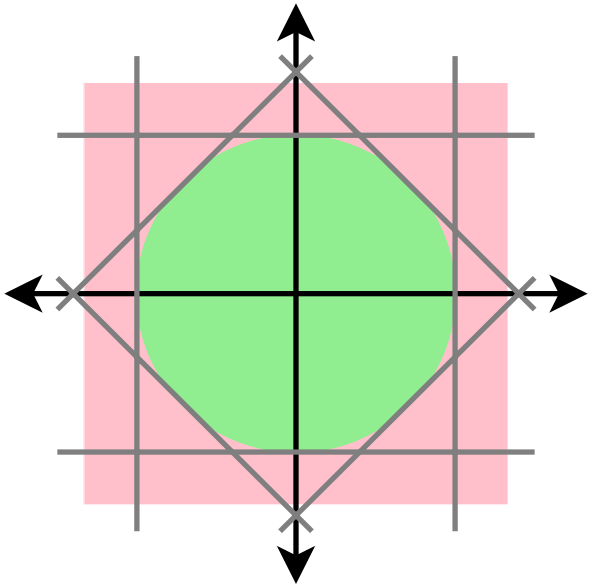


8 perceptrons

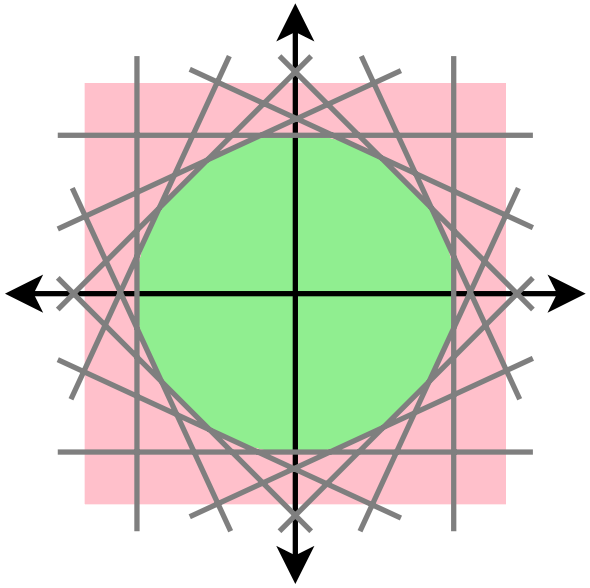
A Powerful Model



Goal

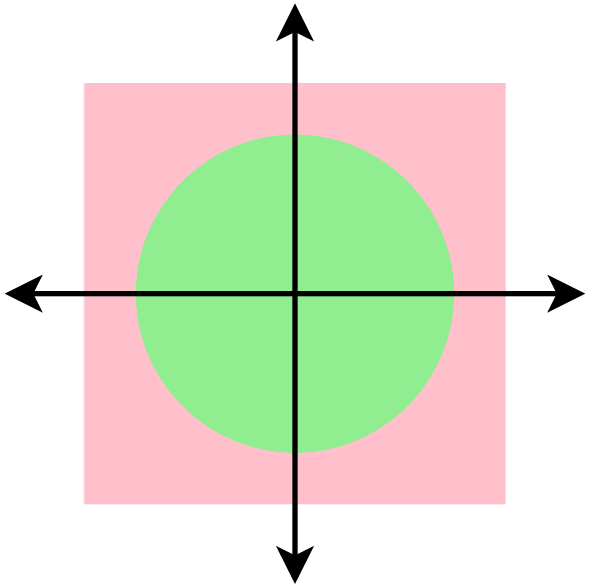


8 perceptrons

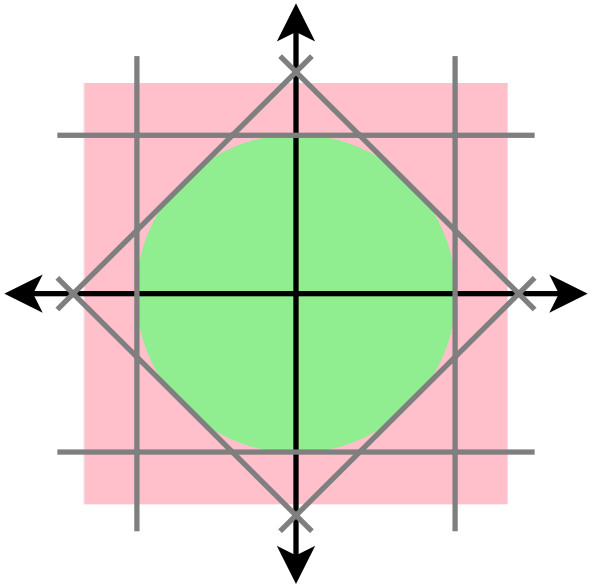


16 perceptrons

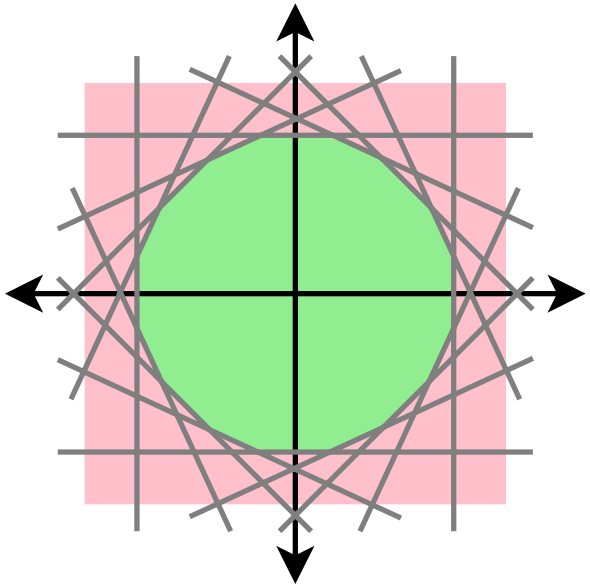
Key Idea 1 : A Powerful Model



Goal

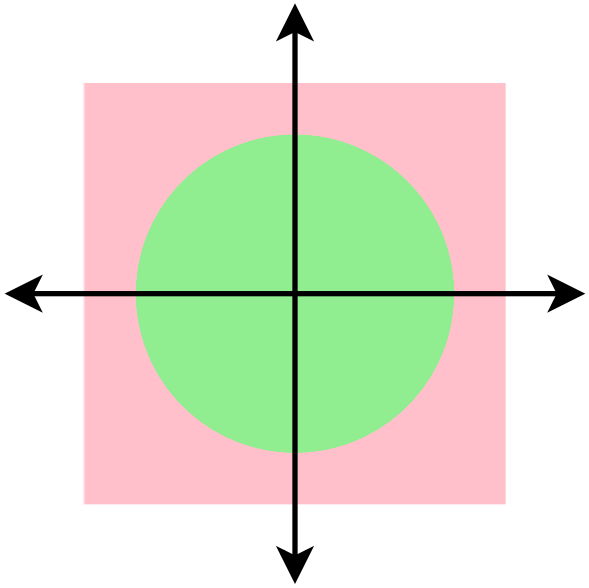


8 perceptrons

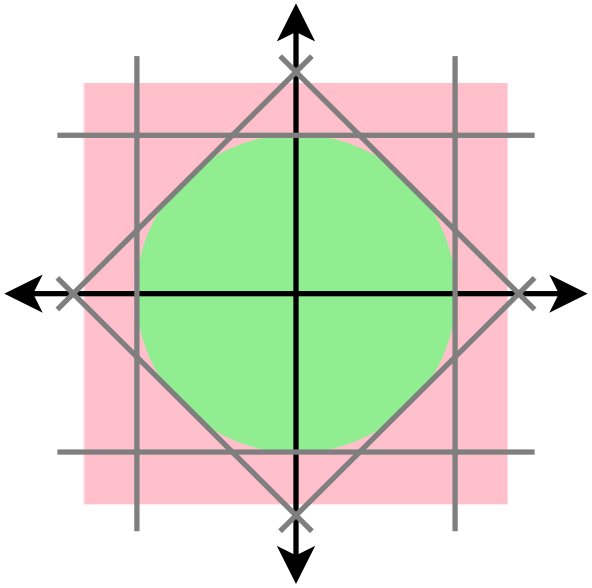


16 perceptrons

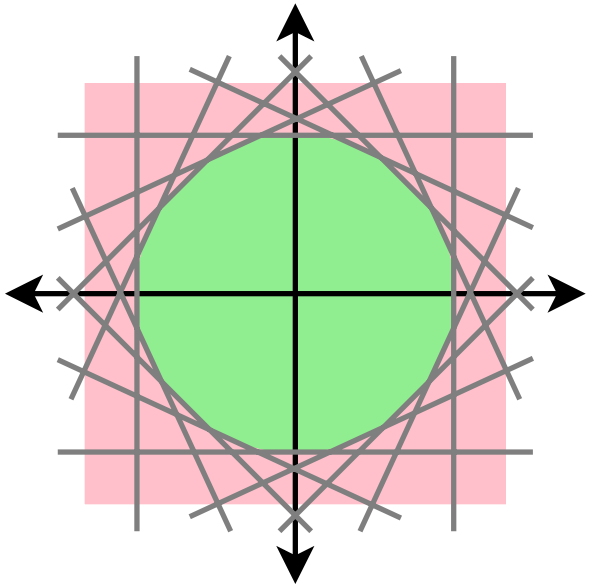
A Powerful Model



Goal



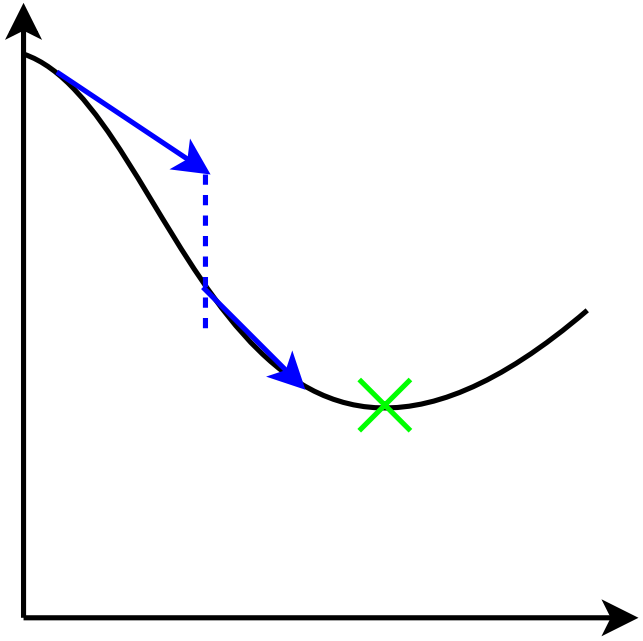
8 perceptrons

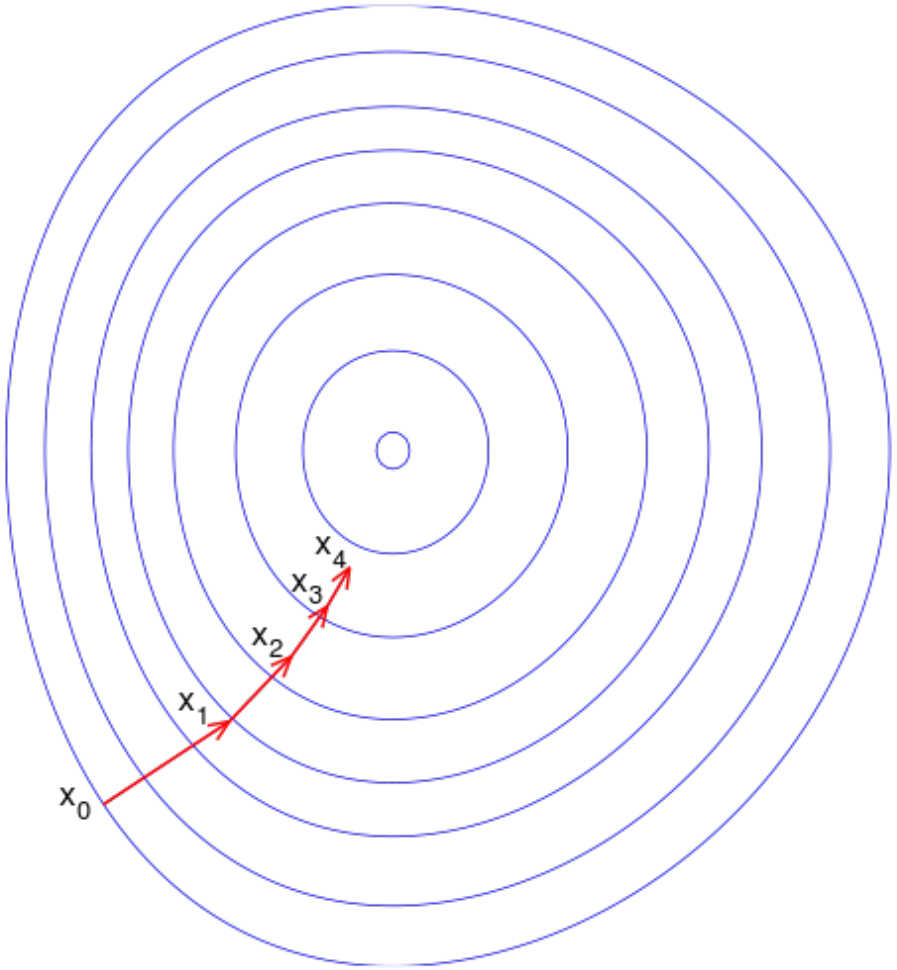


16 perceptrons

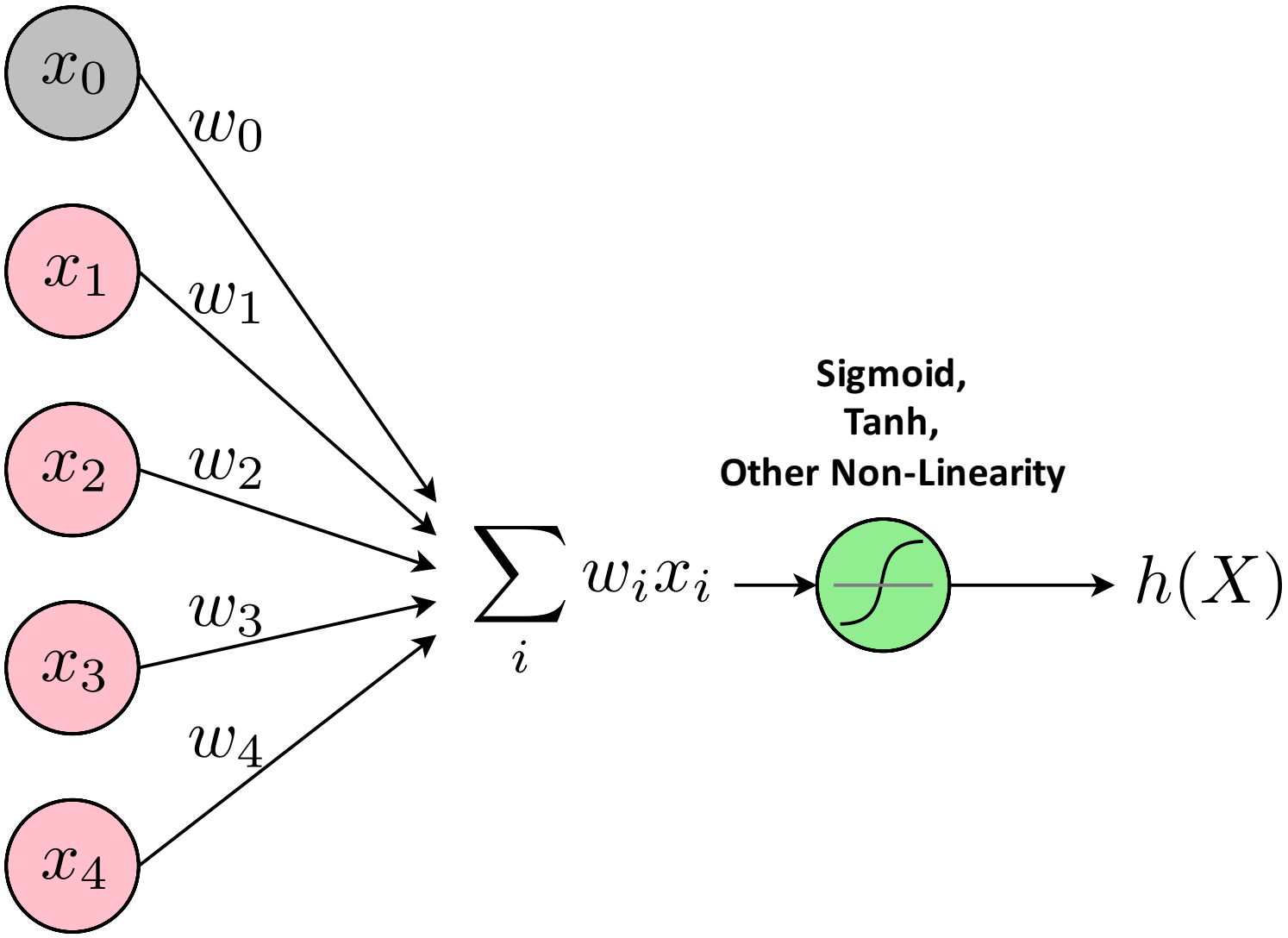
Danger Will Robinson : Red flags : generalization & optimization

Gradient Descent

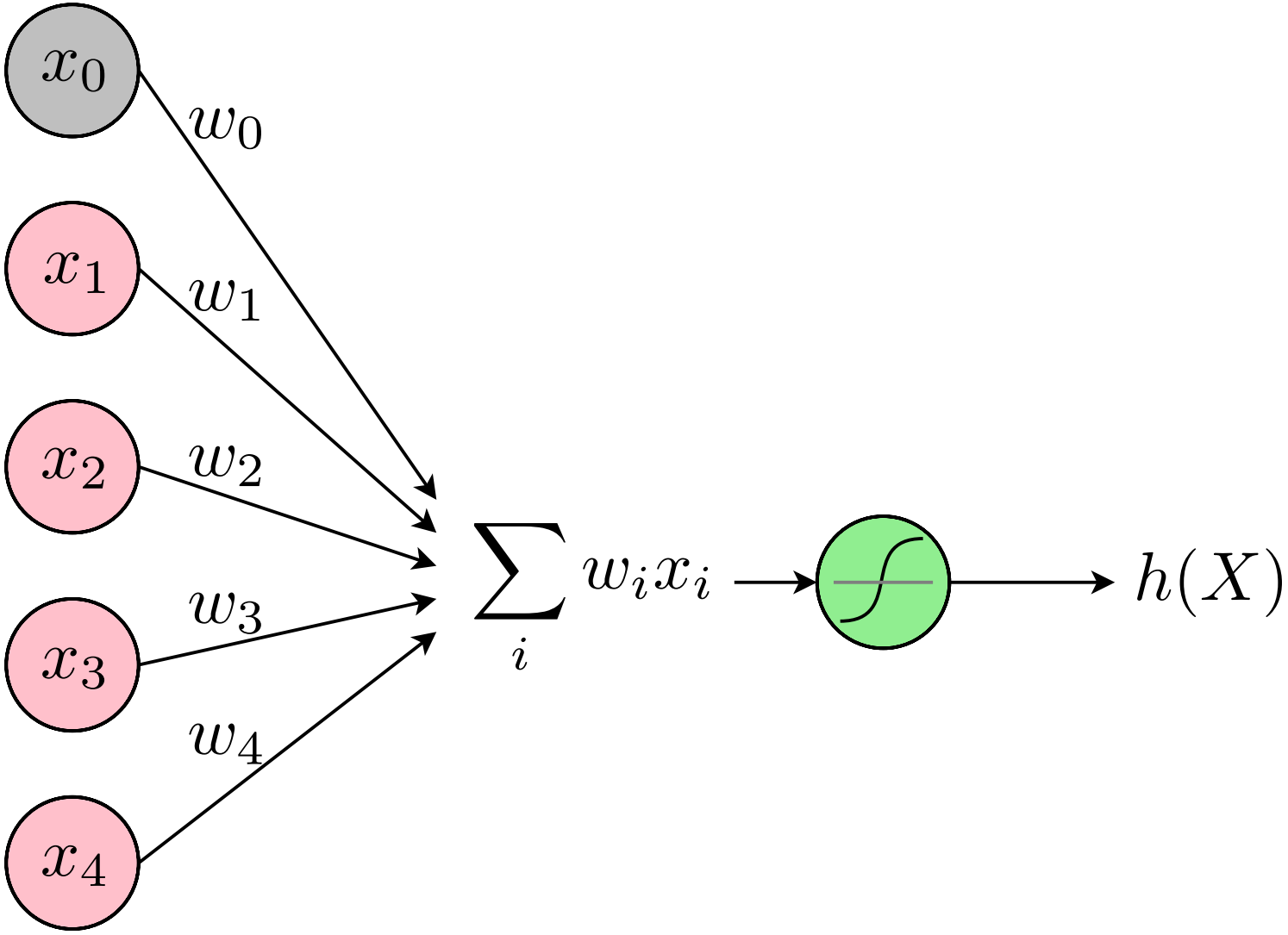




Smooth Thresholding

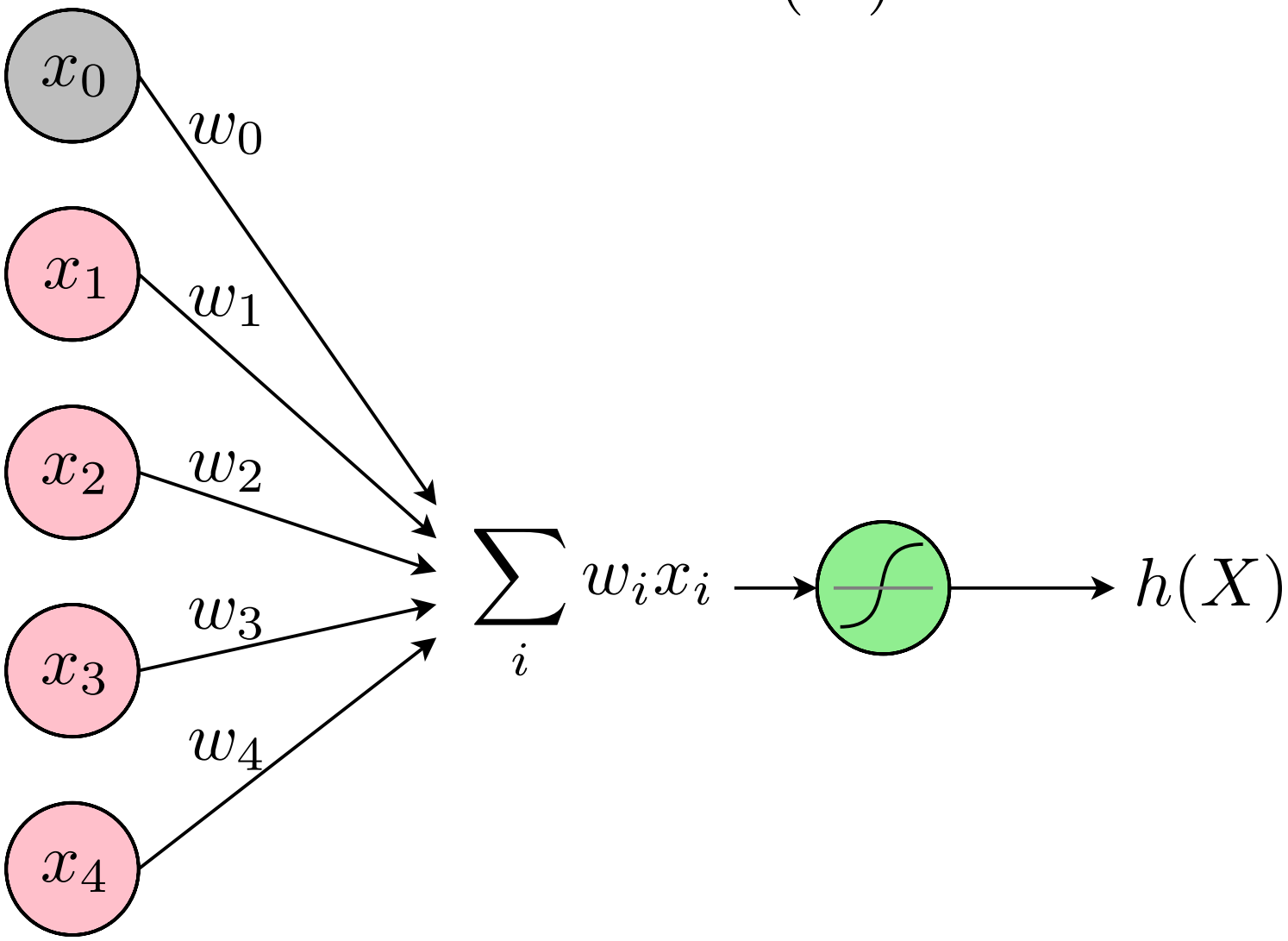


Gradient Descent



Gradient Descent

$h(X)$ is determined entirely by w_i's

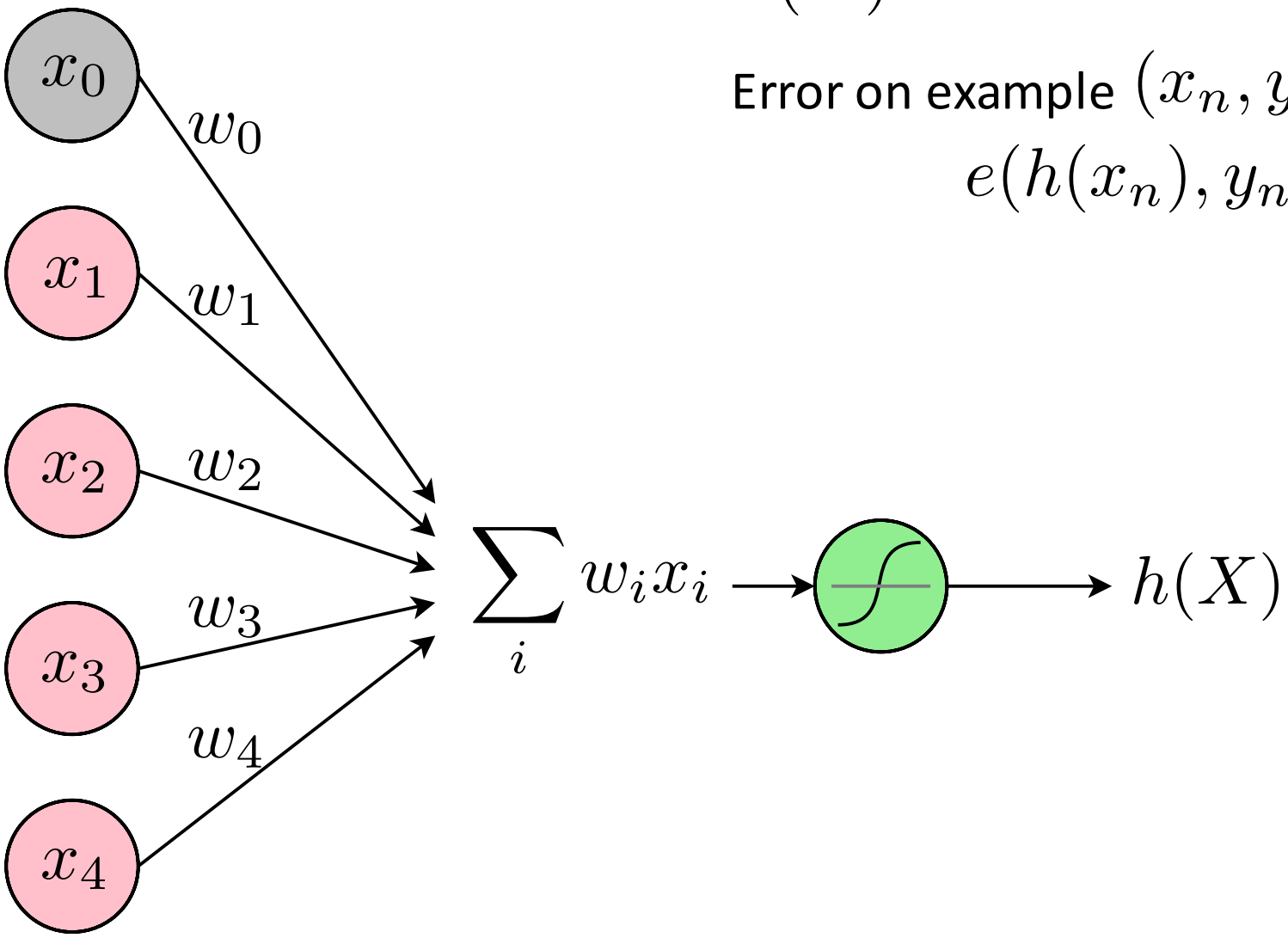


Gradient Descent

$h(X)$ is determined entirely by w_i's

Error on example (x_n, y_n) is

$$e(h(x_n), y_n) = e(w)$$



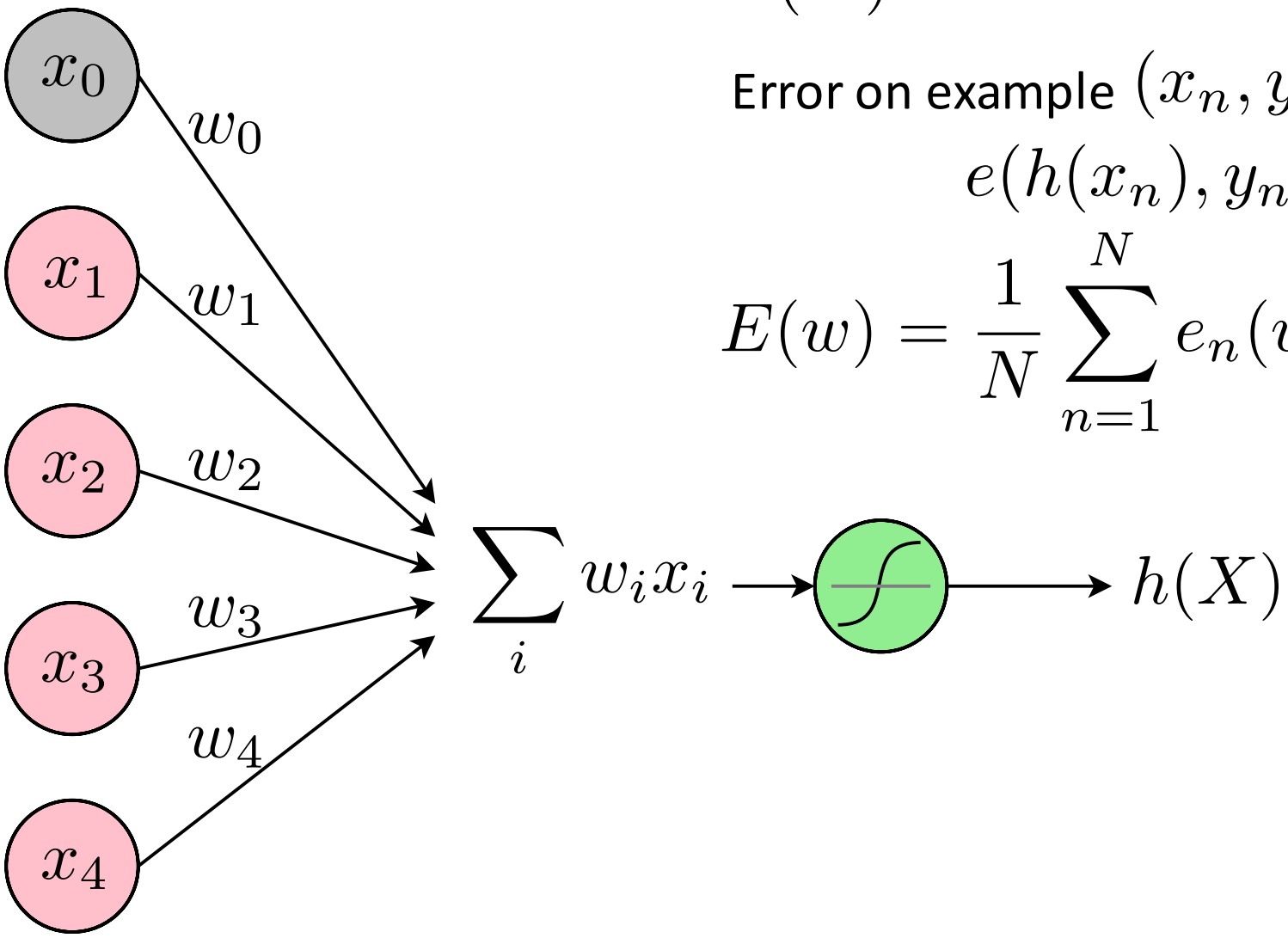
Gradient Descent

$h(X)$ is determined entirely by w_i's

Error on example (x_n, y_n) is

$$e(h(x_n), y_n) = e(w)$$

$$E(w) = \frac{1}{N} \sum_{n=1}^{N} e_n(w)$$



Gradient Descent

$h(X)$ is determined entirely by w_i's

Error on example (x_n, y_n) is

$$e(h(x_n), y_n) = e(w)$$

Gradient Descent Minimizes : $$E(w) = \frac{1}{N} \sum_{n=1}^{N} e_n(w)$$

Gradient Descent

$h(X)$ is determined entirely by w_i's

Error on example (x_n, y_n) is

$$e(h(x_n), y_n) = e(w)$$

Gradient Descent Minimizes : $$E(w) = \frac{1}{N} \sum_{n=1}^{N} e_n(w)$$

By Iterative Steps Along : $-\Delta E$

Gradient Descent

$h(X)$ is determined entirely by w_i's

Error on example (x_n, y_n) is

$$e(h(x_n), y_n) = e(w)$$

Gradient Descent Minimizes : $$E(w) = \frac{1}{N} \sum_{n=1}^{N} e_n(w)$$

By Iterative Steps Along : $-\Delta E$

So the Changed Weights are: $\Delta w = \eta \Delta E(w)$

Gradient Descent

$h(X)$ is determined entirely by w_i's

Error on example (x_n, y_n) is

$$e(h(x_n), y_n) = e(w)$$

Gradient Descent Minimizes : $E(w) = \frac{1}{N} \sum_{n=1}^{N} e_n(w)$

By Iterative Steps Along : $-\Delta E$

So the Changed Weights are: $\Delta w = \eta \Delta E(w)$

$-\Delta E$ Depends on ALL the examples (x_n, y_n)

Stochastic Gradient Descent

Pick ONE (x_n, y_n) at a time

Stochastic Gradient Descent

Pick ONE (x_n, y_n) at a time

Apply GD to $e(h(x_n), y_n) = e(w)$

Stochastic Gradient Descent

Pick ONE (x_n, y_n) at a time

Apply GD to $e(h(x_n), y_n) = e(w)$

"Average" Direction:

$$-\Delta e(h(x_n), y_n)$$

Stochastic Gradient Descent

Pick ONE (x_n, y_n) at a time

Apply GD to $e(h(x_n), y_n) = e(w)$

"Average" Direction:

$$\mathbb{E}_n[-\Delta e(h(x_n), y_n)] = \frac{1}{N}\sum_{n=1}^{N} -e(h(x_n), y_n)$$

Stochastic Gradient Descent

Pick ONE (x_n, y_n) at a time

Apply GD to $e(h(x_n), y_n) = e(w)$

"Average" Direction:

$$\mathbb{E}_n[-\Delta e(h(x_n), y_n)] = \frac{1}{N}\sum_{n=1}^{N} -e(h(x_n), y_n)$$

$$= -\Delta E$$

Stochastic Gradient Descent

Pick ONE (x_n, y_n) at a time

Apply GD to $e(h(x_n), y_n) = e(w)$

"Average" Direction:

$$\mathbb{E}_n[-\Delta e(h(x_n), y_n)] = \frac{1}{N}\sum_{n=1}^{N} -e(h(x_n), y_n)$$

$$= -\Delta E$$

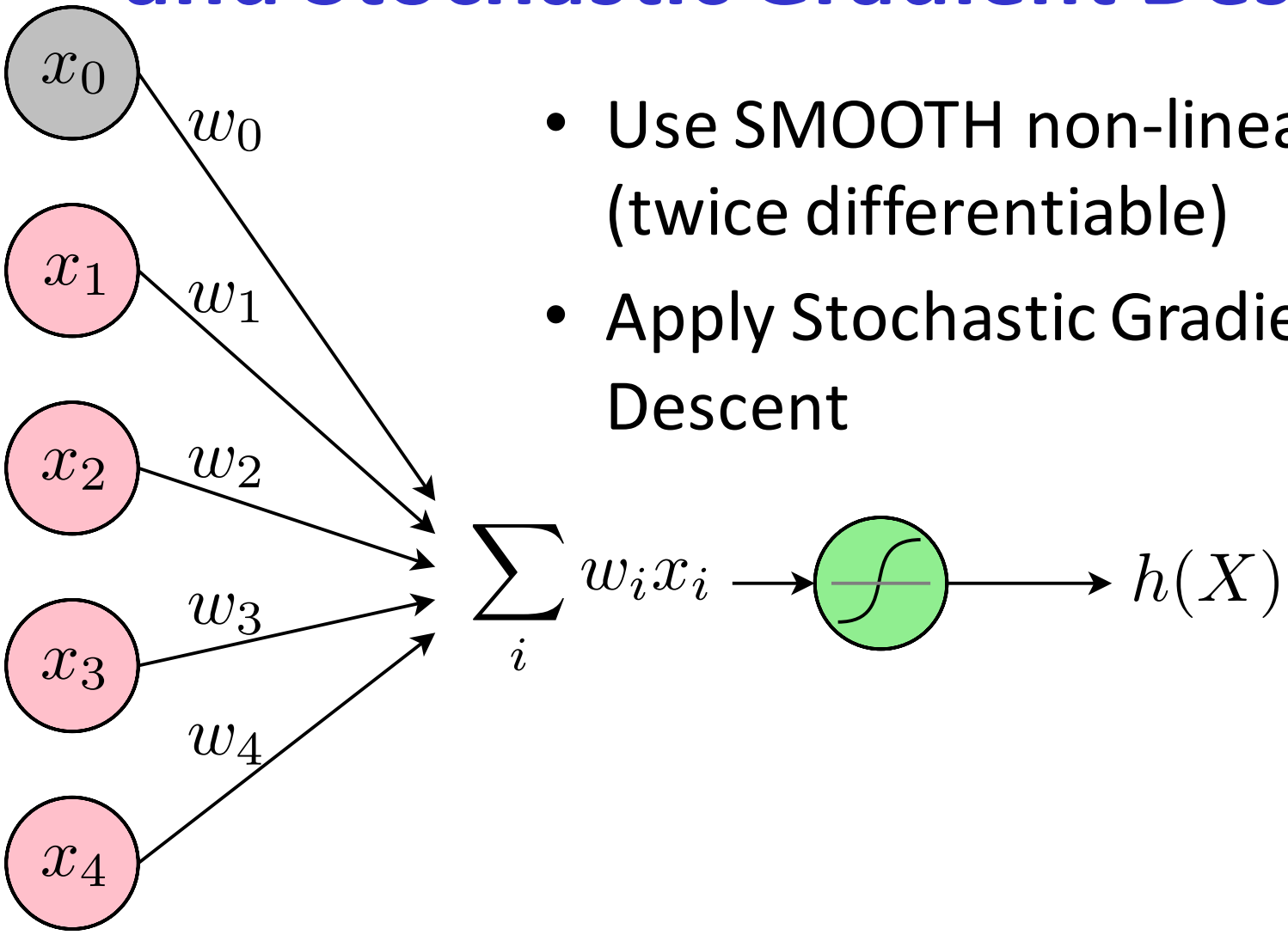
Randomized Gradient Descent : "Stochastic Gradient Descent (SGD)"

Stochastic Gradient Descent
A Story to Remember

Benefits of SGD

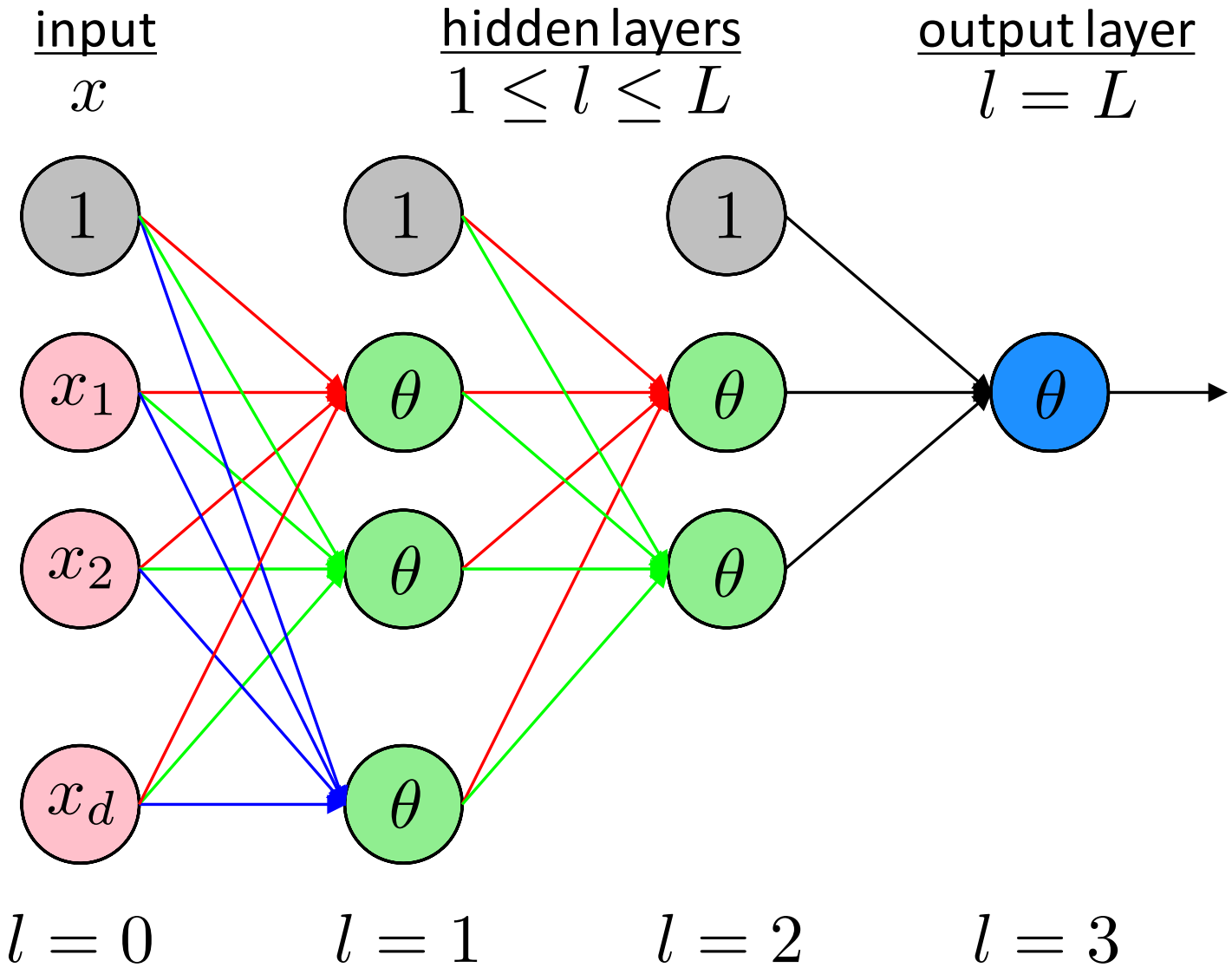
- Faster computation (impatient)
- Randomization
- Simple

Key Idea 2 : Smooth Thresholding and Stochastic Gradient Descent

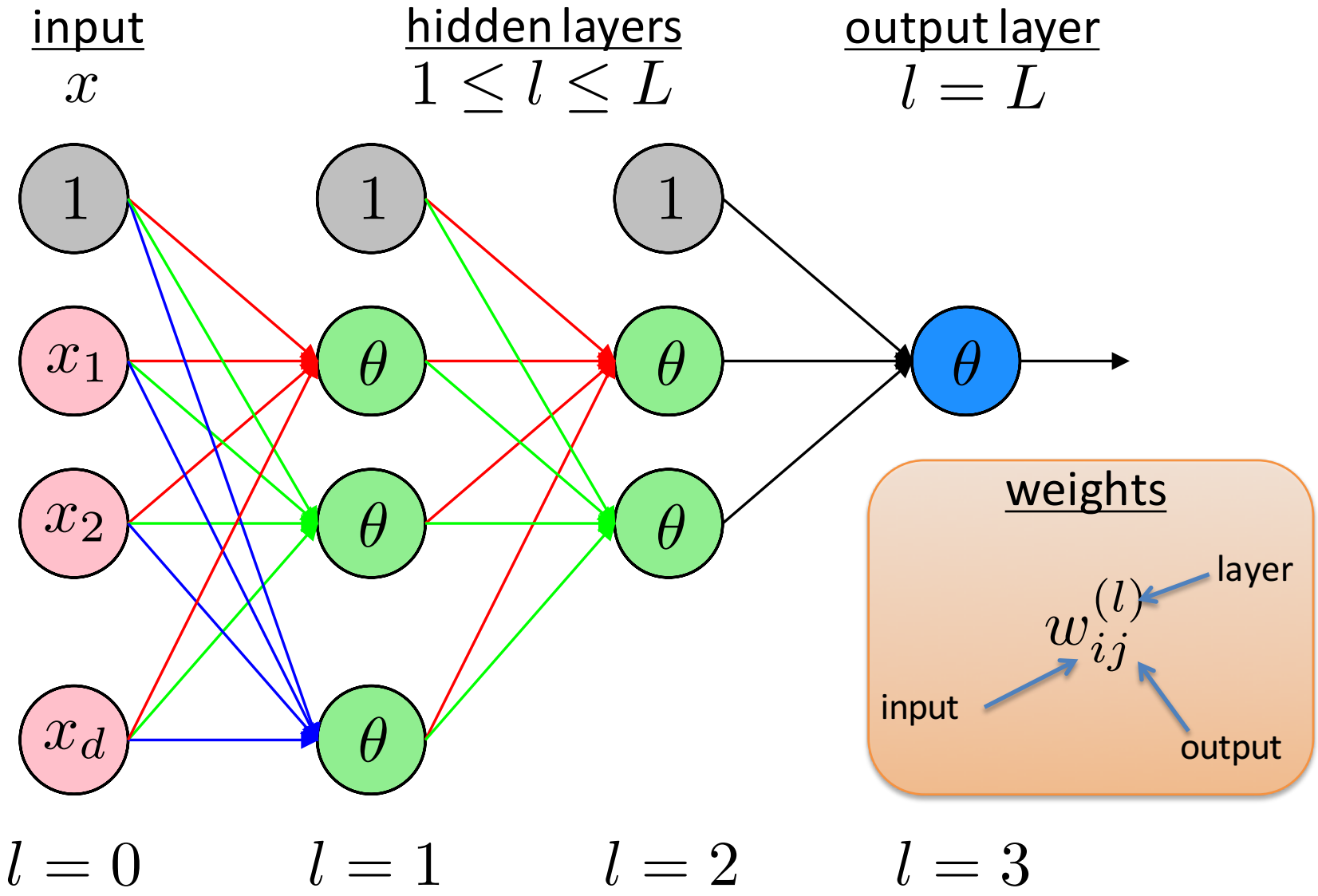


- Use SMOOTH non-linearity's (twice differentiable)
- Apply Stochastic Gradient Descent

Neural Network Notation



Neural Network Notation



Computing the Outputs

inputs	weights	output
$x_i^{(l-1)}$	$w_{ij}^{(l)}$	$x_j^{(l)}$

$$x_j^{(l)} = \theta(S_j^{(l)}) = \theta\left(\sum_{i=0}^{d^{(l-1)}} w_{ij}^{(l)} x_i^{(l-1)}\right)$$

Computing the Outputs

inputs	weights	output
$x_i^{(l-1)}$	$w_{ij}^{(l)}$	$x_j^{(l)}$

$$x_j^{(l)} = \theta(S_j^{(l)}) = \theta\left(\sum_{i=0}^{d^{(l-1)}} w_{ij}^{(l)} x_i^{(l-1)}\right)$$

Recursive definition

Computing the Outputs

inputs	weights	output
$x_i^{(l-1)}$	$w_{ij}^{(l)}$	$x_j^{(l)}$

$$x_j^{(l)} = \theta(S_j^{(l)}) = \theta\left(\sum_{i=0}^{d^{(l-1)}} w_{ij}^{(l)} x_i^{(l-1)}\right)$$

Recursive definition

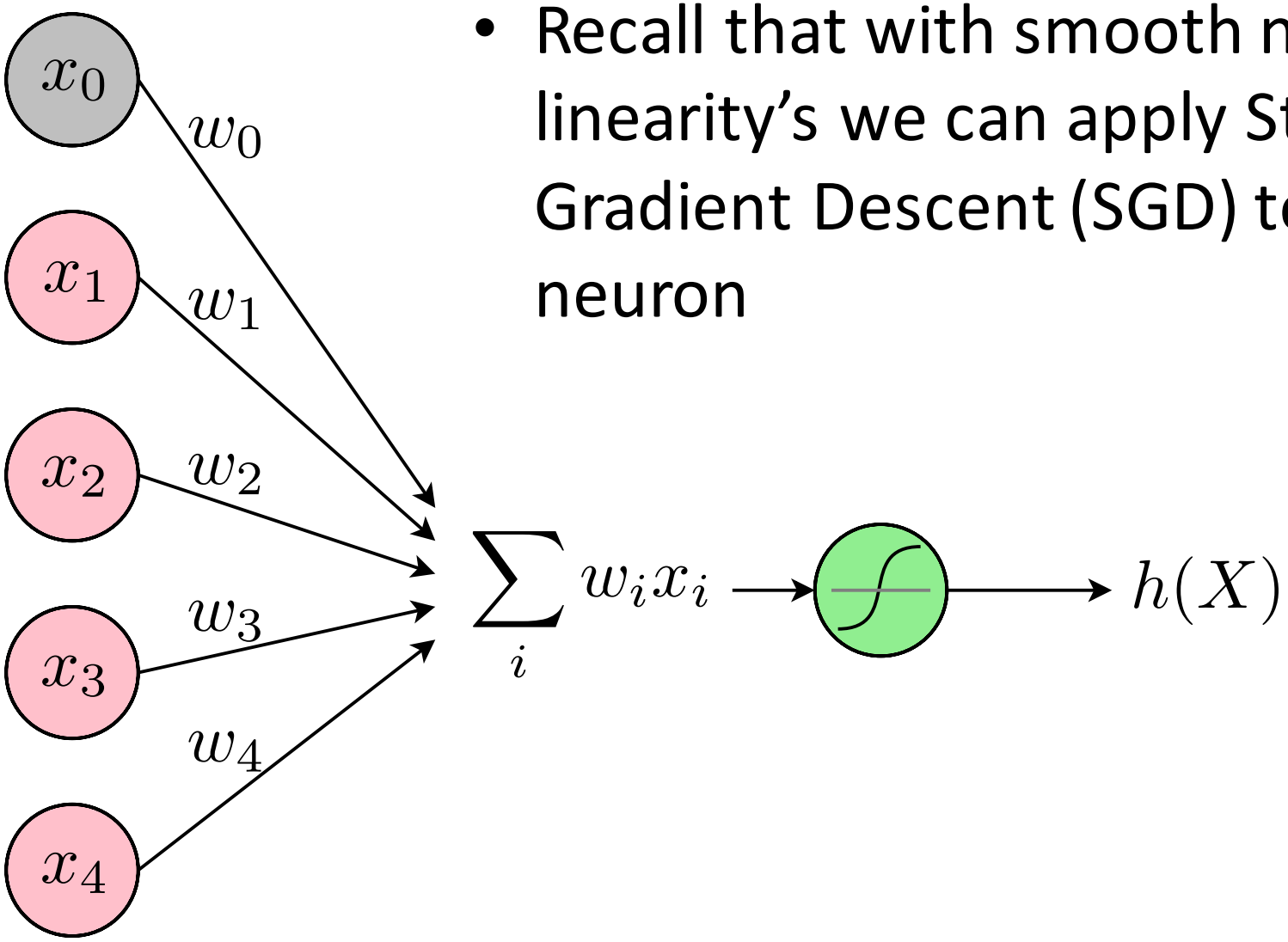
Just need those darn weights!

Recall – The Chain Rule

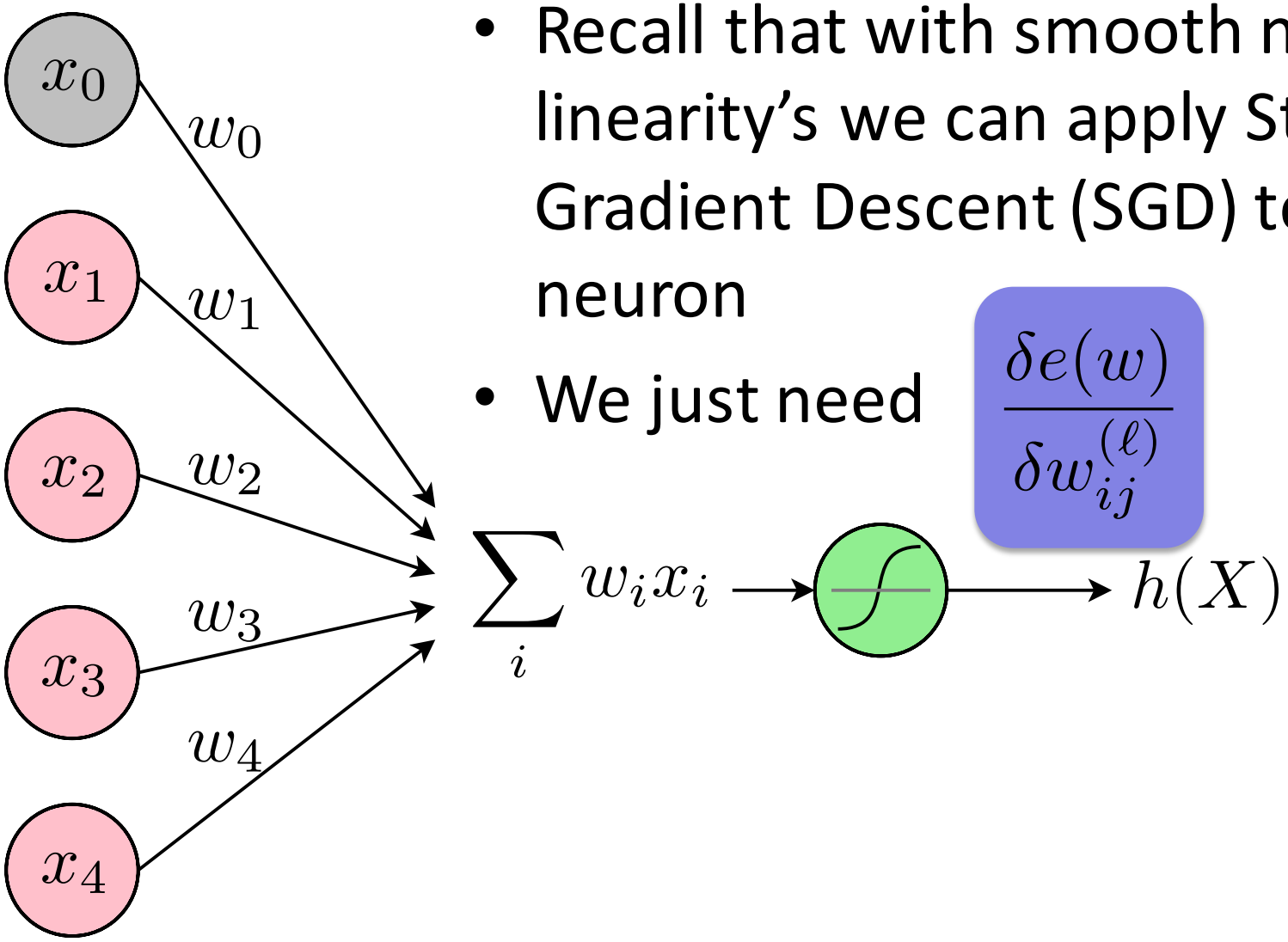
Recall – The Chain Rule Example

Key Idea 3 : Backpropagation

- Recall that with smooth non-linearity's we can apply Stochastic Gradient Descent (SGD) to train a neuron



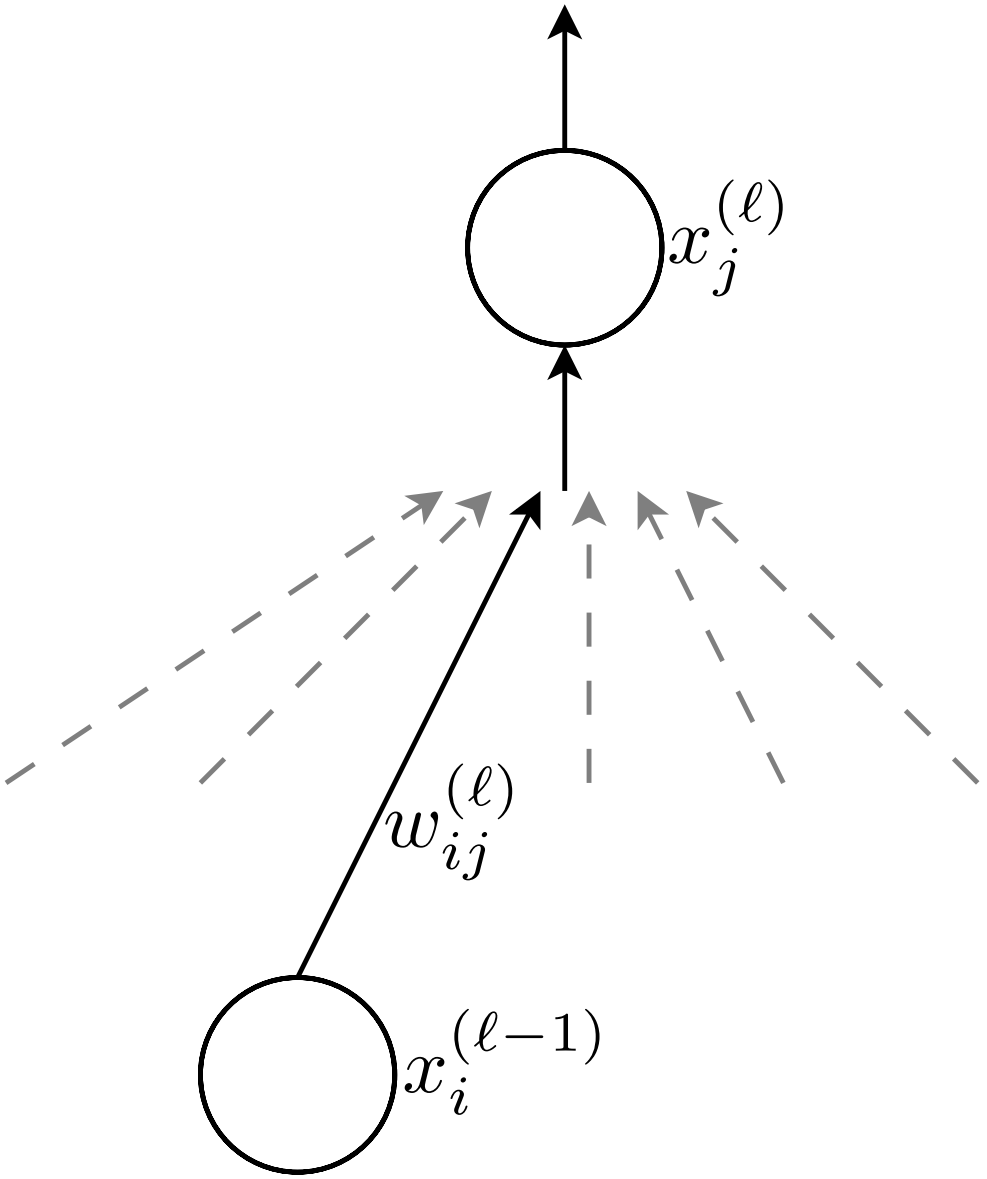
Key Idea 3 : Backpropagation



- Recall that with smooth non-linearity's we can apply Stochastic Gradient Descent (SGD) to train a neuron
- We just need

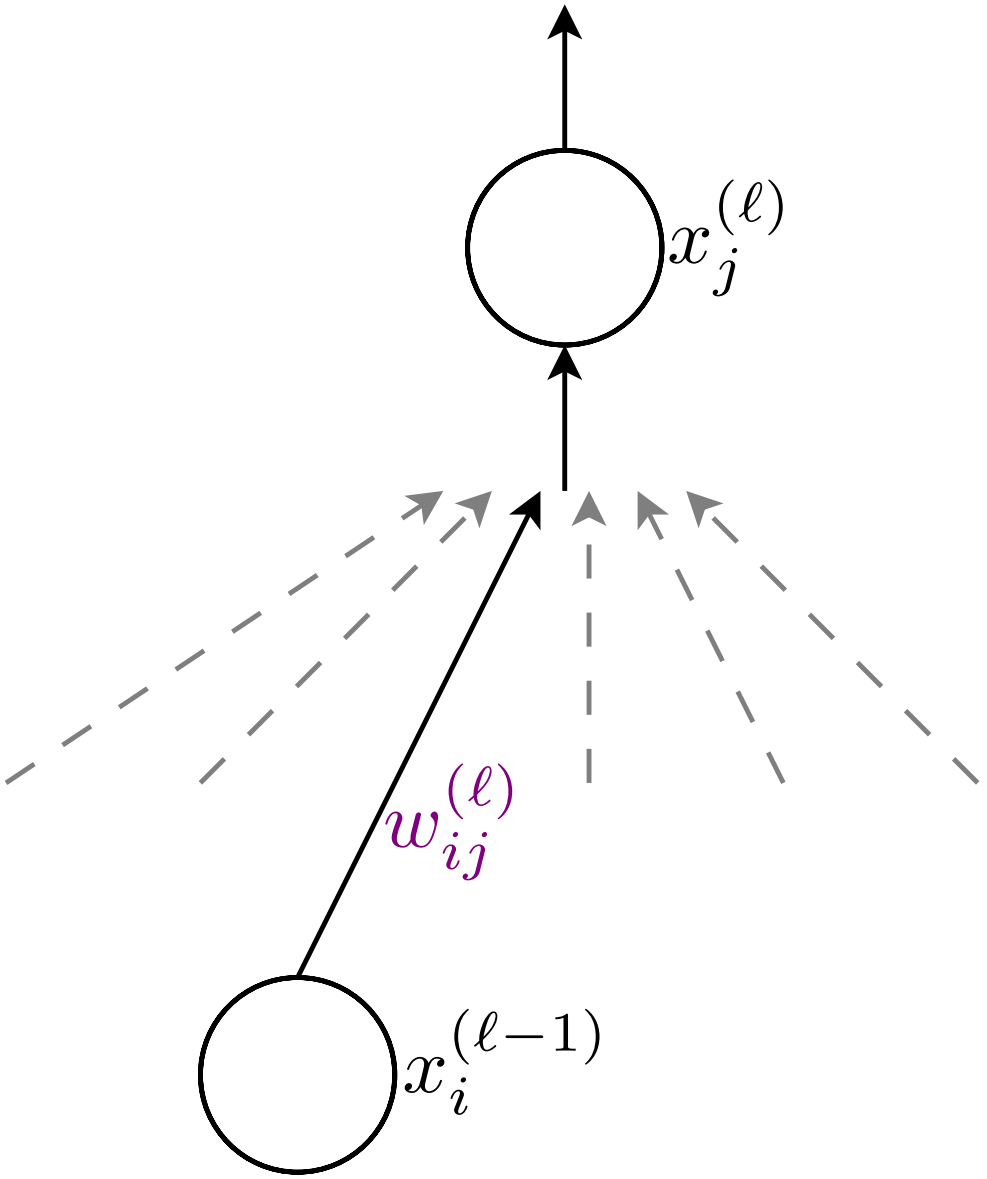
$$\frac{\delta e(w)}{\delta w_{ij}^{(\ell)}}$$

Partial Derivatives



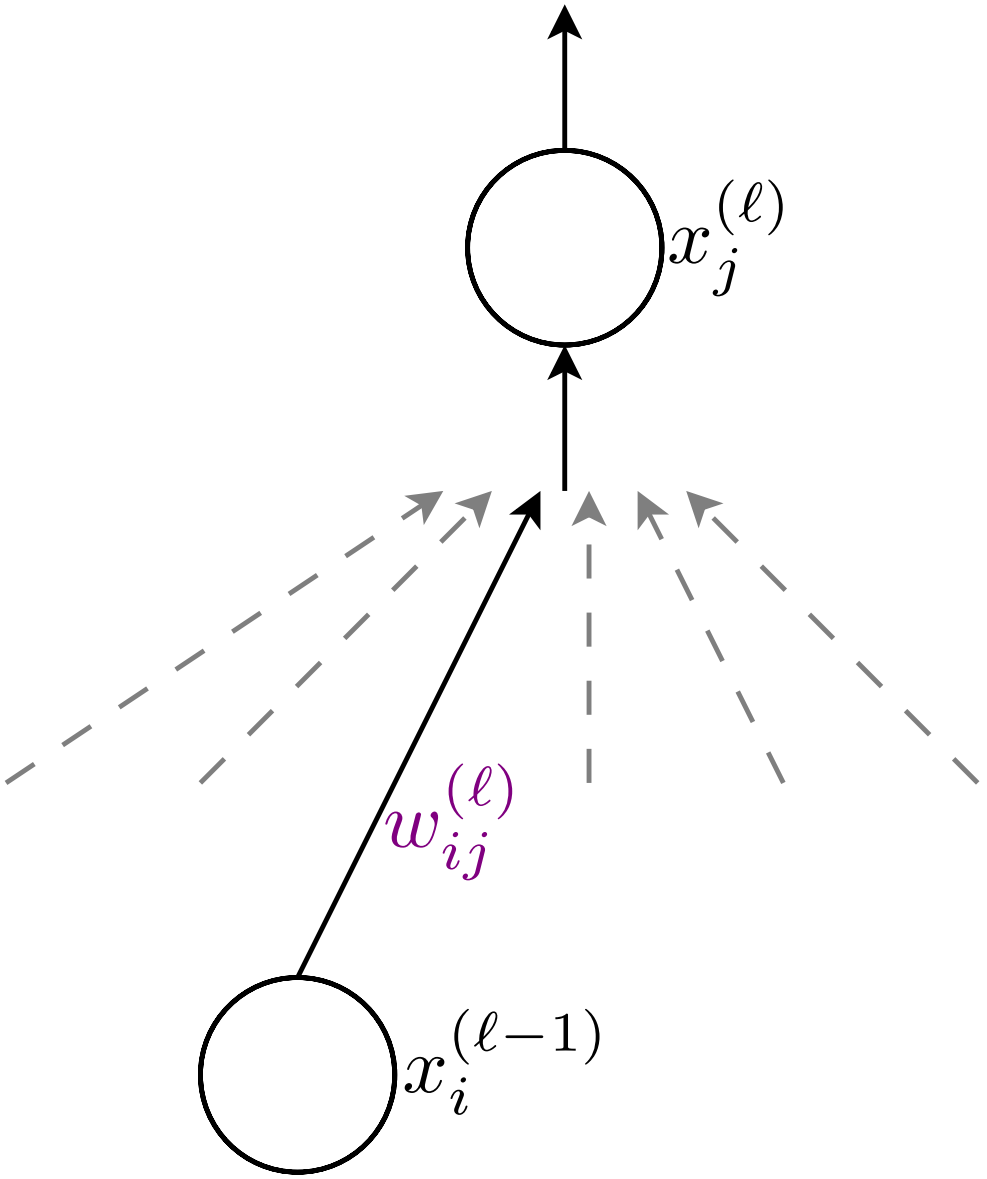
Partial Derivatives

$$\frac{\delta e(w)}{\delta w_{ij}^{(\ell)}}$$



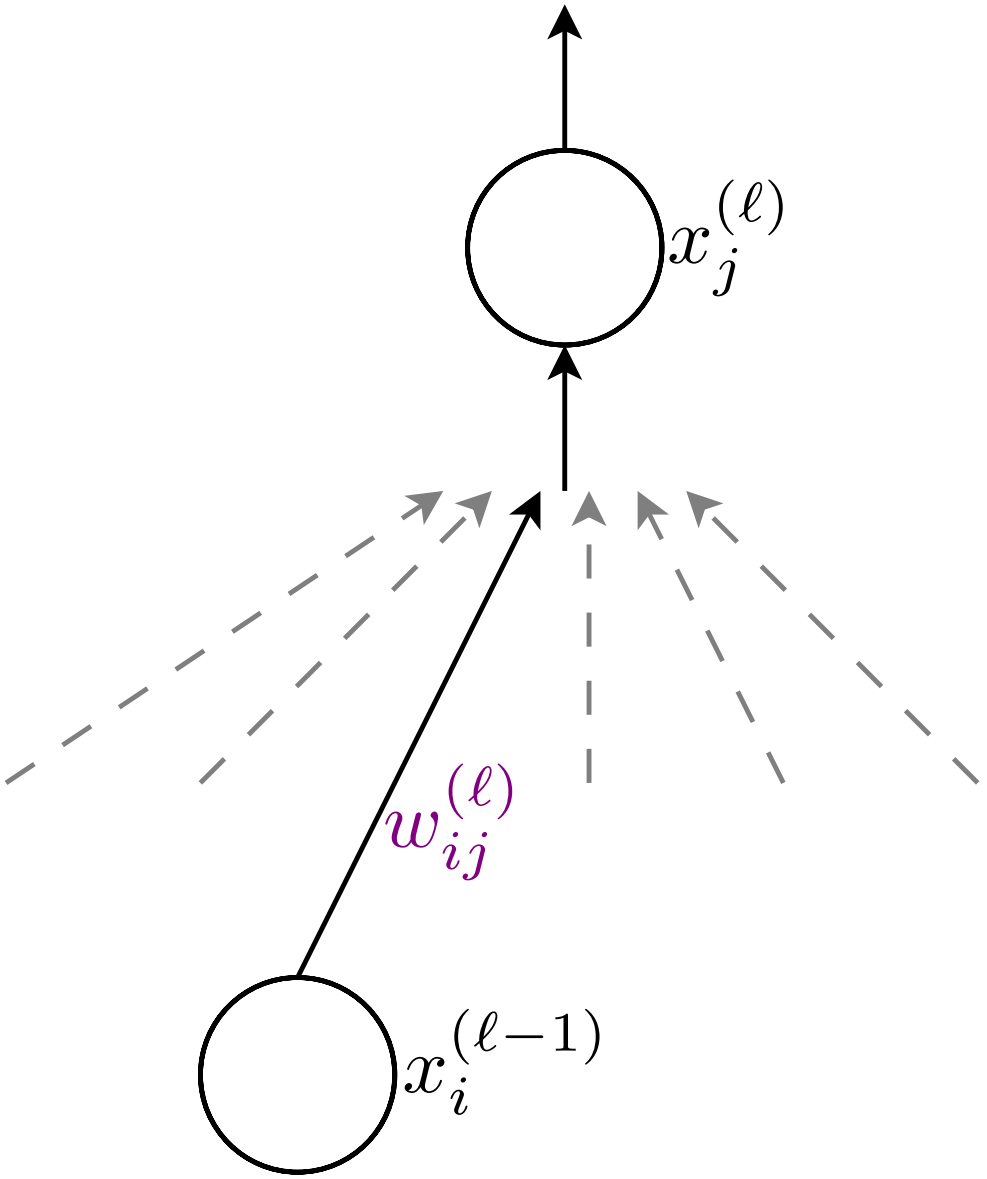
Partial Derivatives

$$\frac{\delta e(w)}{\delta w_{ij}^{(\ell)}}$$



Partial Derivatives

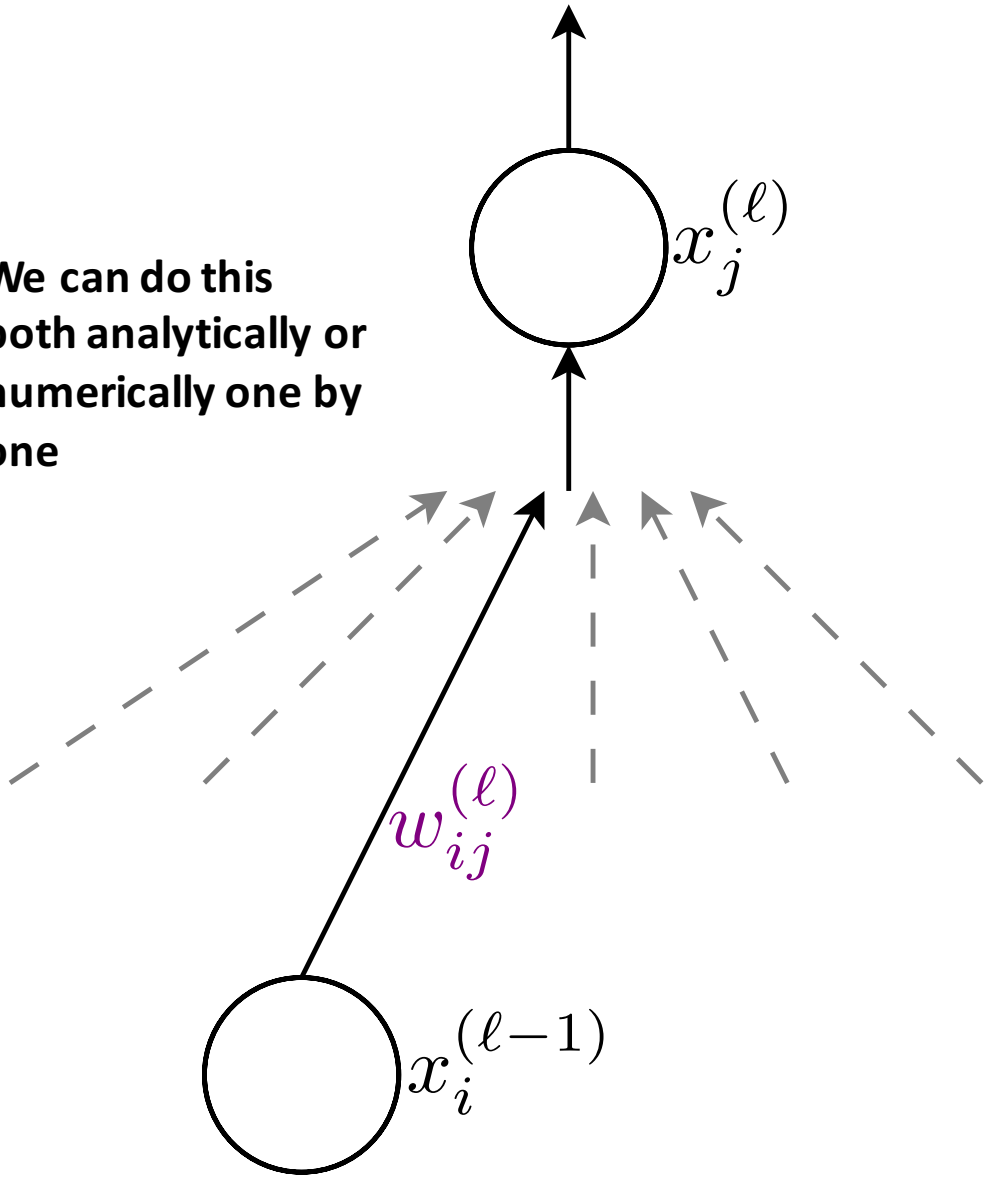
$$\frac{\delta e(w)}{\delta w_{ij}^{(\ell)}}$$



Partial Derivatives

$$\frac{\delta e(w)}{\delta w_{ij}^{(\ell)}}$$

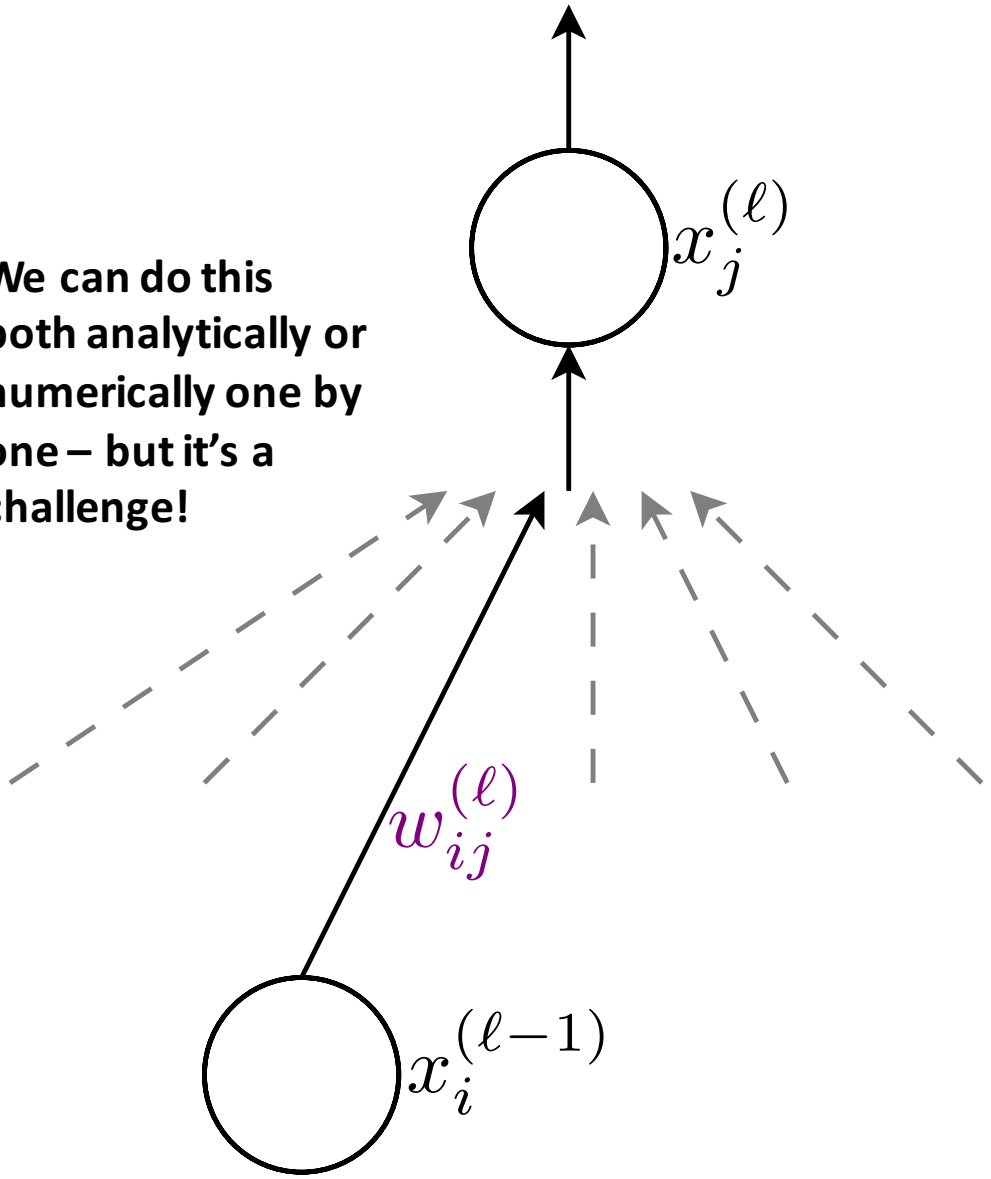
We can do this both analytically or numerically one by one



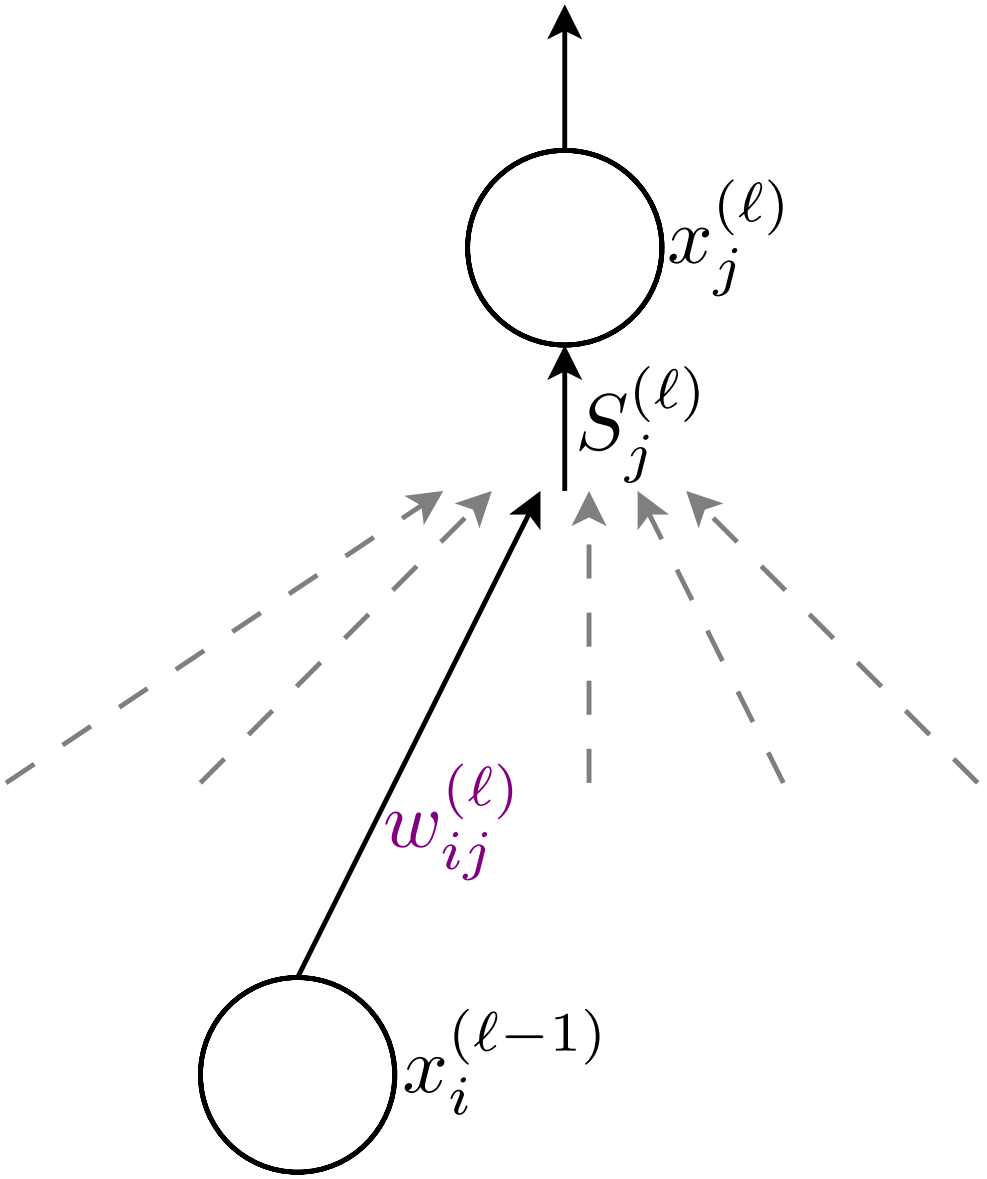
Partial Derivatives

$$\frac{\delta e(w)}{\delta w_{ij}^{(\ell)}}$$

We can do this both analytically or numerically one by one – but it's a challenge!

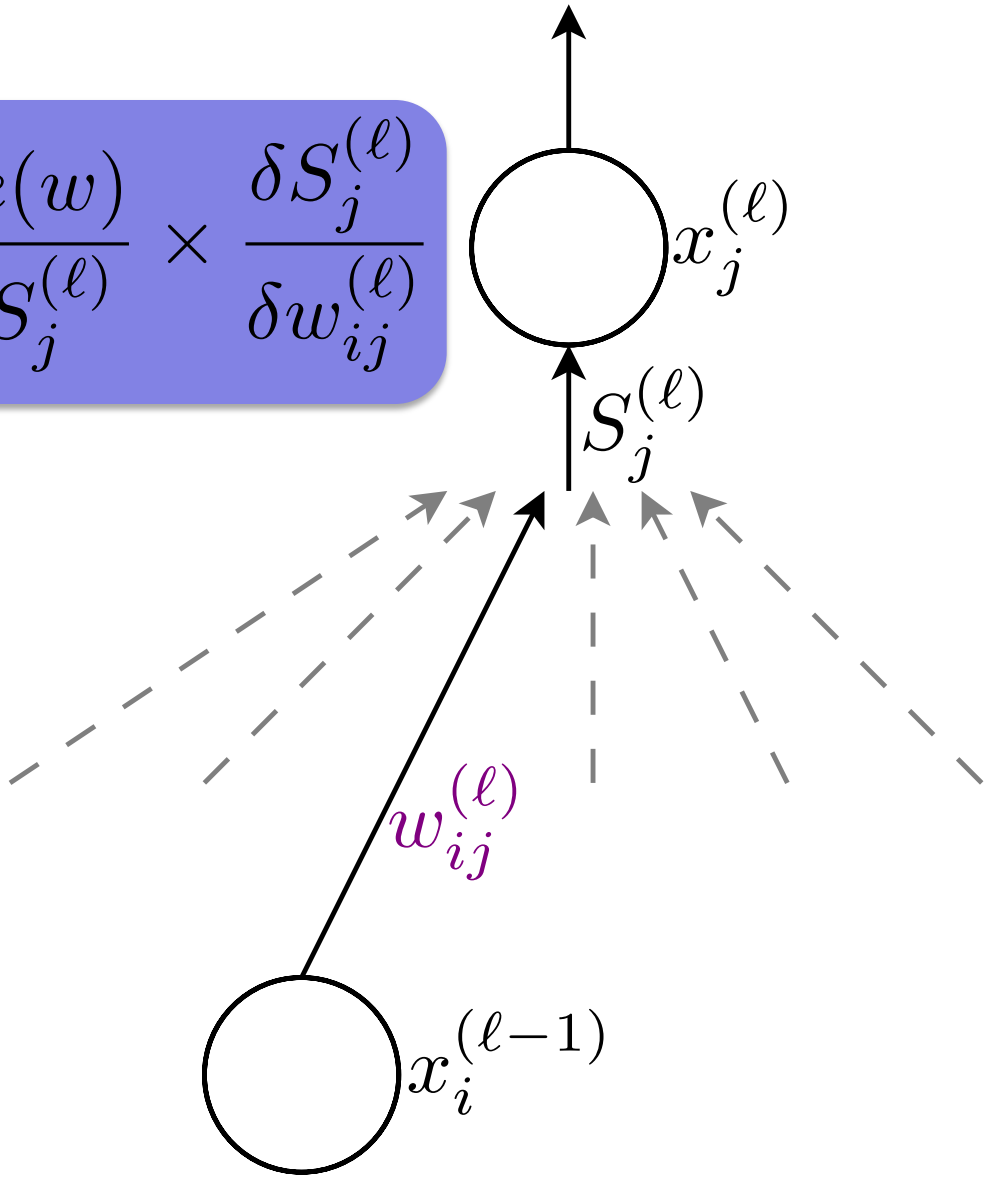


Partial Derivatives



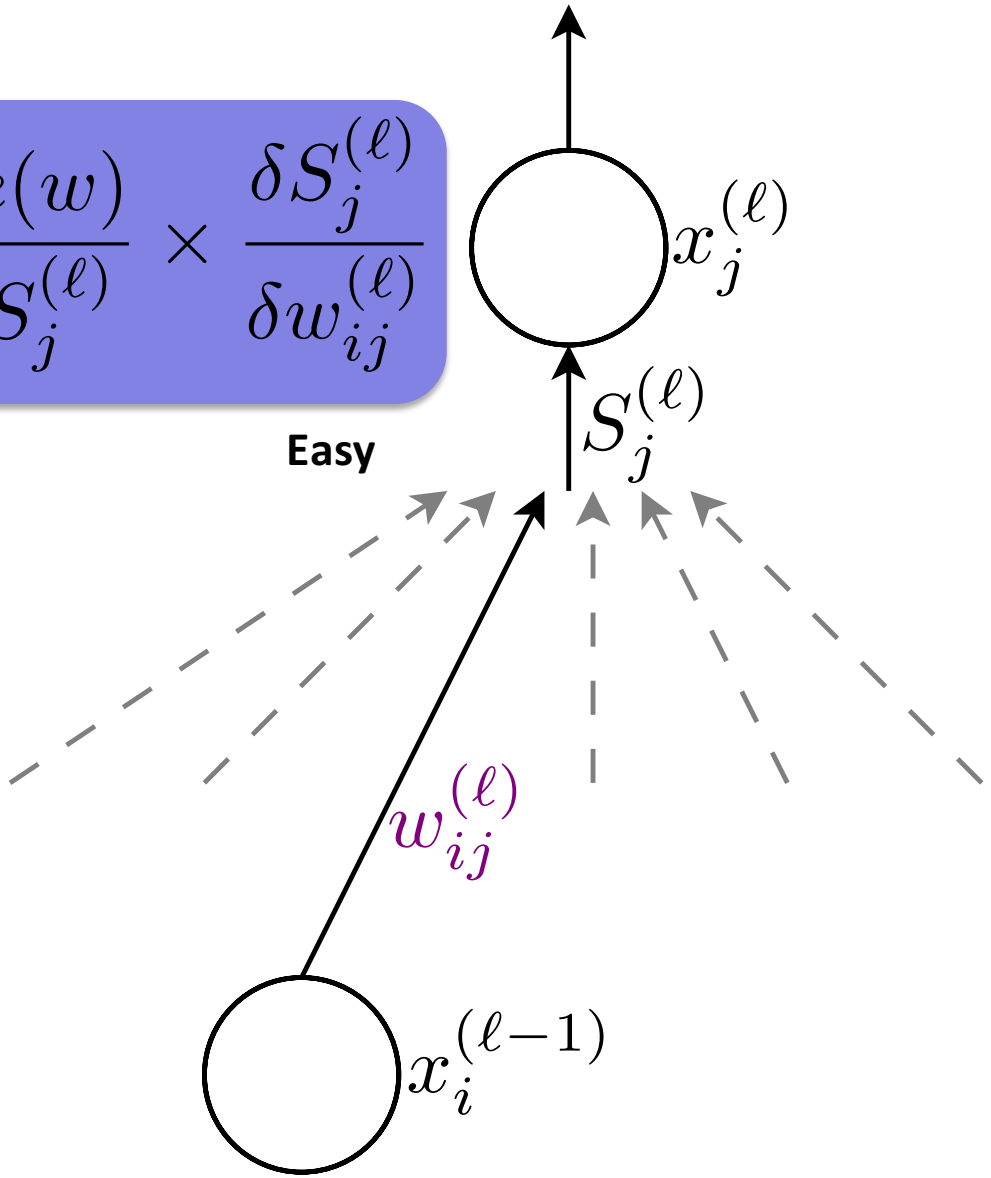
Partial Derivatives : Chain Rule

$$\frac{\delta e(w)}{\delta w_{ij}^{(\ell)}} = \frac{\delta e(w)}{\delta S_j^{(\ell)}} \times \frac{\delta S_j^{(\ell)}}{\delta w_{ij}^{(\ell)}}$$



Partial Derivatives : Chain Rule

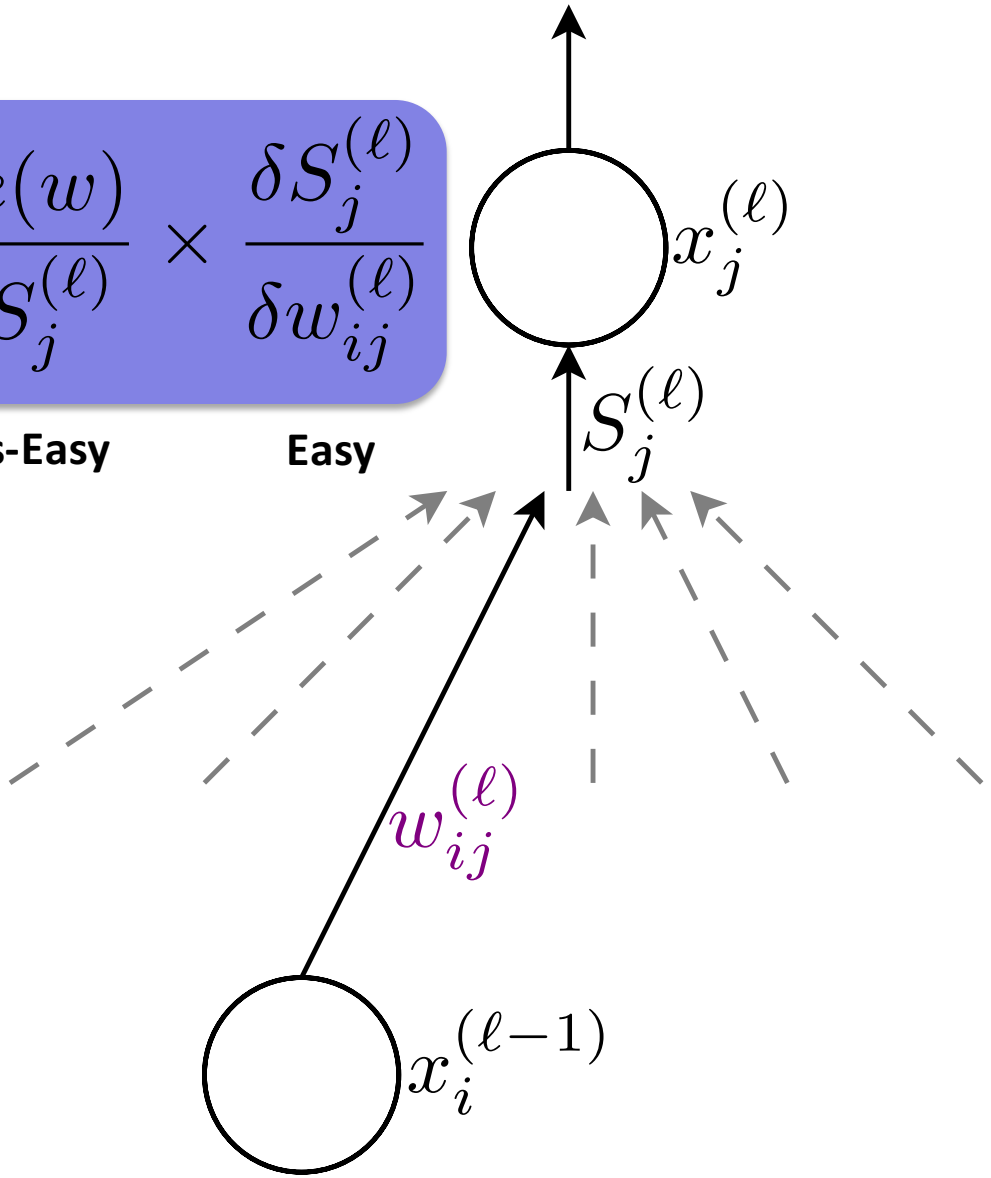
$$\frac{\delta e(w)}{\delta w_{ij}^{(\ell)}} = \frac{\delta e(w)}{\delta S_j^{(\ell)}} \times \frac{\delta S_j^{(\ell)}}{\delta w_{ij}^{(\ell)}}$$



Partial Derivatives : Chain Rule

$$\frac{\delta e(w)}{\delta w_{ij}^{(\ell)}} = \frac{\delta e(w)}{\delta S_j^{(\ell)}} \times \frac{\delta S_j^{(\ell)}}{\delta w_{ij}^{(\ell)}}$$

Less-Easy **Easy**



Up at the Top Layer

$$\frac{\delta e(w)}{\delta s_j^{(\ell)}} = \frac{\delta e(w)}{\delta s_1^{(L)}}$$

$$e(w) = e(h(x_n), y_n)$$

Up at the Top Layer

$$\frac{\delta e(w)}{\delta s_j^{(\ell)}} = \frac{\delta e(w)}{\delta s_1^{(L)}}$$

$$e(w) = e(\ x_1^{(L)}, y_n)$$

Up at the Top Layer

$$\frac{\delta e(w)}{\delta s_j^{(\ell)}} = \frac{\delta e(w)}{\delta s_1^{(L)}}$$

$$e(w) = (x_1^{(L)} - y_n)^2$$

Up at the Top Layer

$$\frac{\delta e(w)}{\delta s_j^{(\ell)}} = \frac{\delta e(w)}{\delta s_1^{(L)}}$$

$$e(w) = (x_1^{(L)} - y_n)^2$$

$$x_1^{(L)} = \theta(s_1^{(L)})$$

Up at the Top Layer

$$\frac{\delta e(w)}{\delta s_j^{(\ell)}} = \frac{\delta e(w)}{\delta s_1^{(L)}}$$

$$e(w) = (x_1^{(L)} - y_n)^2$$

$$x_1^{(L)} = \theta(s_1^{(L)})$$

$$\theta'(s)$$

Up at the Top Layer

$$\frac{\delta e(w)}{\delta s_j^{(\ell)}} = \frac{\delta e(w)}{\delta s_1^{(L)}}$$

$$e(w) = (x_1^{(L)} - y_n)^2$$

$$x_1^{(L)} = \theta(s_1^{(L)})$$

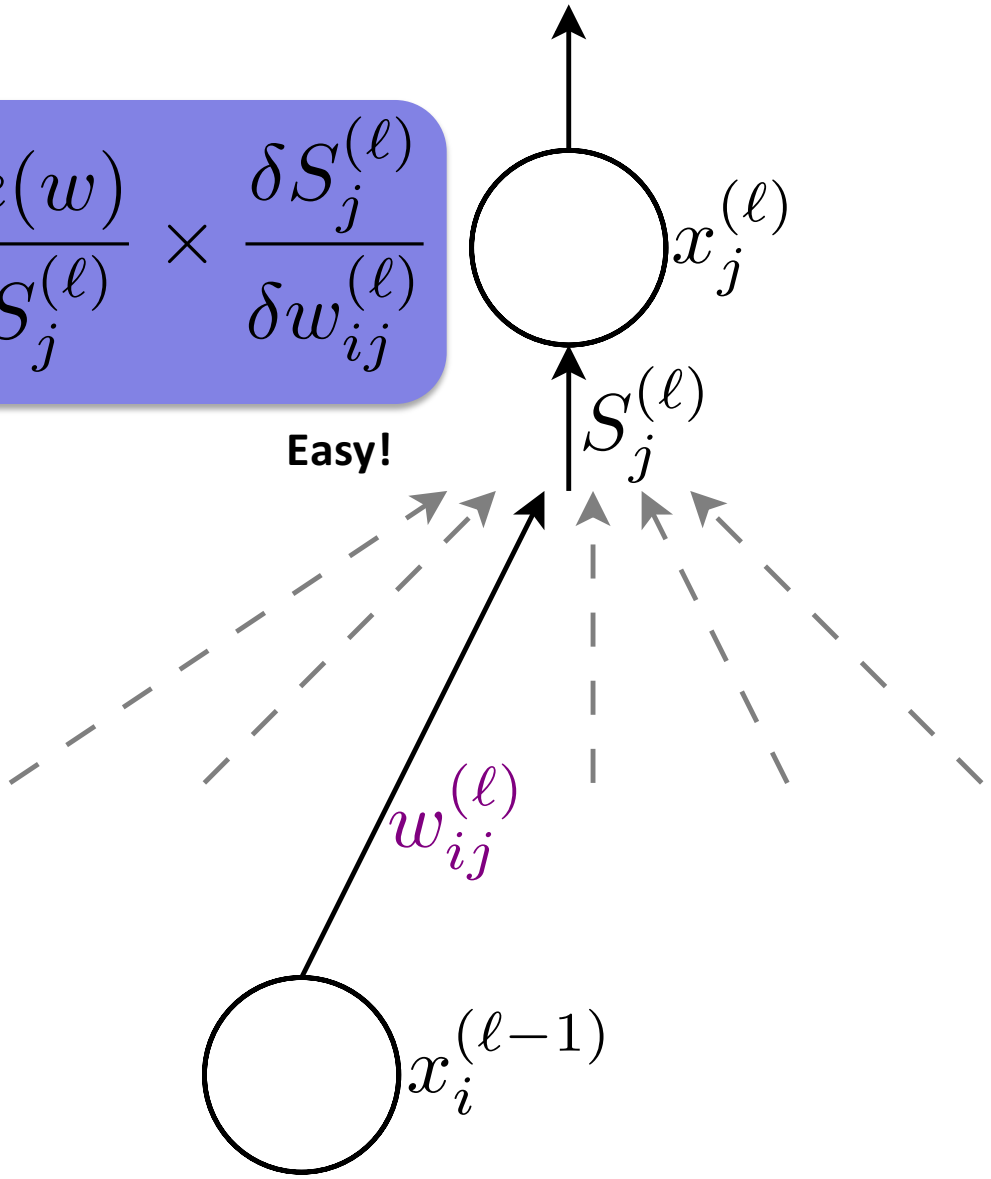
$$\theta'(s)$$

Which we made sure was differentiable earlier

Up at the Top Layer

$$\frac{\delta e(w)}{\delta w_{ij}^{(\ell)}} = \frac{\delta e(w)}{\delta S_j^{(\ell)}} \times \frac{\delta S_j^{(\ell)}}{\delta w_{ij}^{(\ell)}}$$

Easy!

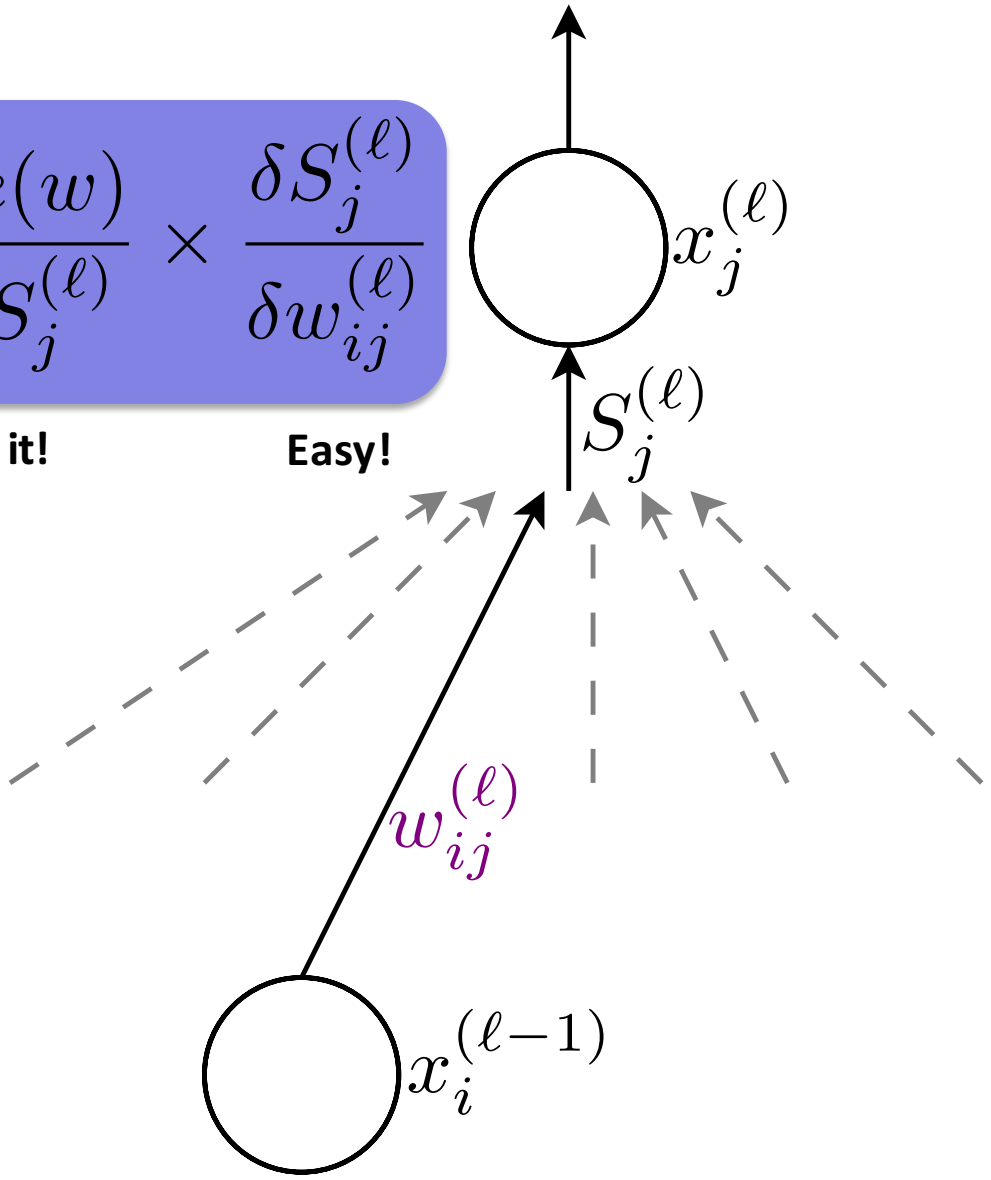


Up at the Top Layer

$$\frac{\delta e(w)}{\delta w_{ij}^{(\ell)}} = \frac{\delta e(w)}{\delta S_j^{(\ell)}} \times \frac{\delta S_j^{(\ell)}}{\delta w_{ij}^{(\ell)}}$$

Got it!

Easy!

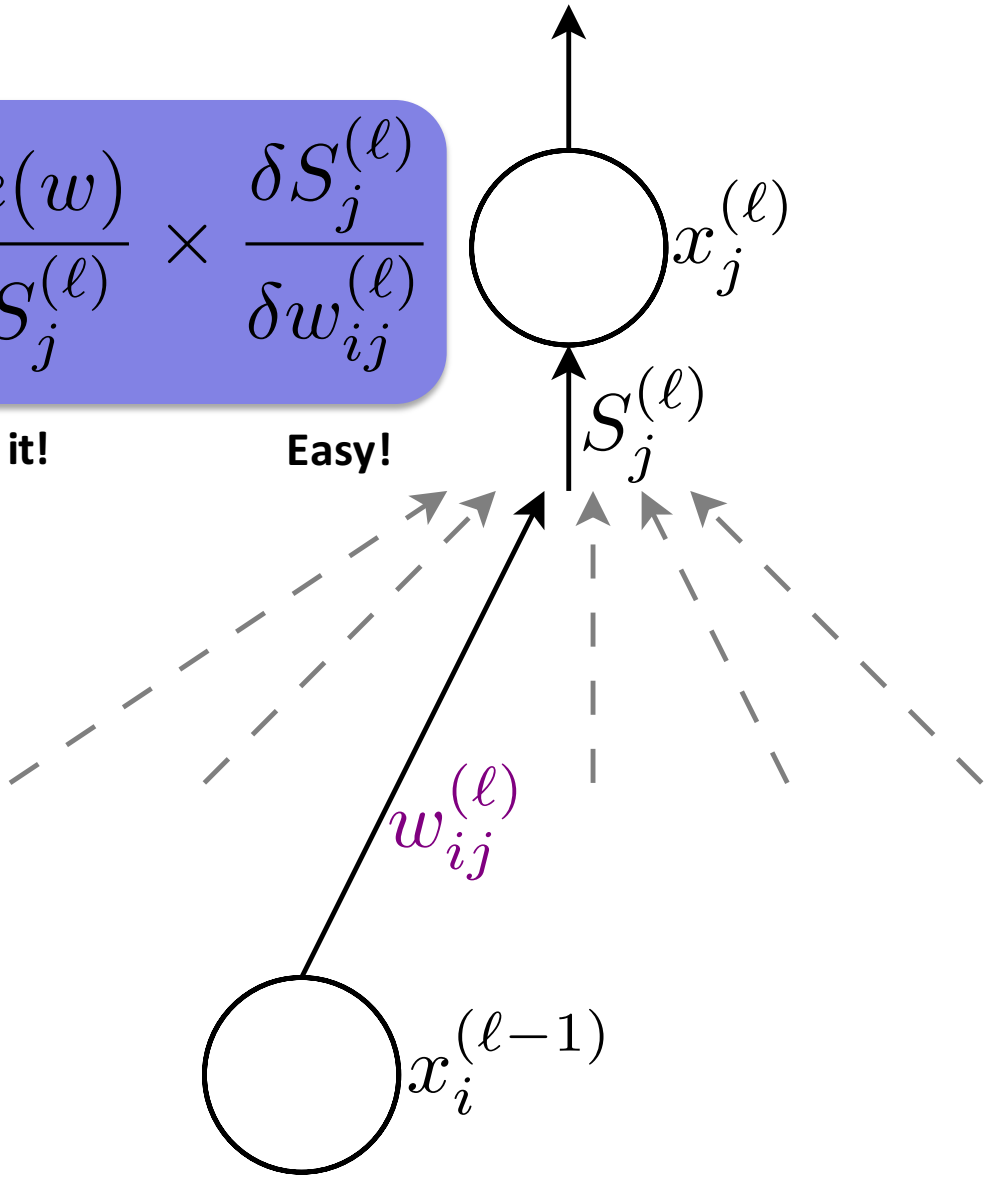


…and propagate backwards

$$\frac{\delta e(w)}{\delta w_{ij}^{(\ell)}} = \frac{\delta e(w)}{\delta S_j^{(\ell)}} \times \frac{\delta S_j^{(\ell)}}{\delta w_{ij}^{(\ell)}}$$

Got it!

Easy!

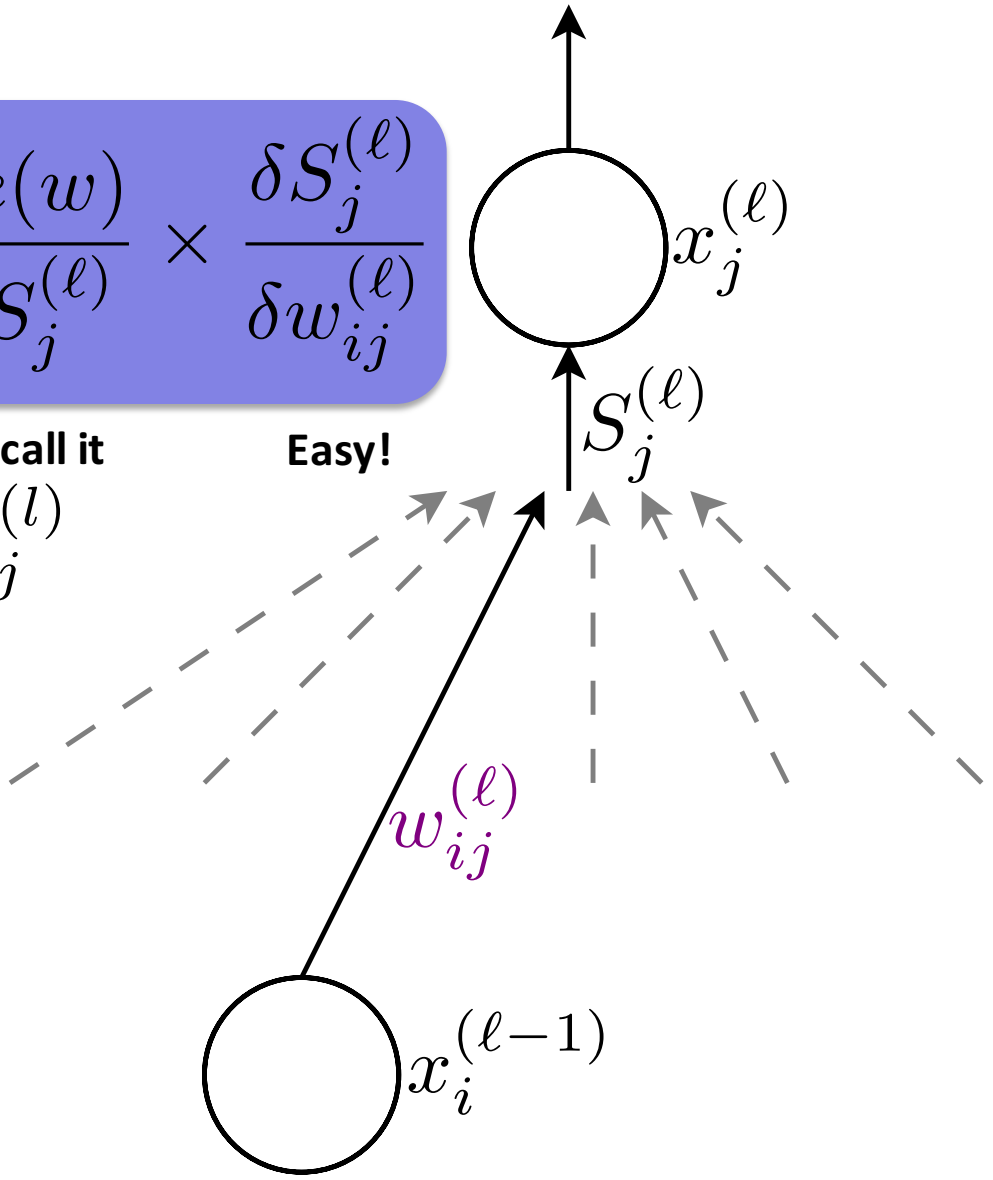


…and propagate backwards

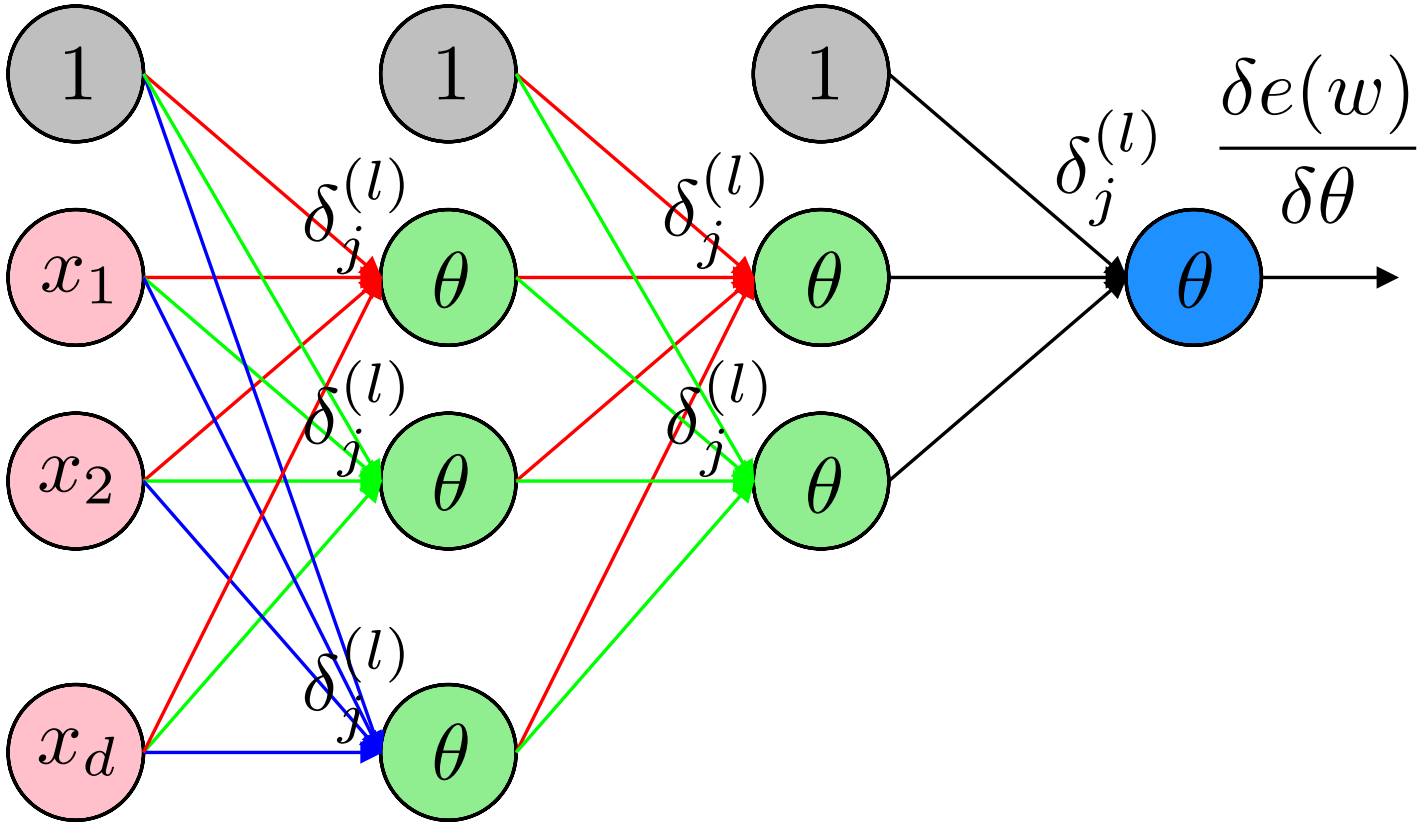
$$\frac{\delta e(w)}{\delta w_{ij}^{(\ell)}} = \frac{\delta e(w)}{\delta S_j^{(\ell)}} \times \frac{\delta S_j^{(\ell)}}{\delta w_{ij}^{(\ell)}}$$

Let's call it $\delta_j^{(l)}$

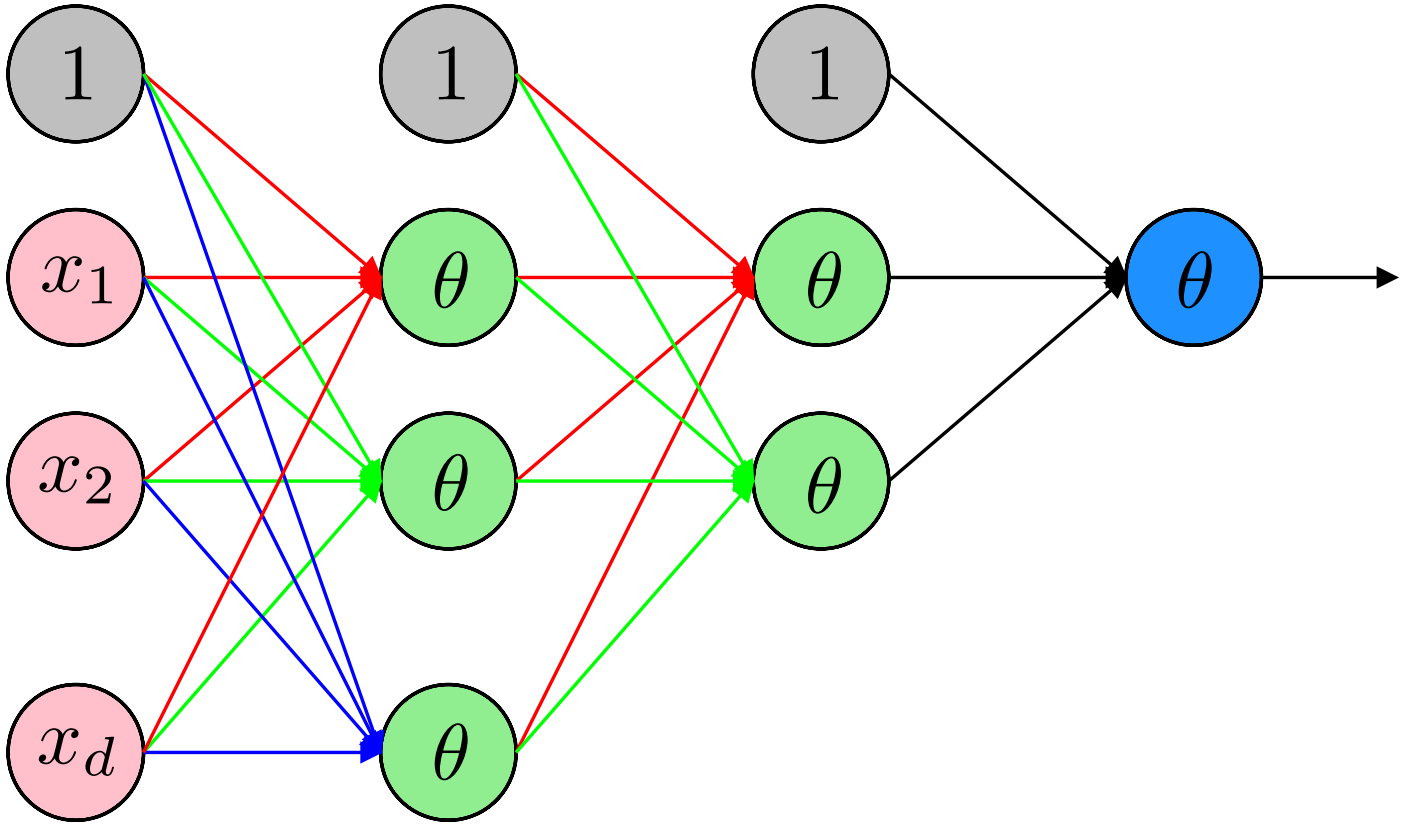
Easy!



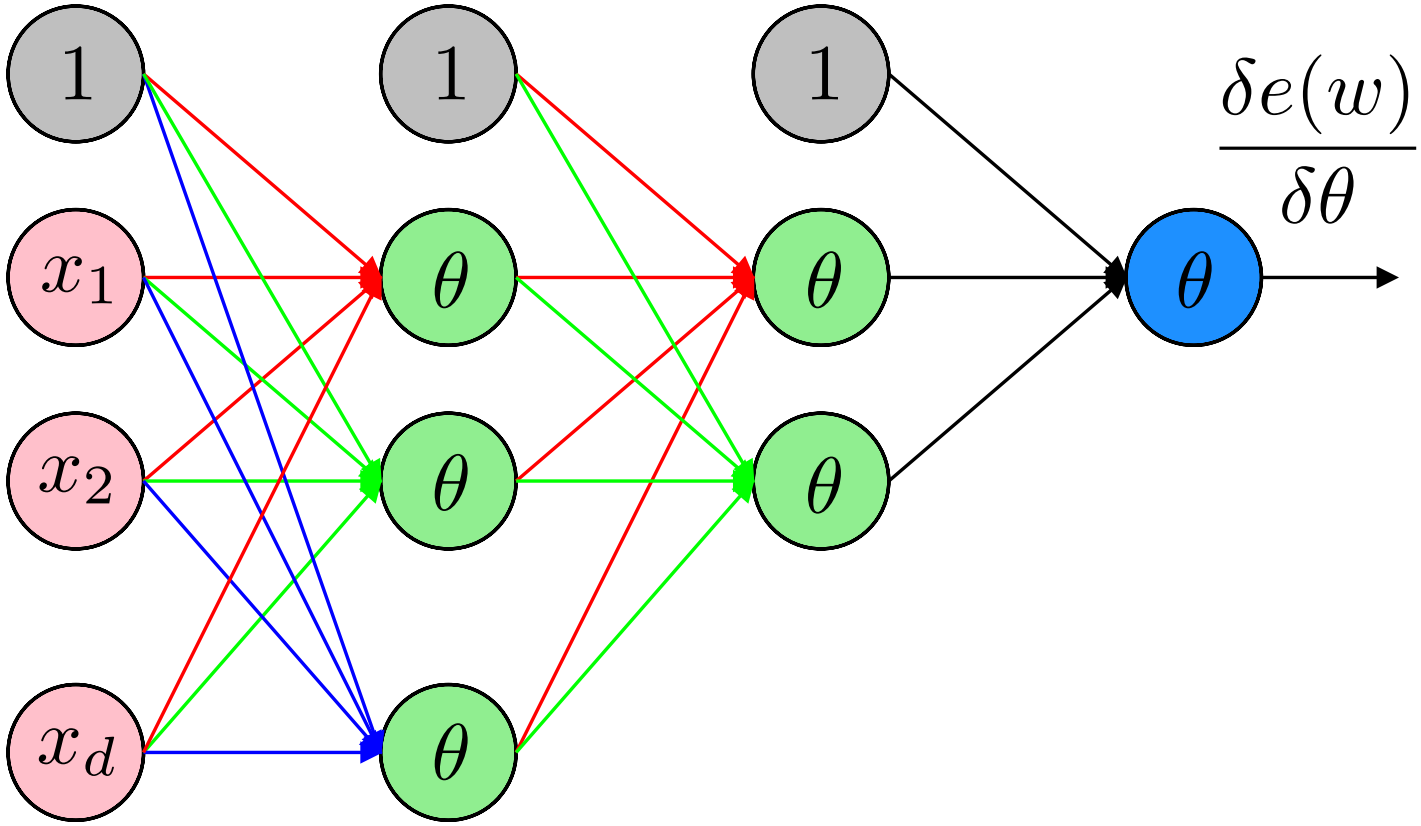
Computing the "Delta's" Recursively



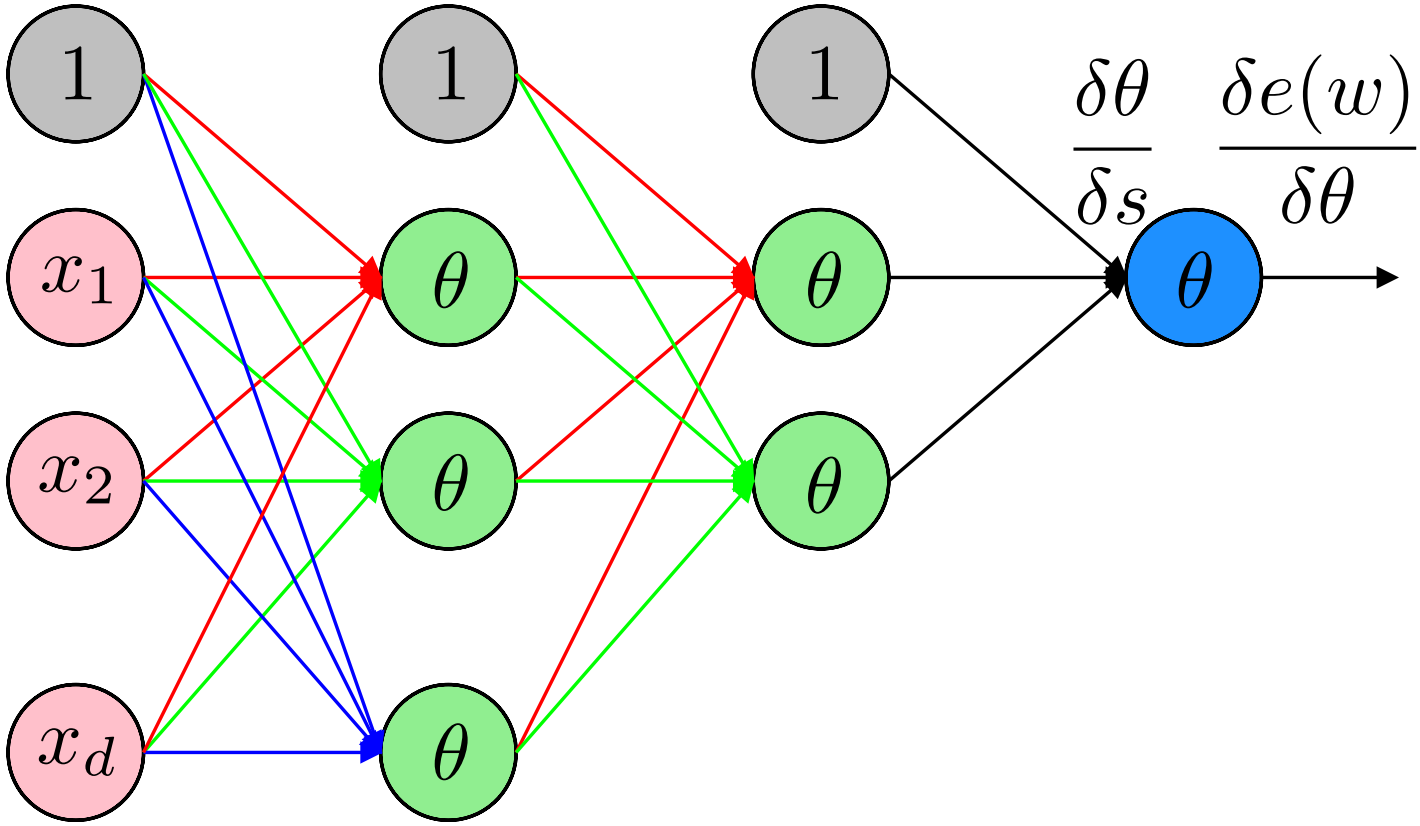
Backpropagation



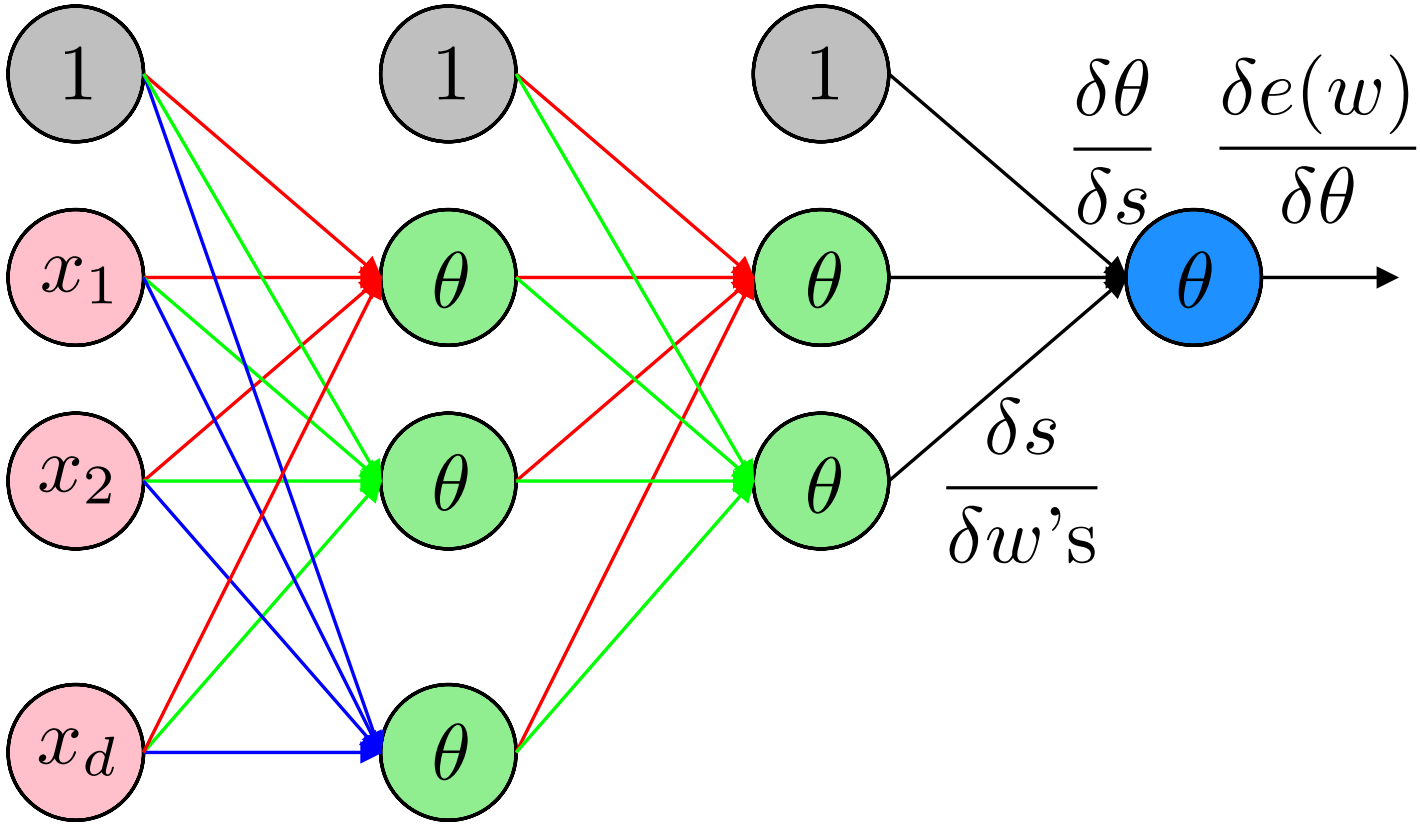
Backpropagation



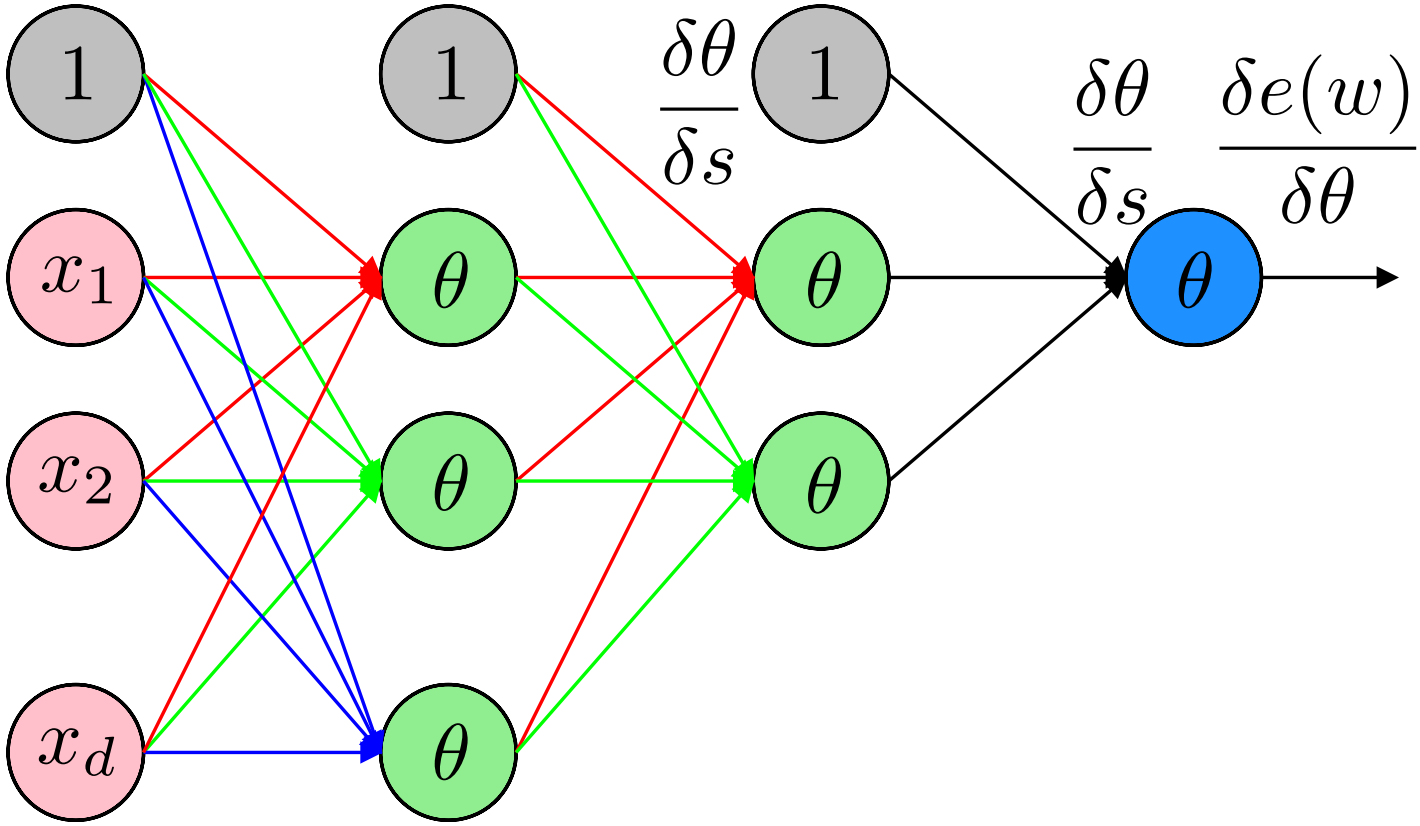
Backpropagation



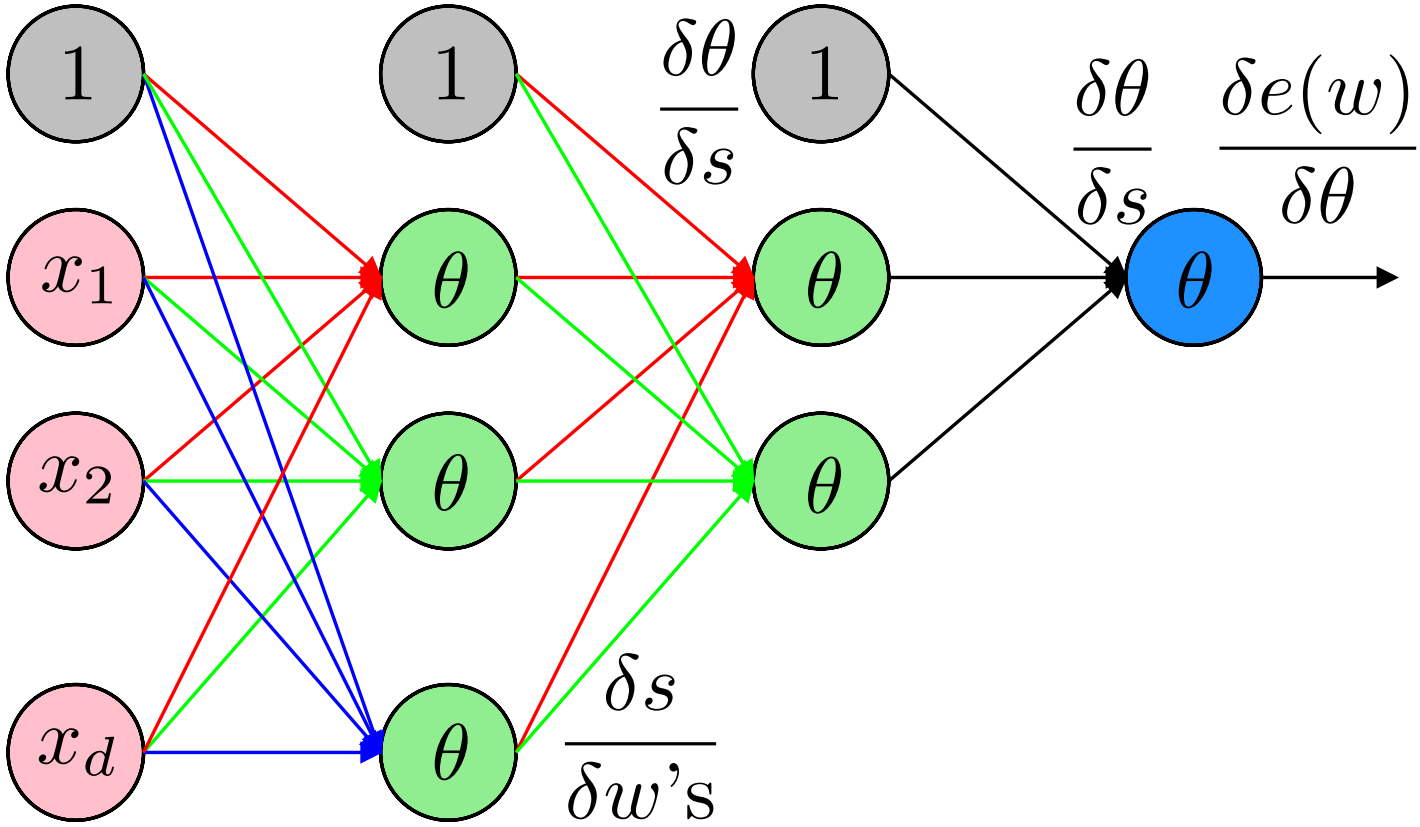
Backpropagation



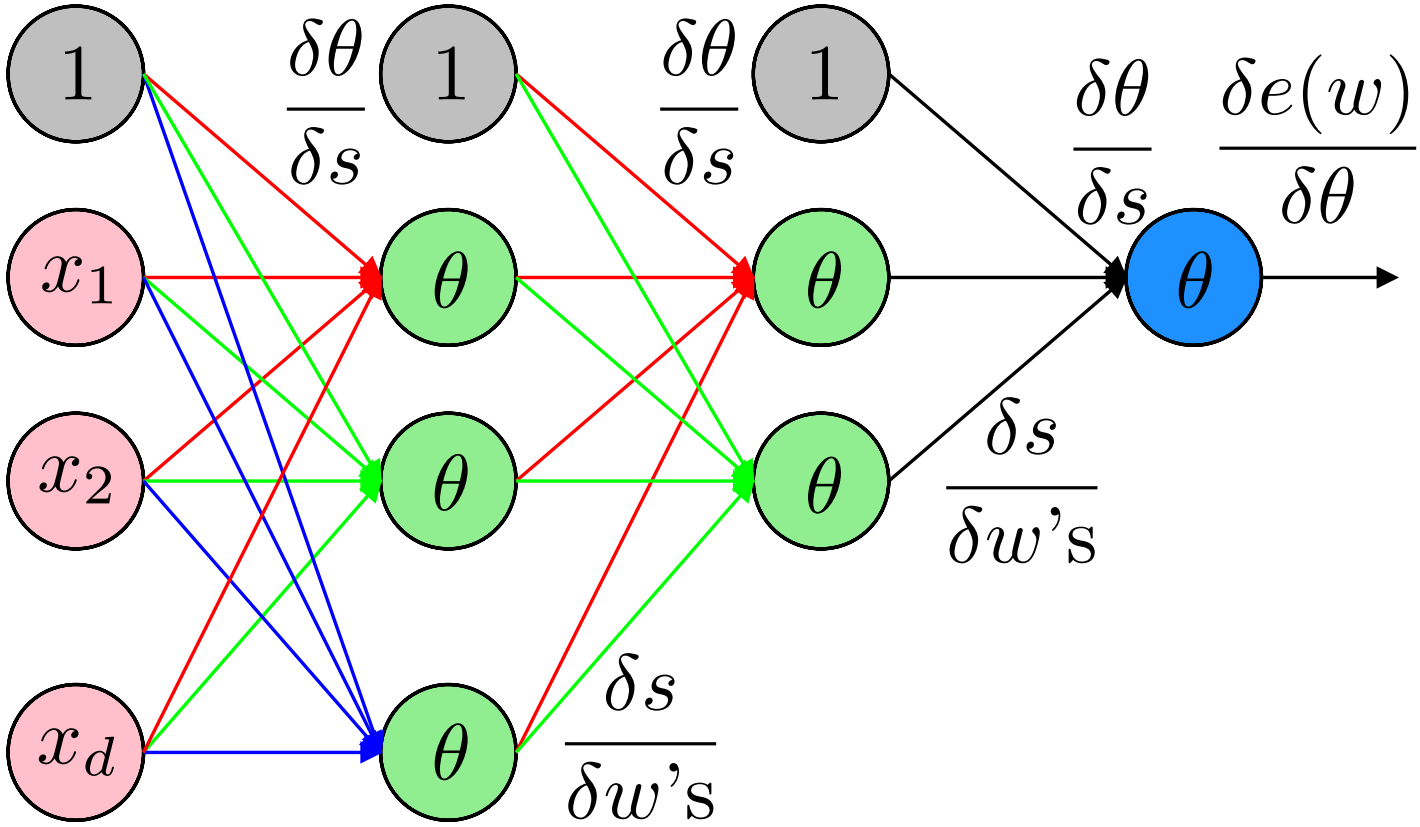
Backpropagation



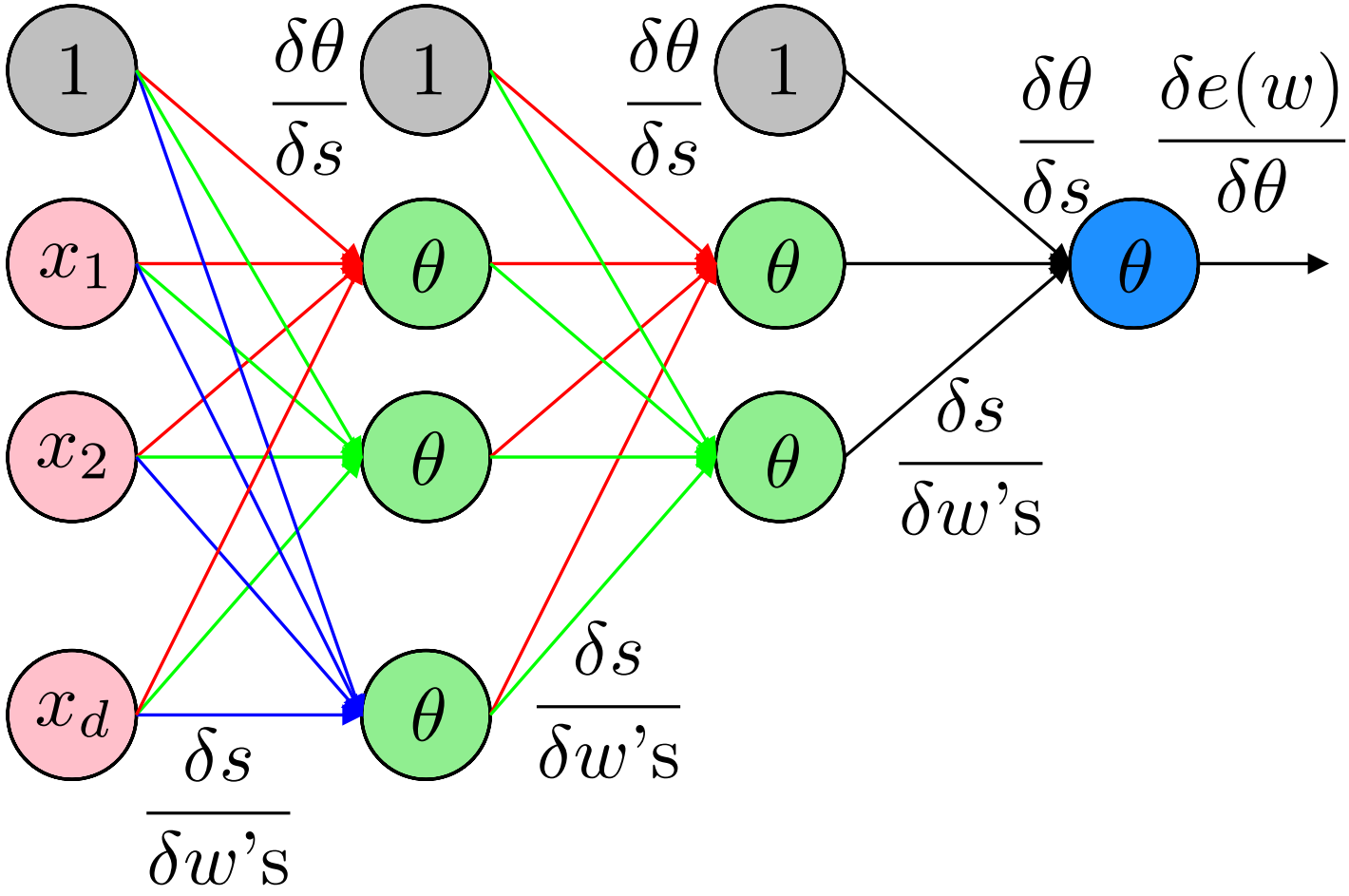
Backpropagation



Backpropagation

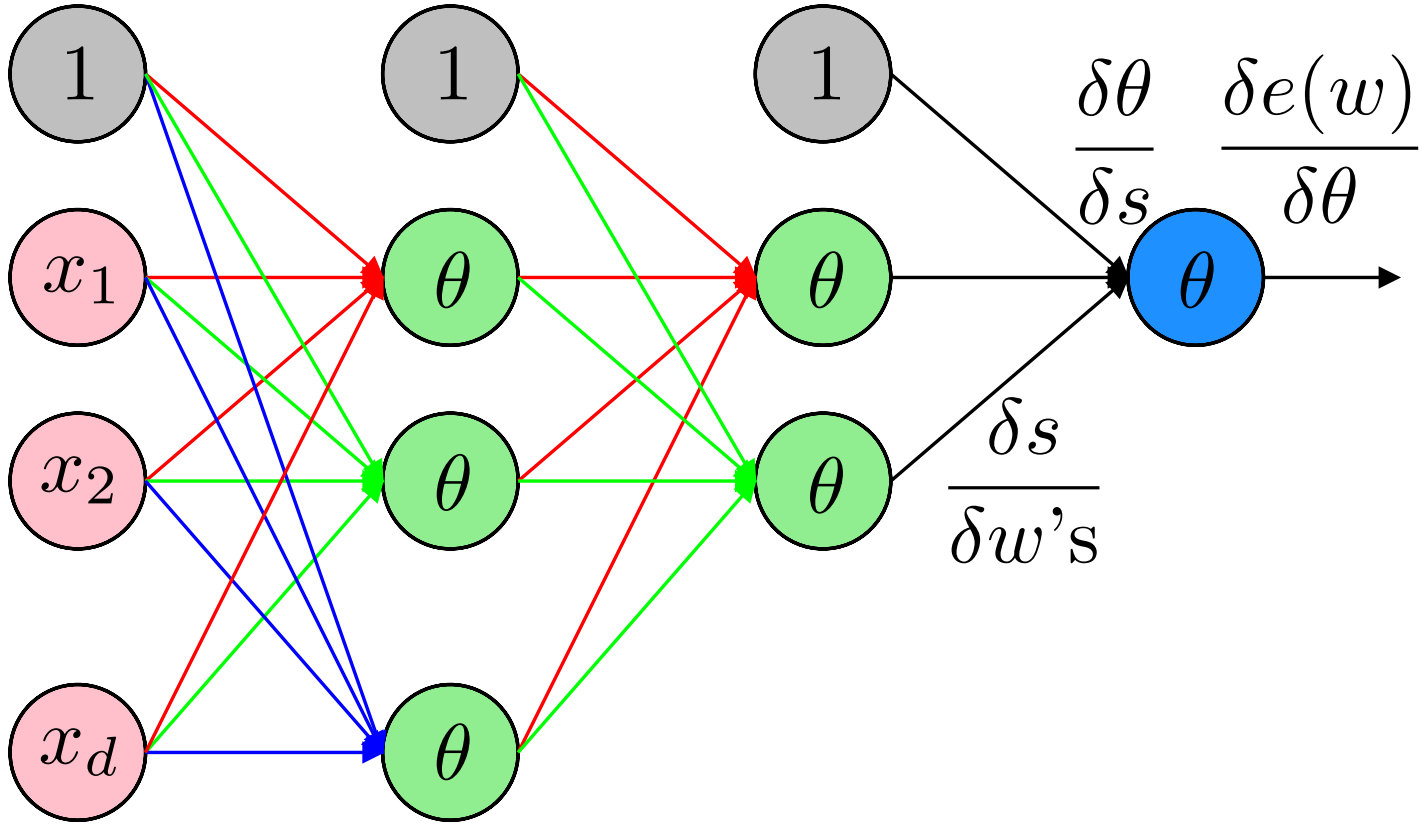


Backpropagation



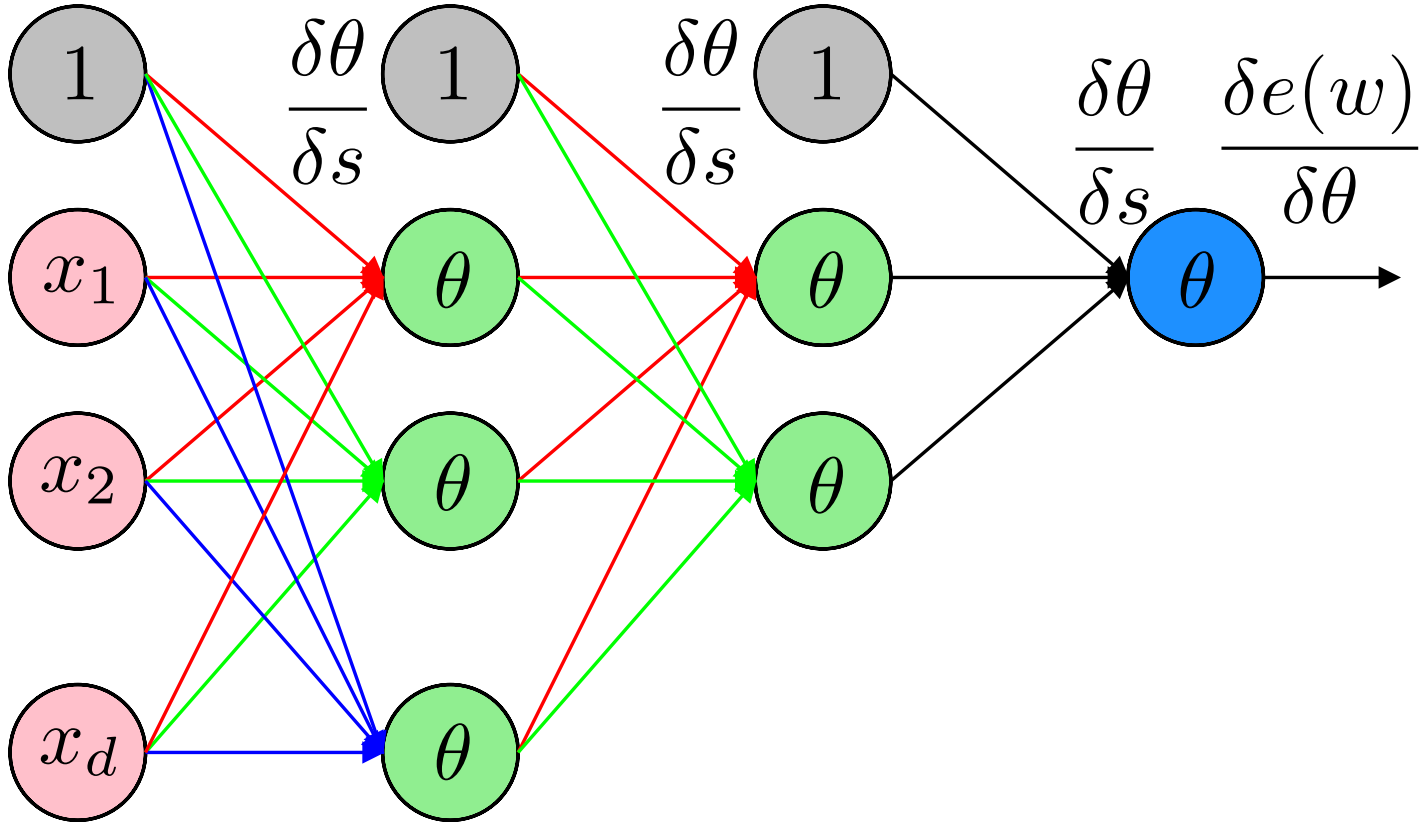
Chain Rule Magic

$$\frac{\delta e(w)}{\delta w_{ij}^{(3)}} = \frac{\delta e(w)}{\delta \theta} \frac{\delta \theta}{\delta s} \frac{\delta s}{\delta w\text{'s}}$$



Chain Rule Magic

$$\frac{\delta e(w)}{\delta w_{ij}^{(1)}} = \frac{\delta e(w)}{\delta \theta}\frac{\delta \theta}{\delta s}\frac{\delta s}{\delta \theta}\frac{\delta \theta}{\delta s}\frac{\delta s}{\delta \theta}\frac{\delta \theta}{\delta s}\frac{\delta s}{\delta w}$$

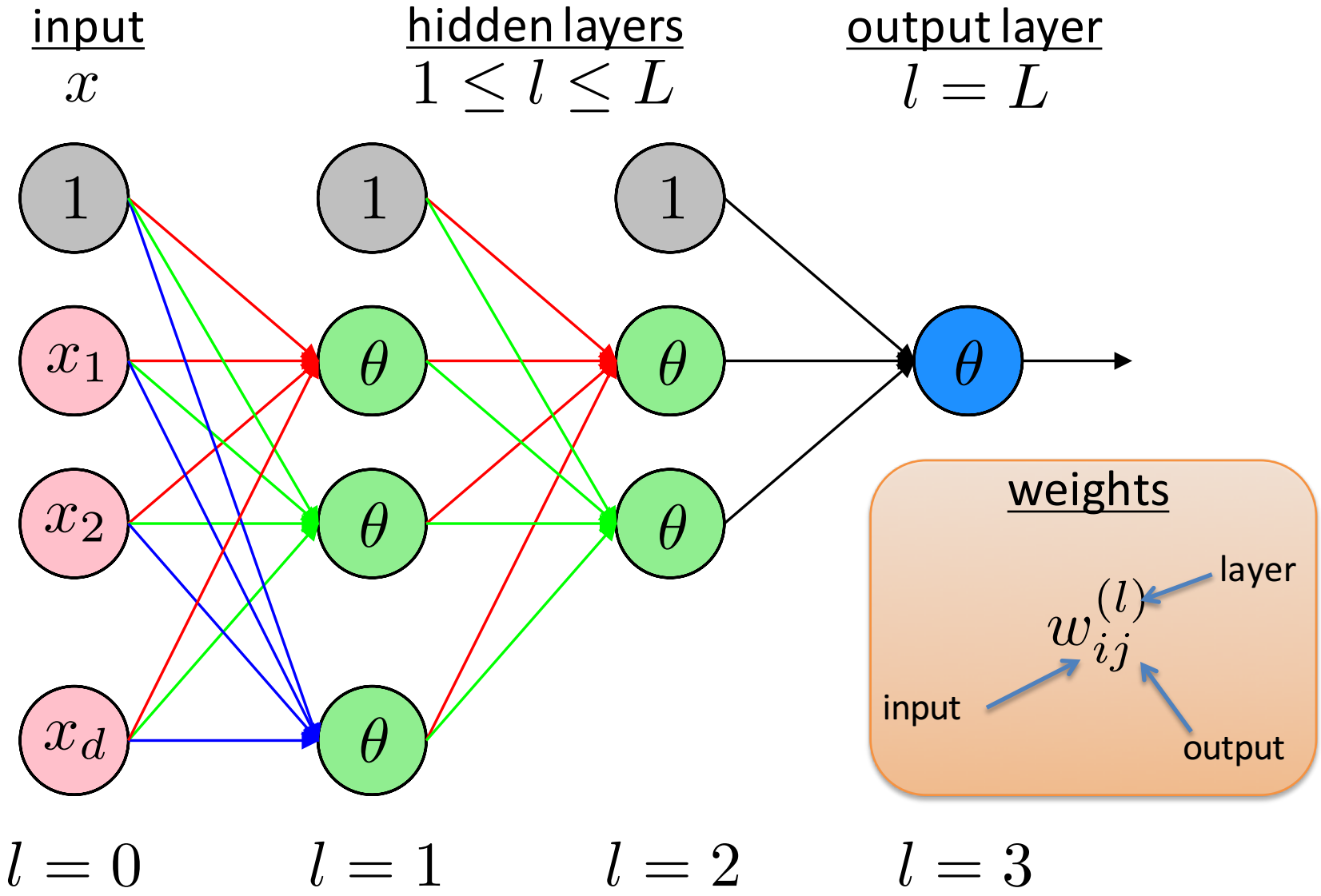


Key Idea 3 : Backpropagation

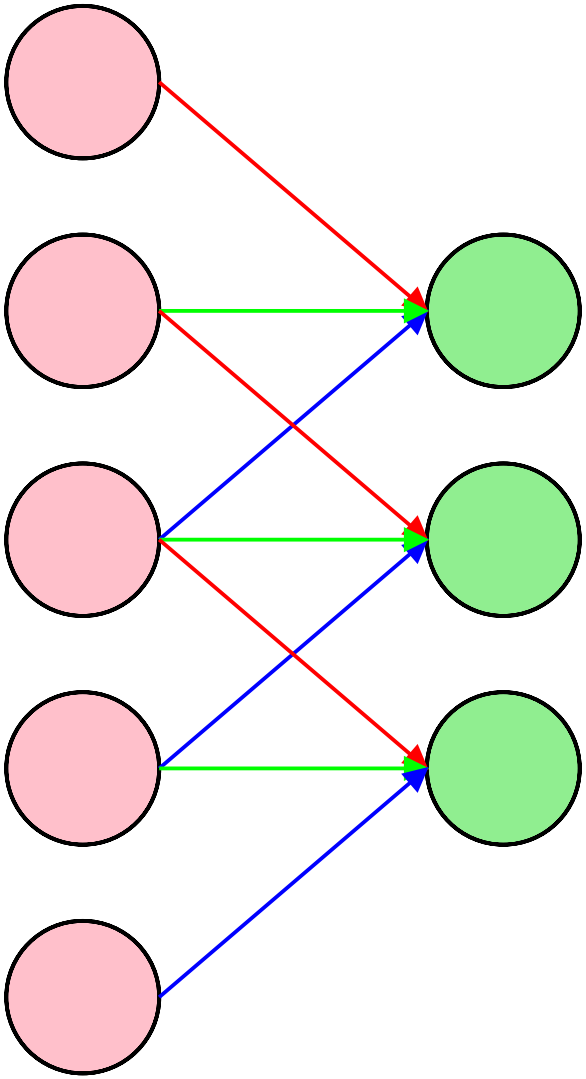
initialize all weights $w_{ij}^{(l)}$
for $t = 1 \ldots T$
    pick a training example n
    FORWARD: Compute all $x_j^{(l)}$
    BACKWARD: Compute all $\delta_j^{(l)}$
    update the weights $w_{ij}^{(l)} \leftarrow w_{ij}^{(l)} \eta x_i^{(l-1)} \delta_j^{(l)}$
endfor return weights

A Few More Ideas

Hidden Layers and Features



Convolutional Neural Networks



- Shared weights used across network
- May mimic a behavior of particular importance to the visual cortex of the brain

The Rise and Fall and Rise of ANN's

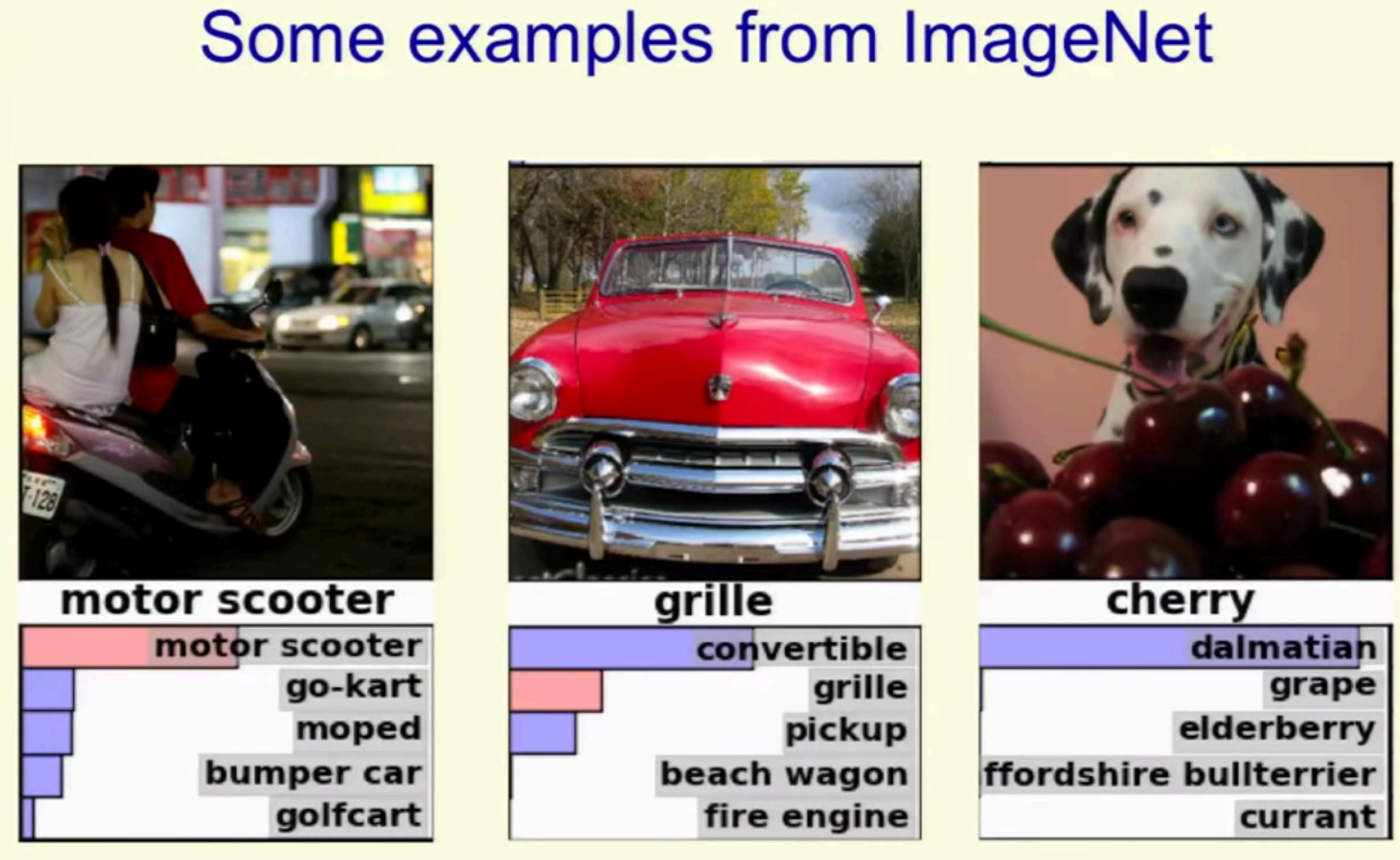
Deep Neural Networks

- Had been considered too difficult to train
- We had almost ‘proved’ that they couldn’t be trained!
- Until someone showed us how
- Recent successes are owed mainly to greater amounts of training data and far greater computational resources (esp. GPU’s)

Impressive Results

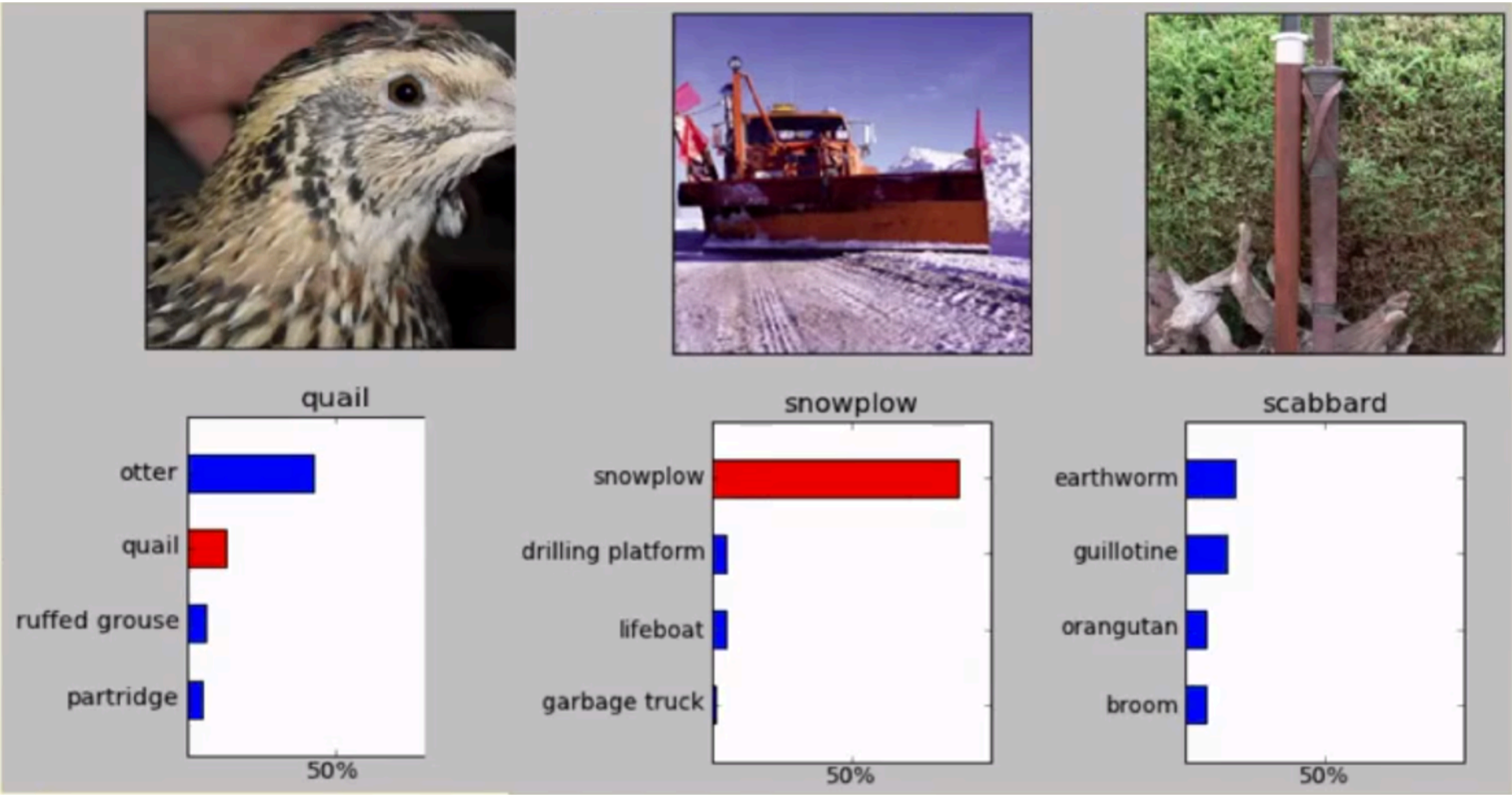
- Classifying handwritten digits (MNIST)
  - Better than humans
- Chinese character recognition
- Traffic Sign Recognition 99% accurate
  - Better than humans
- Winning entries for ImageNet Challenge

Impressive Results



Example from : Hinton Google Talk https://www.youtube.com/watch?v=vShMxxqtDDs

Impressive Results

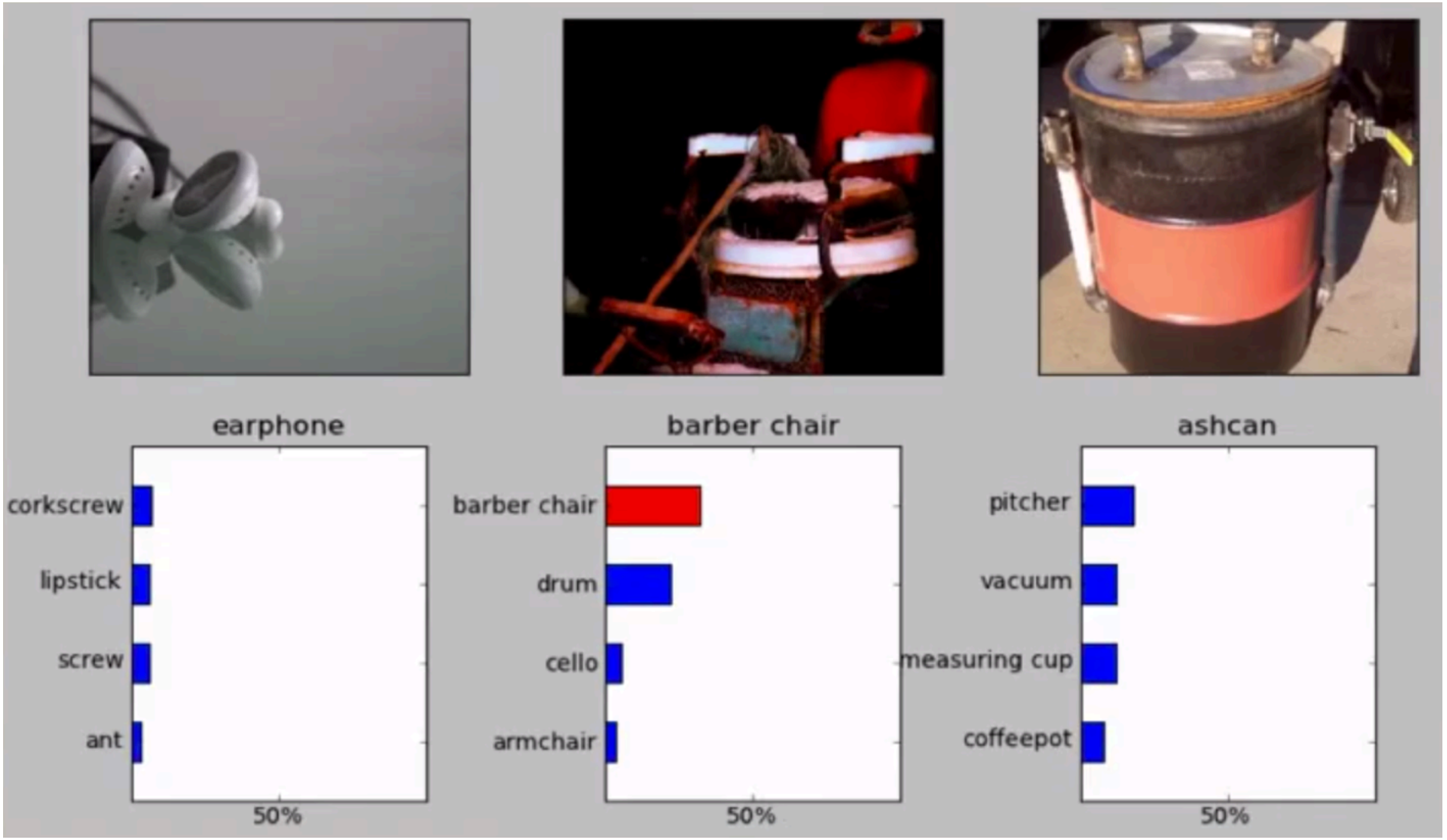


Example from : Coursera Neural Networks Class (Hinton)

Impressive Results

Example from : Coursera Neural Networks Class (Hinton)

Impressive Results



Example from : Coursera Neural Networks Class (Hinton)

Resources

- Coursera Neural Networks Course
  - https://class.coursera.org/neuralnets-2012-001
- Geoff Hinton Google Tech Talk
  - https://www.youtube.com/watch?v=vShMxxqtDDs
- Tutorials and gentle introduction + code
  - http://deeplearning.net
- More flexible deep learning framework
  - http://caffe.berkeleyvision.org/
- Maybe even MORE flexible deep learning framework
  - http://torch.ch/docs/getting-started.htm