CS 412 Introduction to Machine Learning

Neural Networks

Instructor: Wei Tang

Department of Computer Science
University of Illinois at Chicago
Chicago IL 60607

https://tangw.people.uic.edu
tangw@uic.edu

Slide credits: Noah Snavely, Ying Wu, James Tompkin, Ioannis Gkioulekas, Steve Seitz, Rick Szeliski

Announcements

- The final exam will be on Friday, 12/10 from 1-3:00 in TBH (Thomas Beckham Hall) 180F.
- In-person, no other option

1950s Age of the Perceptron

1957 The Perceptron (Rosenblatt)

1969 Perceptrons (Minsky, Papert)

1980s Age of the Neural Network

1986 Back propagation (Hinton)

1990s Age of the Graphical Model

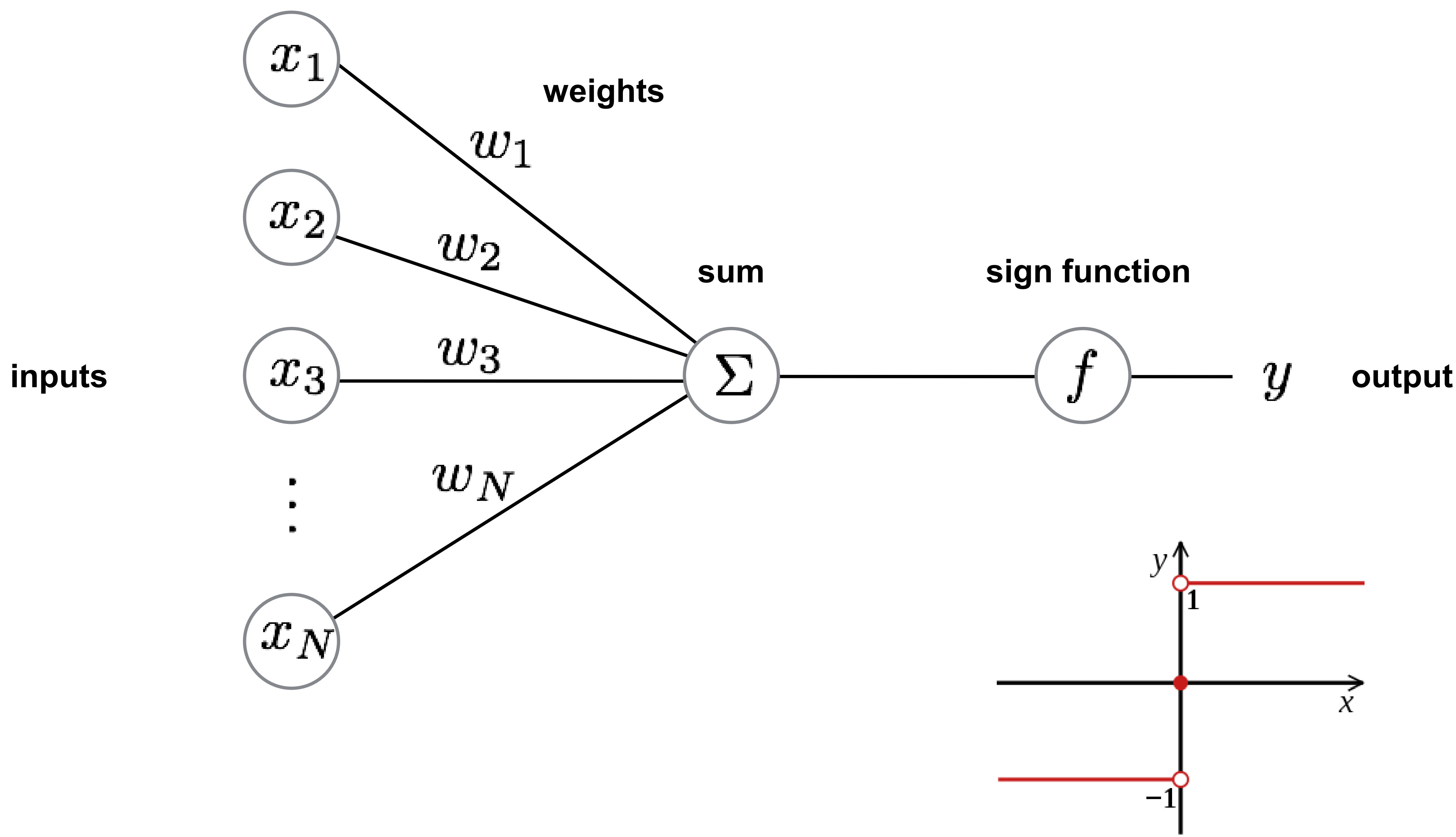
2000s Age of the Support Vector Machine

2010s Age of the Deep Network

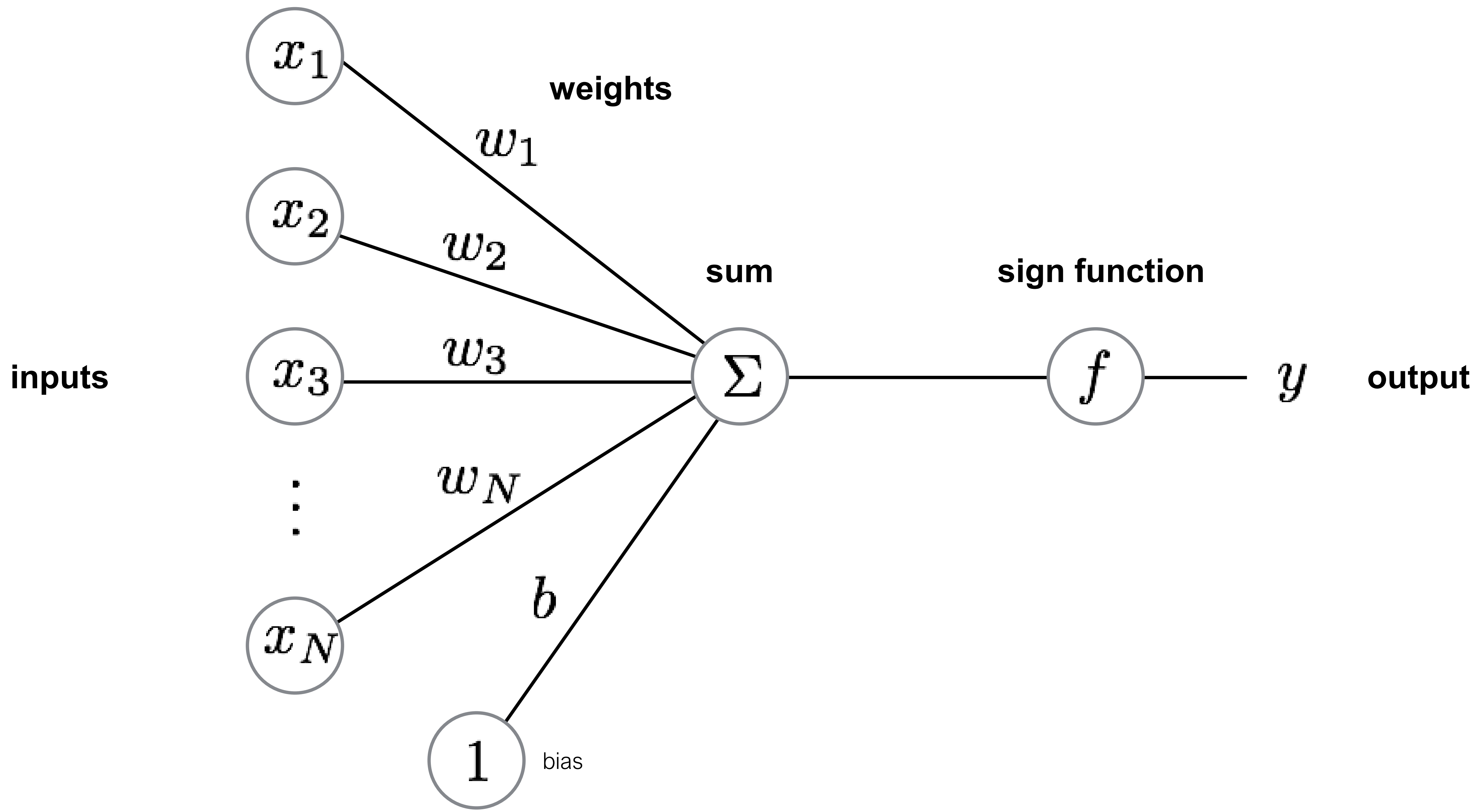
deep learning = known algorithms + computing power + big data

Perceptron

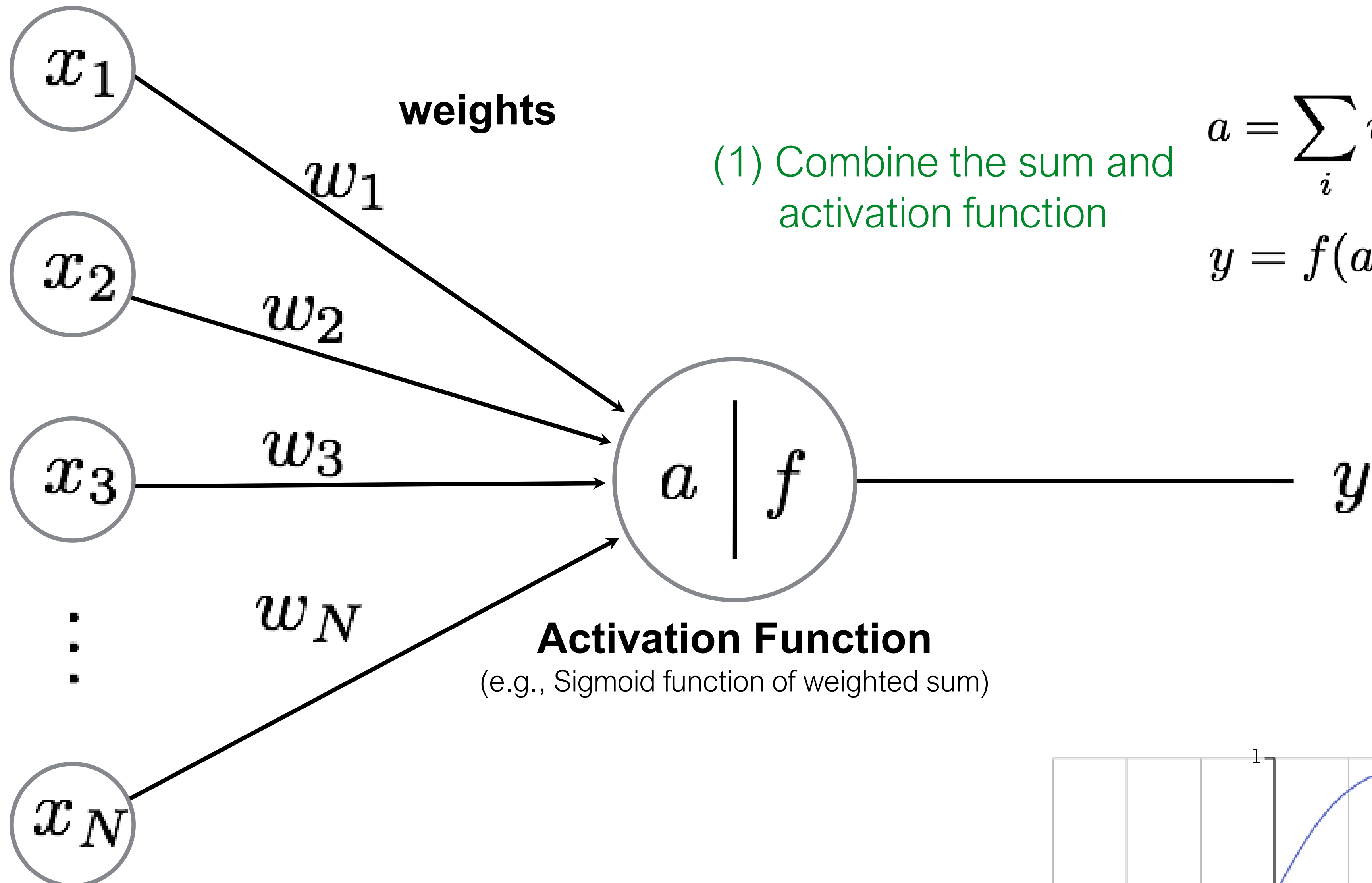
The Perceptron



The Perceptron



Another way to draw it…



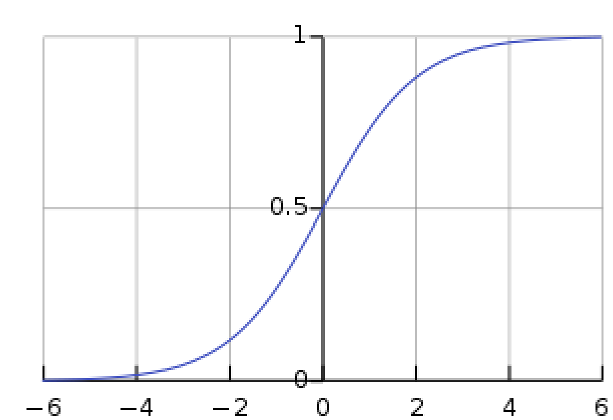
(1) Combine the sum and activation function

$$a = \sum_i w_i x_i$$

$$y = f(a)$$

(2) suppress the bias term (less clutter)

$$x_N = 1$$
$$w_N = b$$

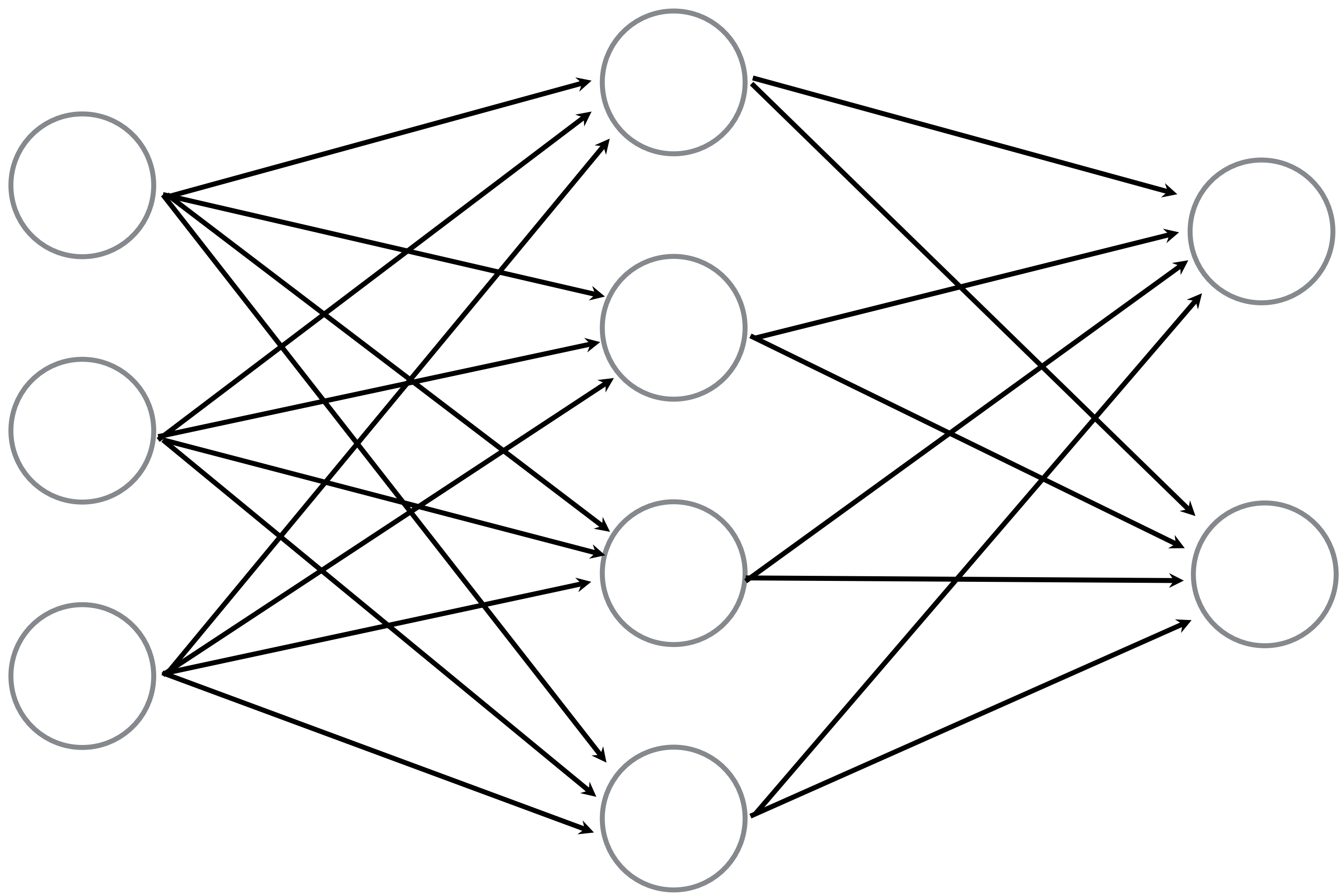


Neural networks

Connect a bunch of perceptrons together …

Neural Network

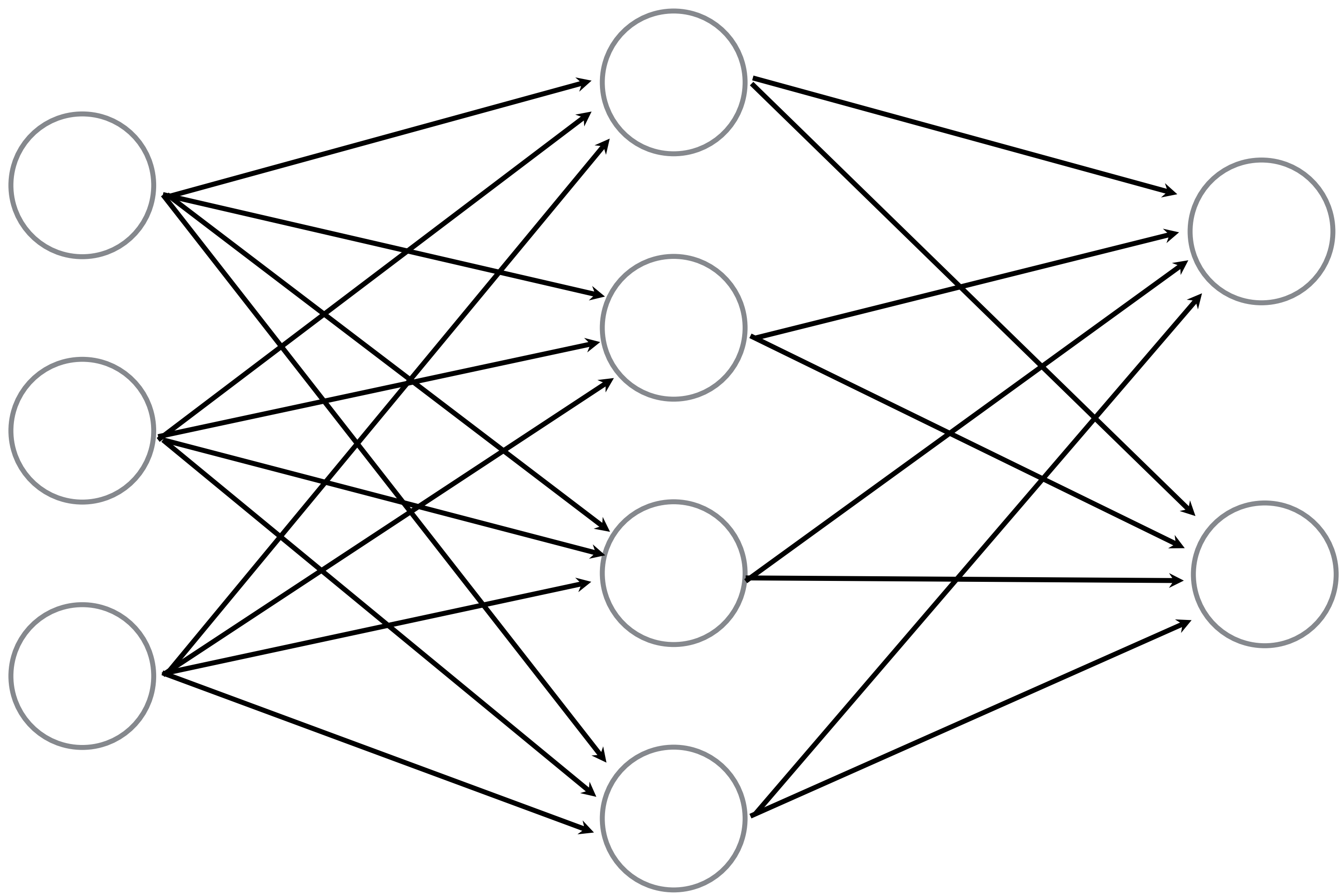
a collection of connected perceptrons



Connect a bunch of perceptrons together …

Neural Network

a collection of connected perceptrons

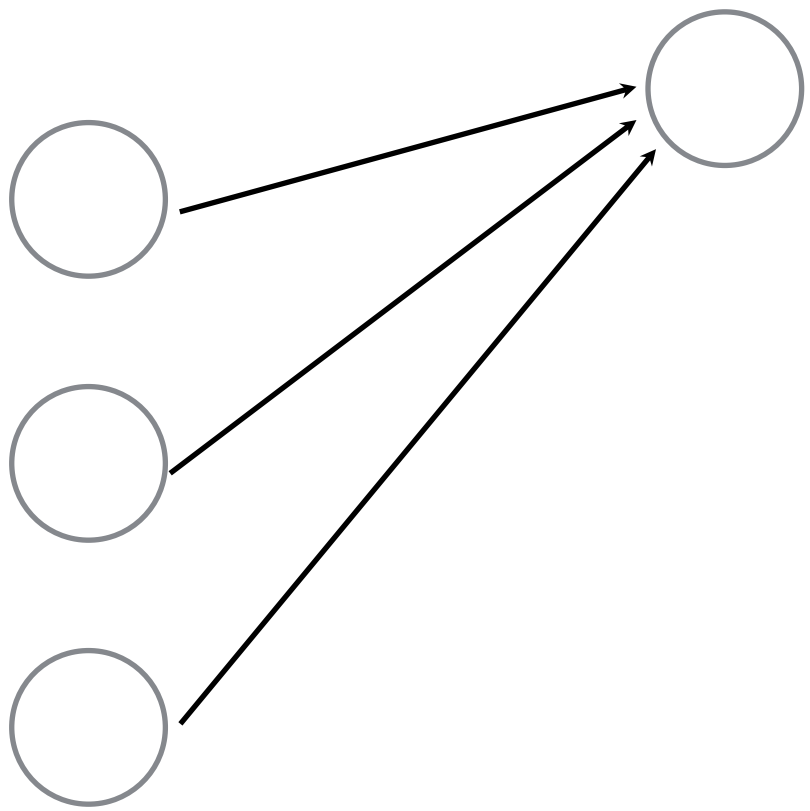


How many perceptrons in this neural network?
(the bias is ignored for cleaness)

Connect a bunch of perceptrons together …

Neural Network

a collection of connected perceptrons

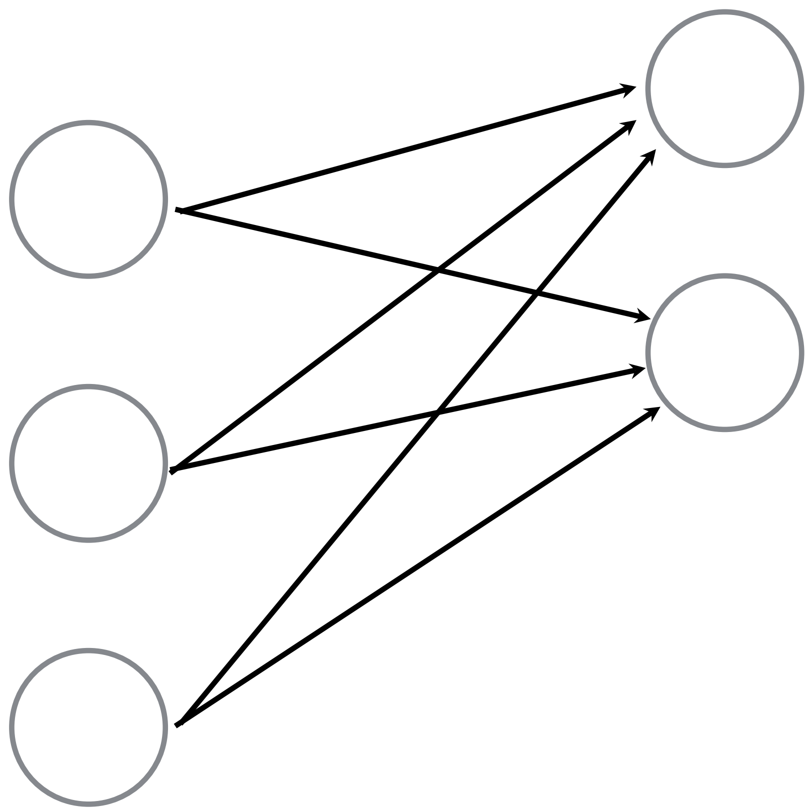


‘one perceptron’

Connect a bunch of perceptrons together …

Neural Network

a collection of connected perceptrons

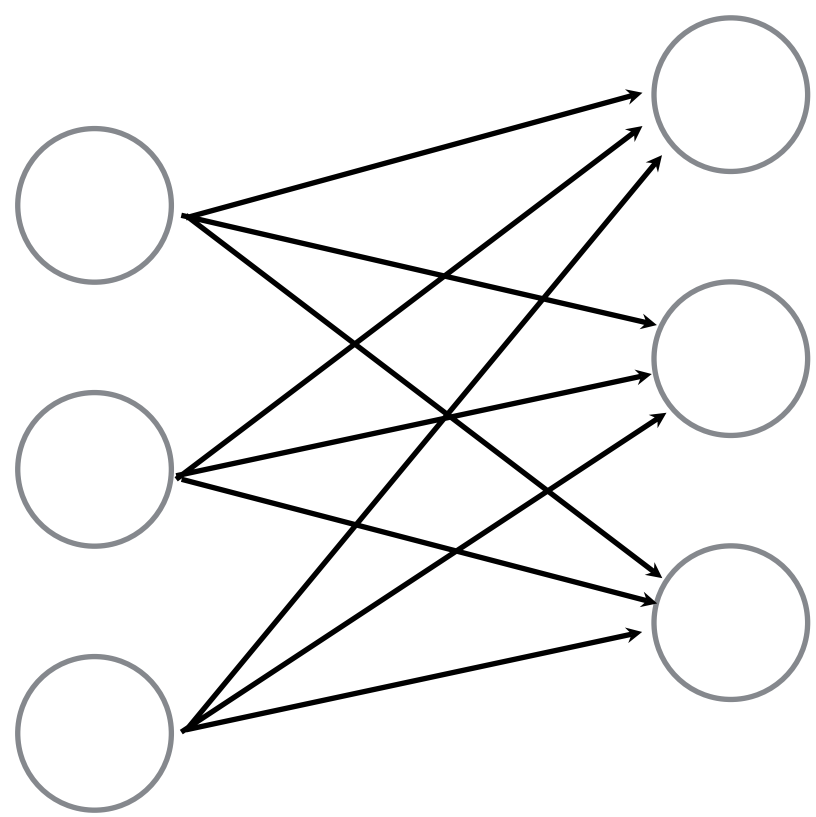


‘two perceptrons’

Connect a bunch of perceptrons together …

Neural Network

a collection of connected perceptrons

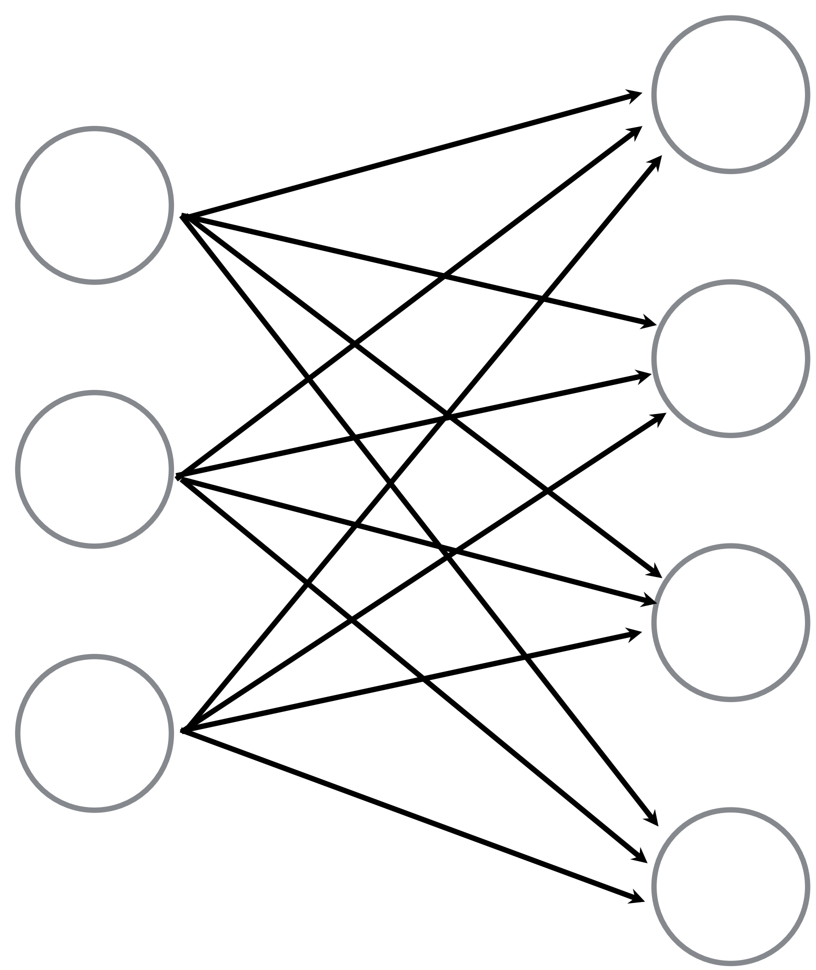


‘three perceptrons’

Connect a bunch of perceptrons together …

Neural Network

a collection of connected perceptrons

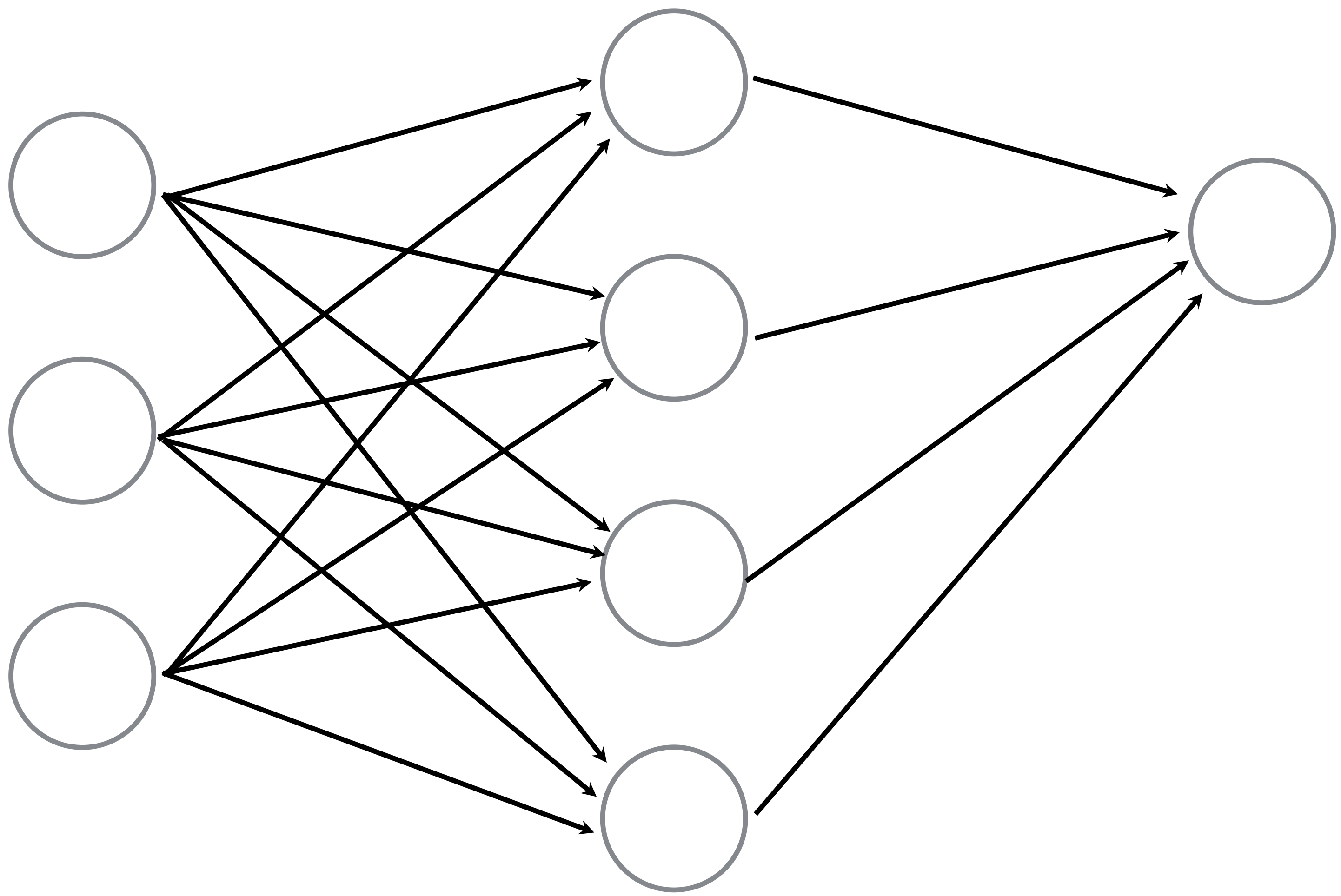


'four perceptrons'

Connect a bunch of perceptrons together …

Neural Network

a collection of connected perceptrons

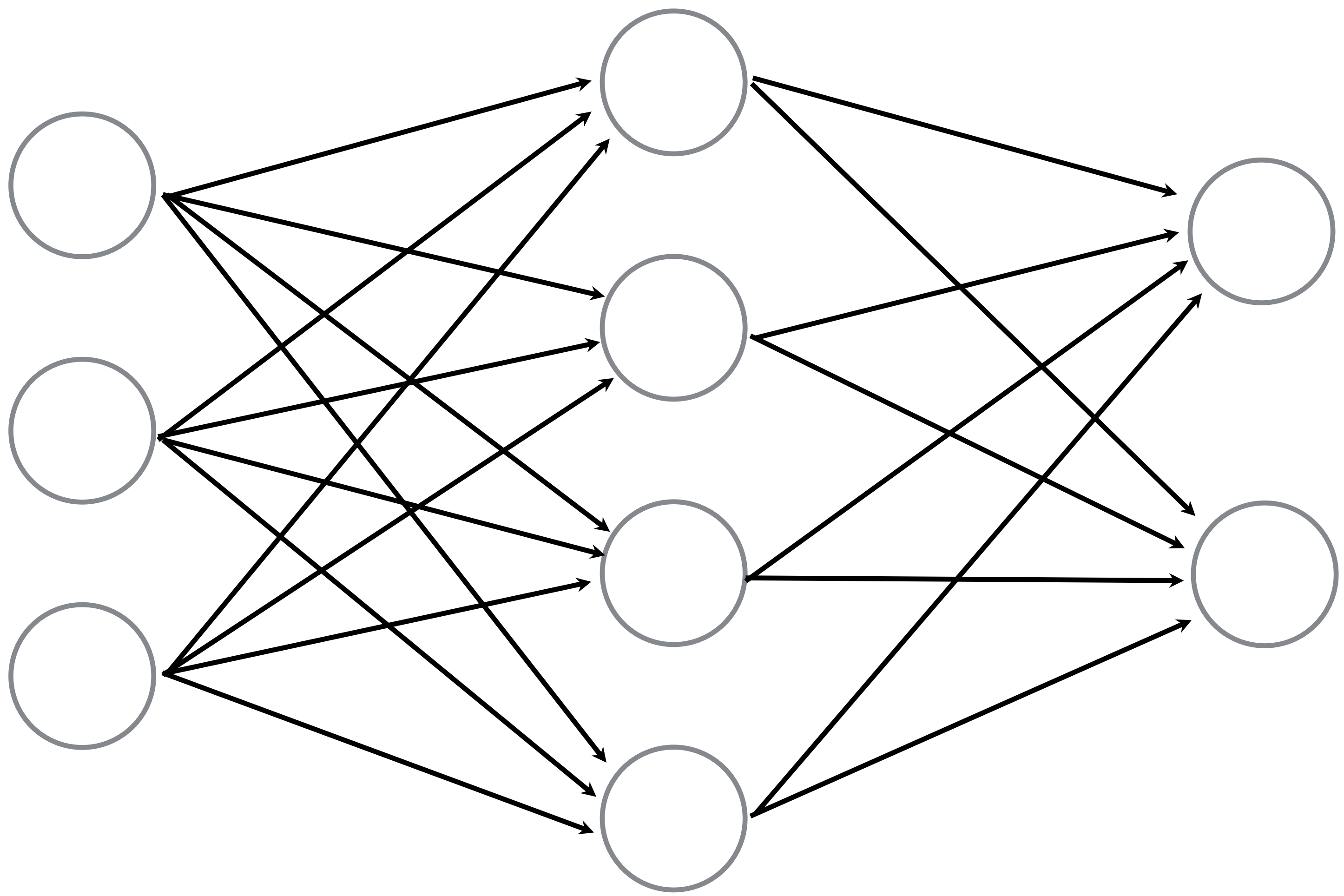


'five perceptrons'

Connect a bunch of perceptrons together …

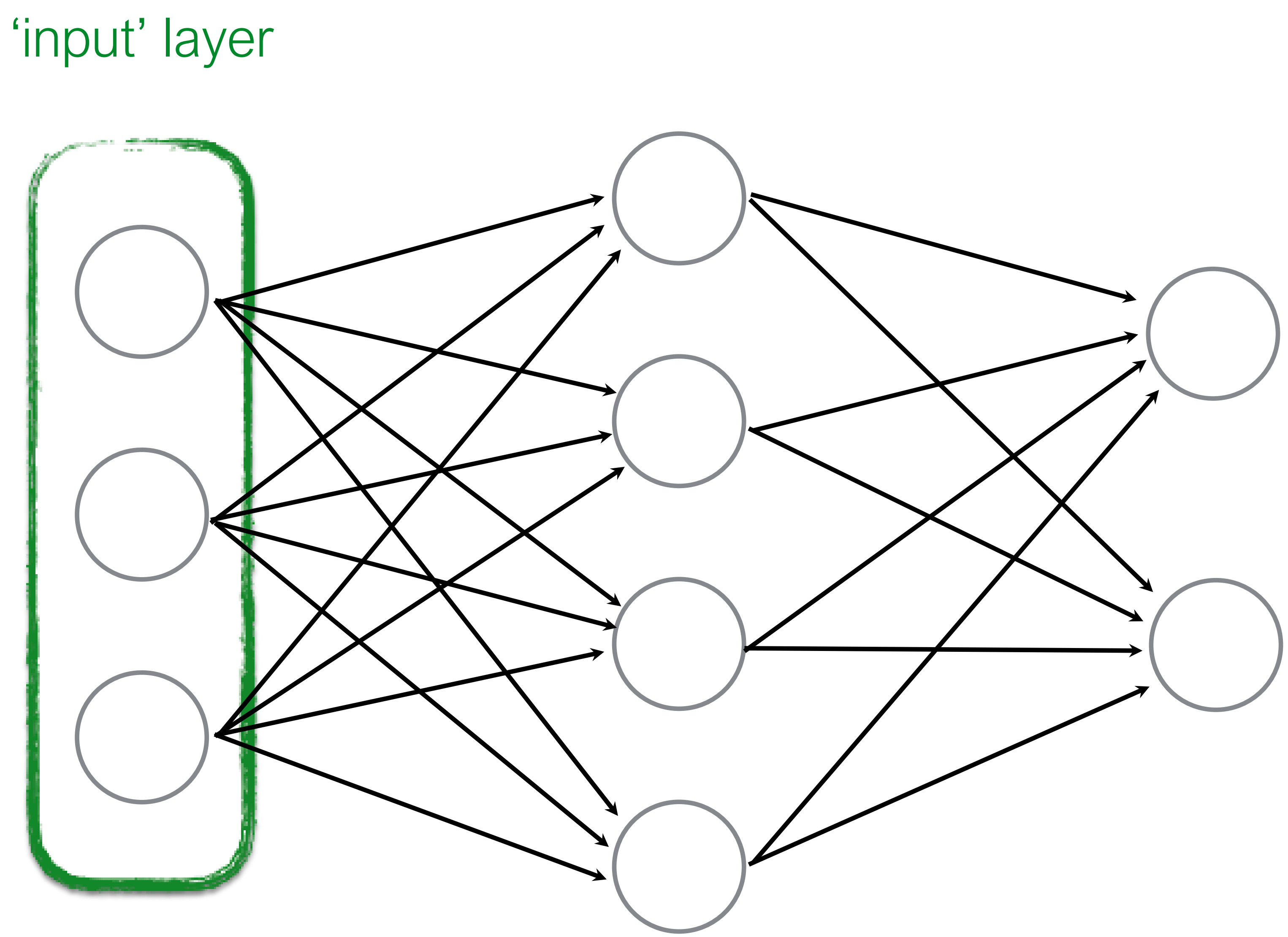
Neural Network

a collection of connected perceptrons



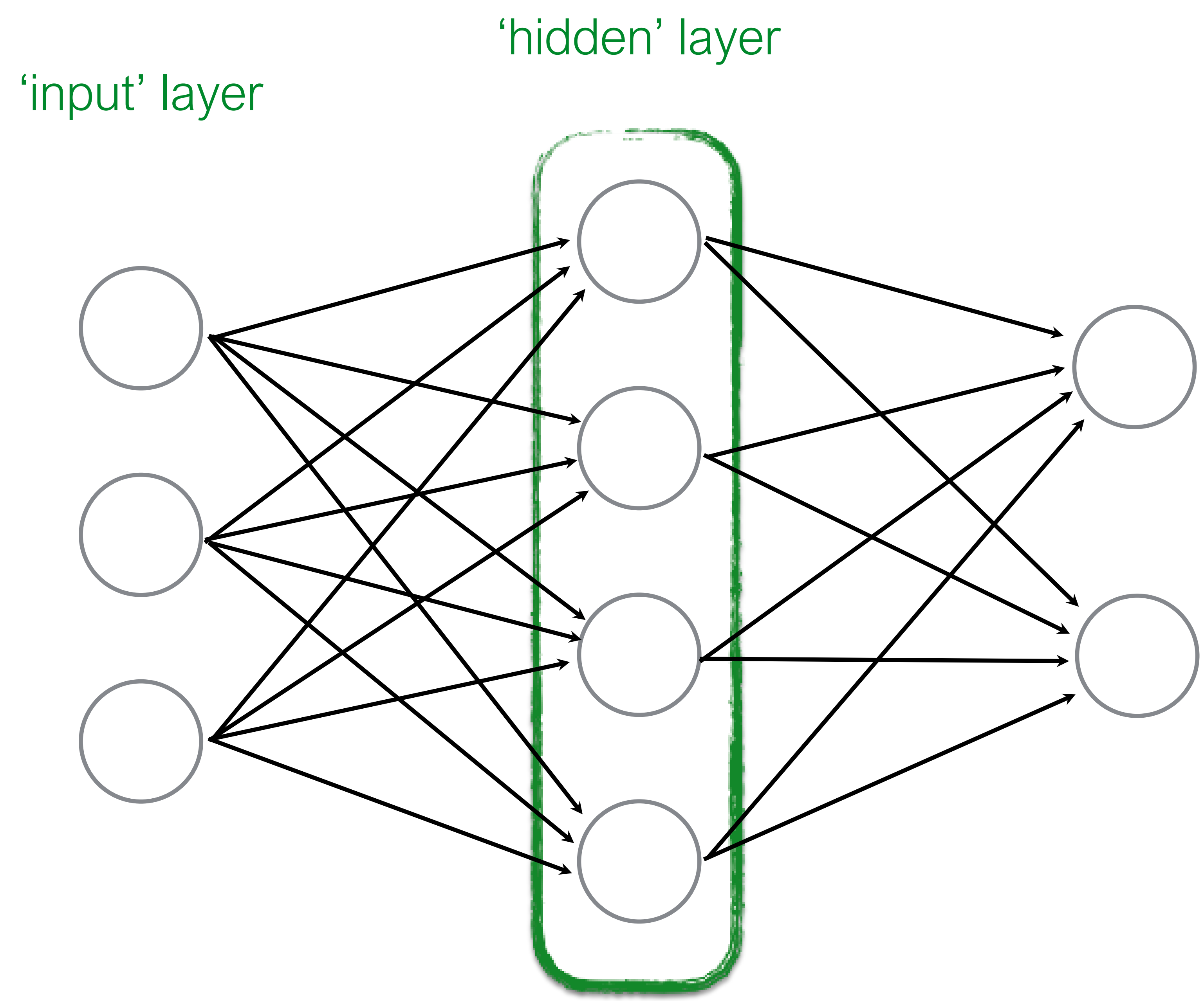
'six perceptrons'

Some terminology…



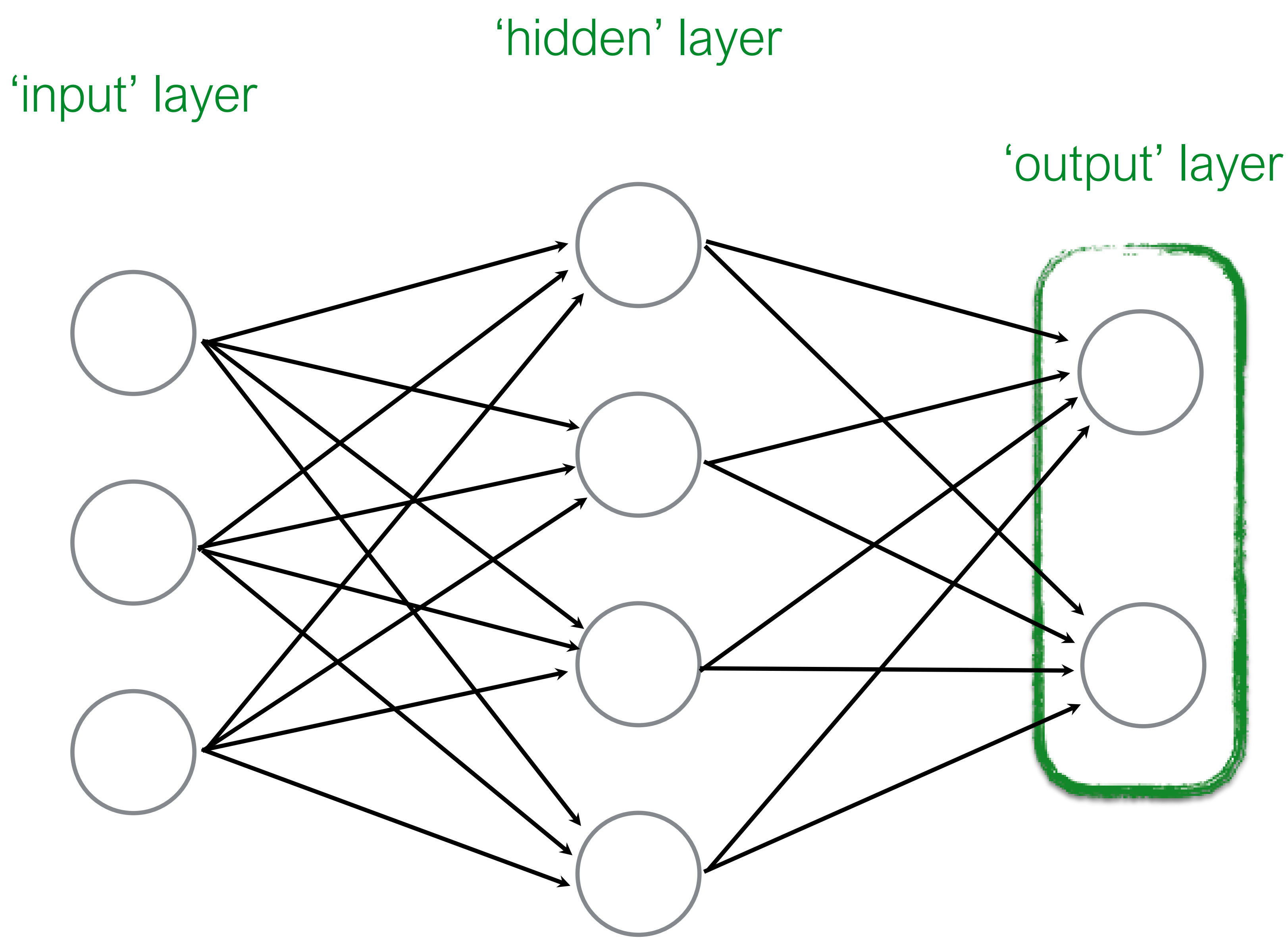
…also called a **Multi-layer Perceptron** (MLP)

Some terminology…



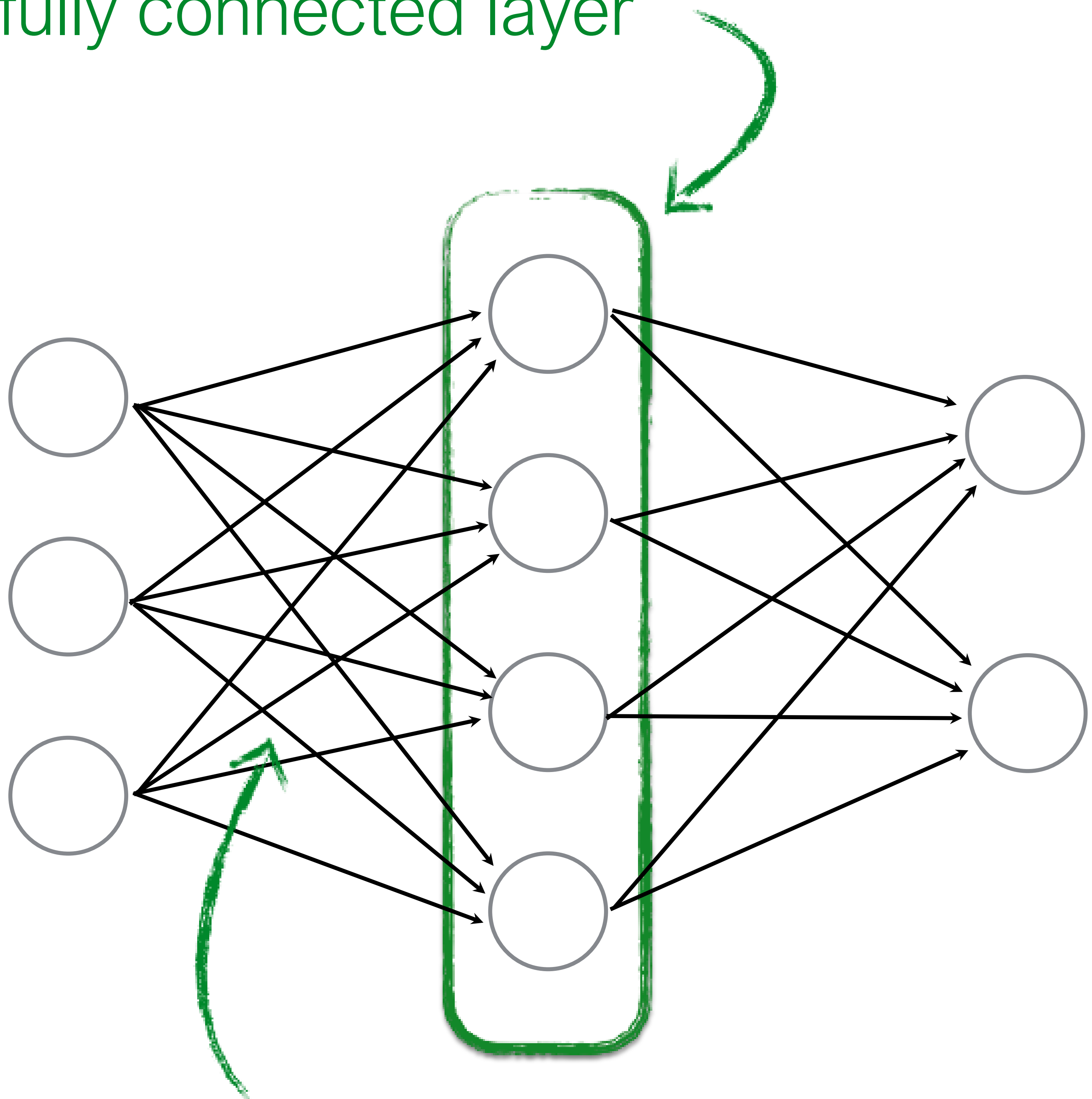
…also called a **Multi-layer Perceptron** (MLP)

Some terminology…

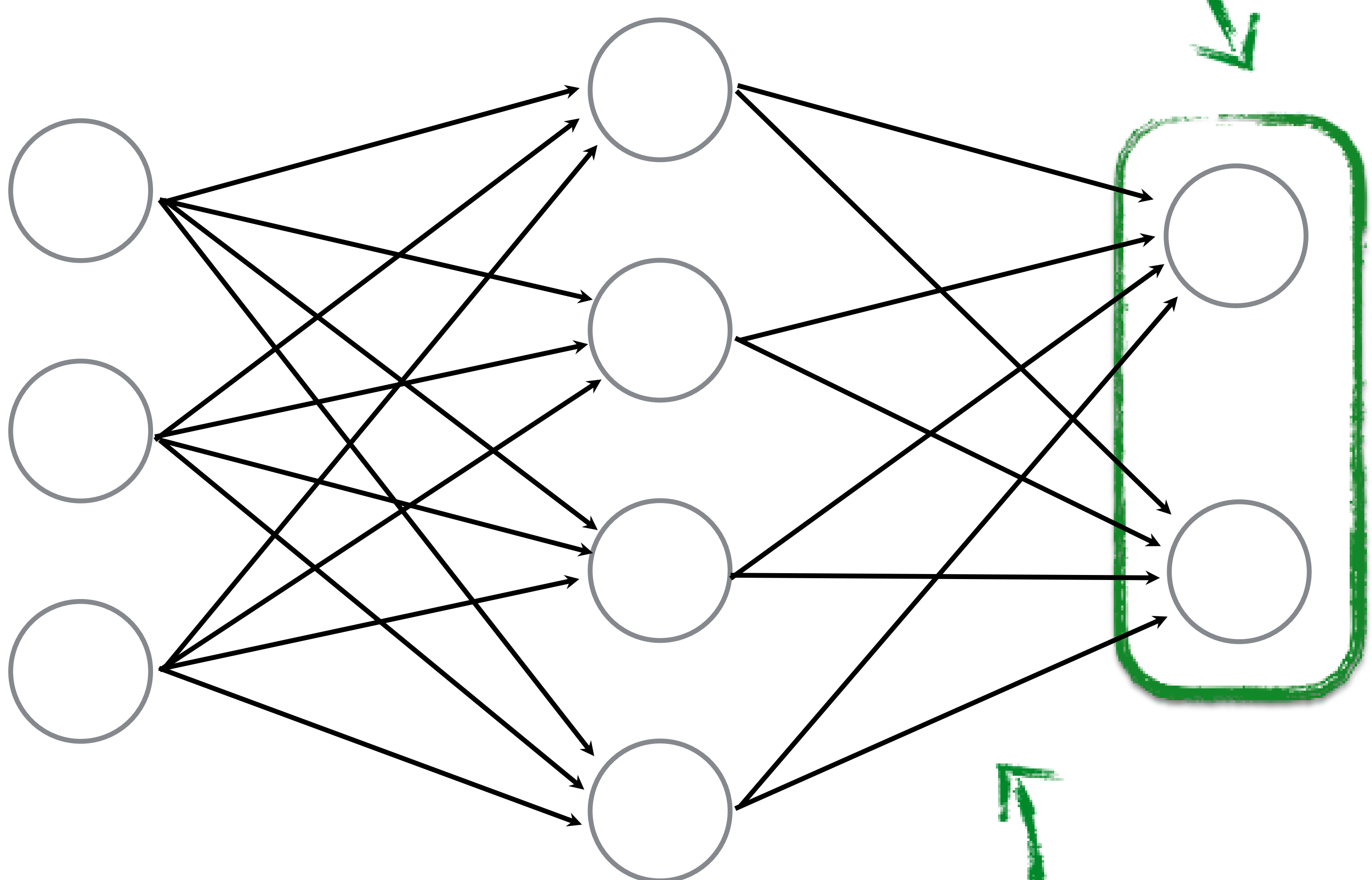


…also called a **Multi-layer Perceptron** (MLP)

this layer is a
'fully connected layer'



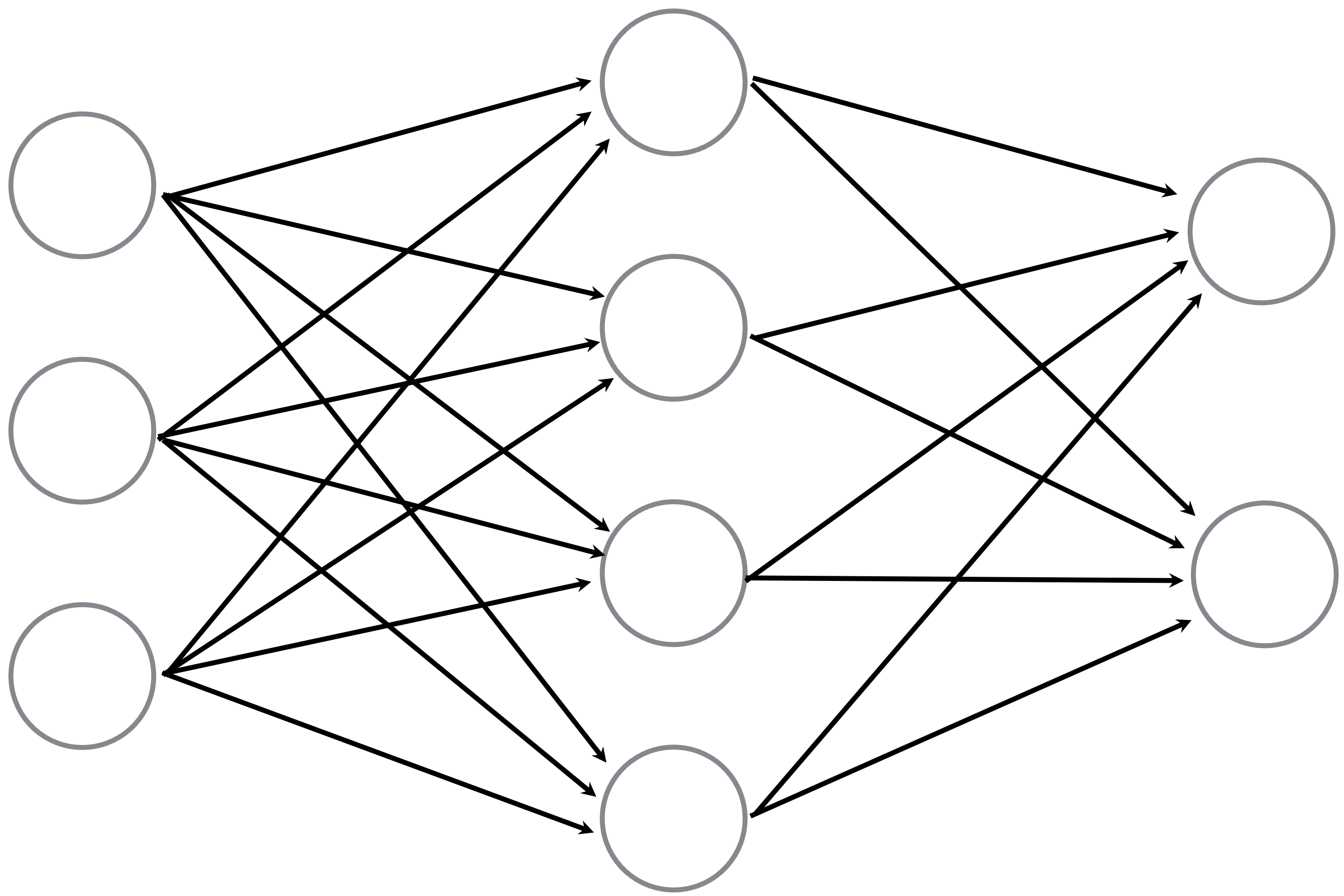
all pairwise neurons between layers are connected so is this



all pairwise neurons between layers are connected

How many neurons (perceptrons)?

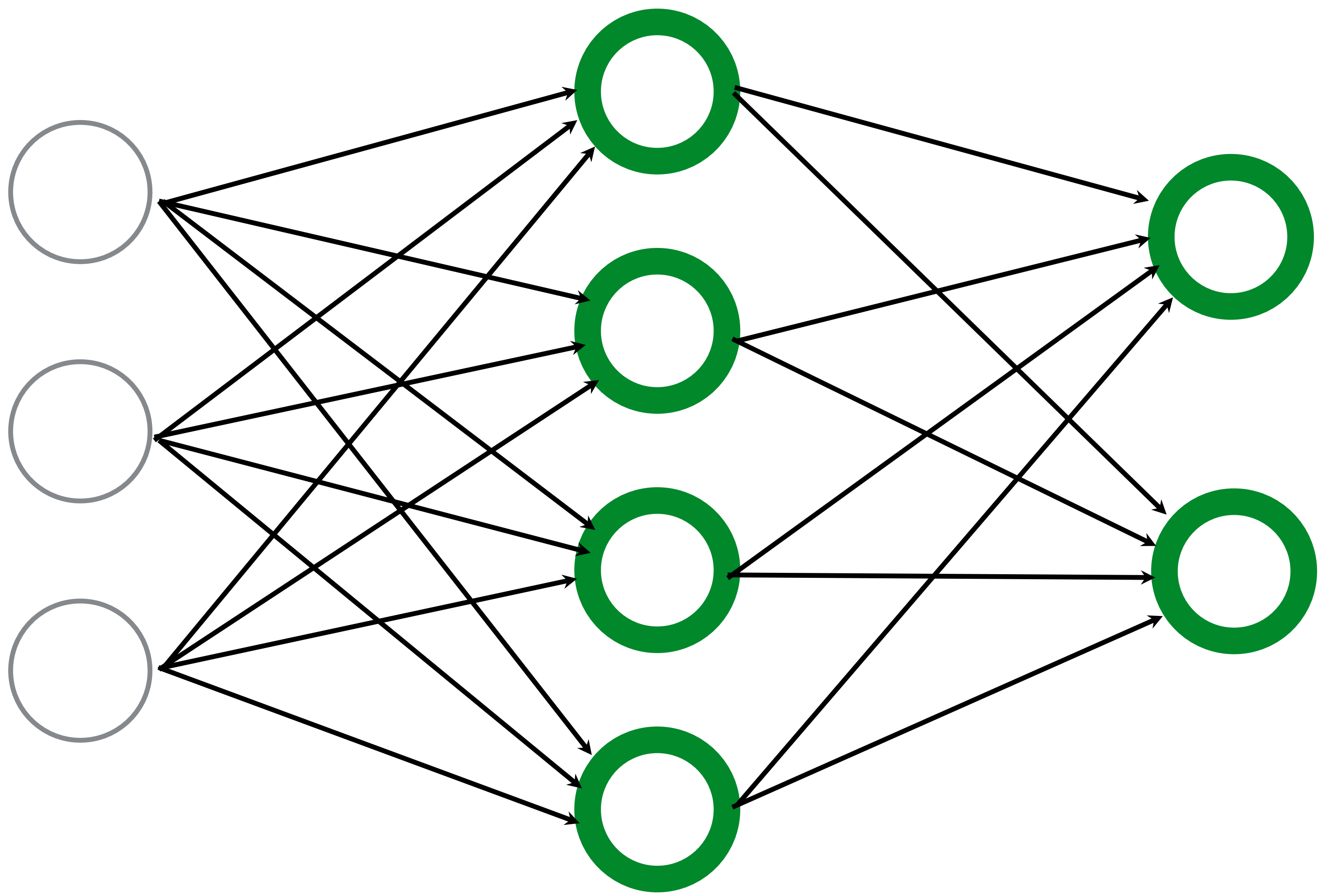
How many weights (edges)?



How many learnable parameters total?

How many neurons (perceptrons)? 4 + 2 = 6

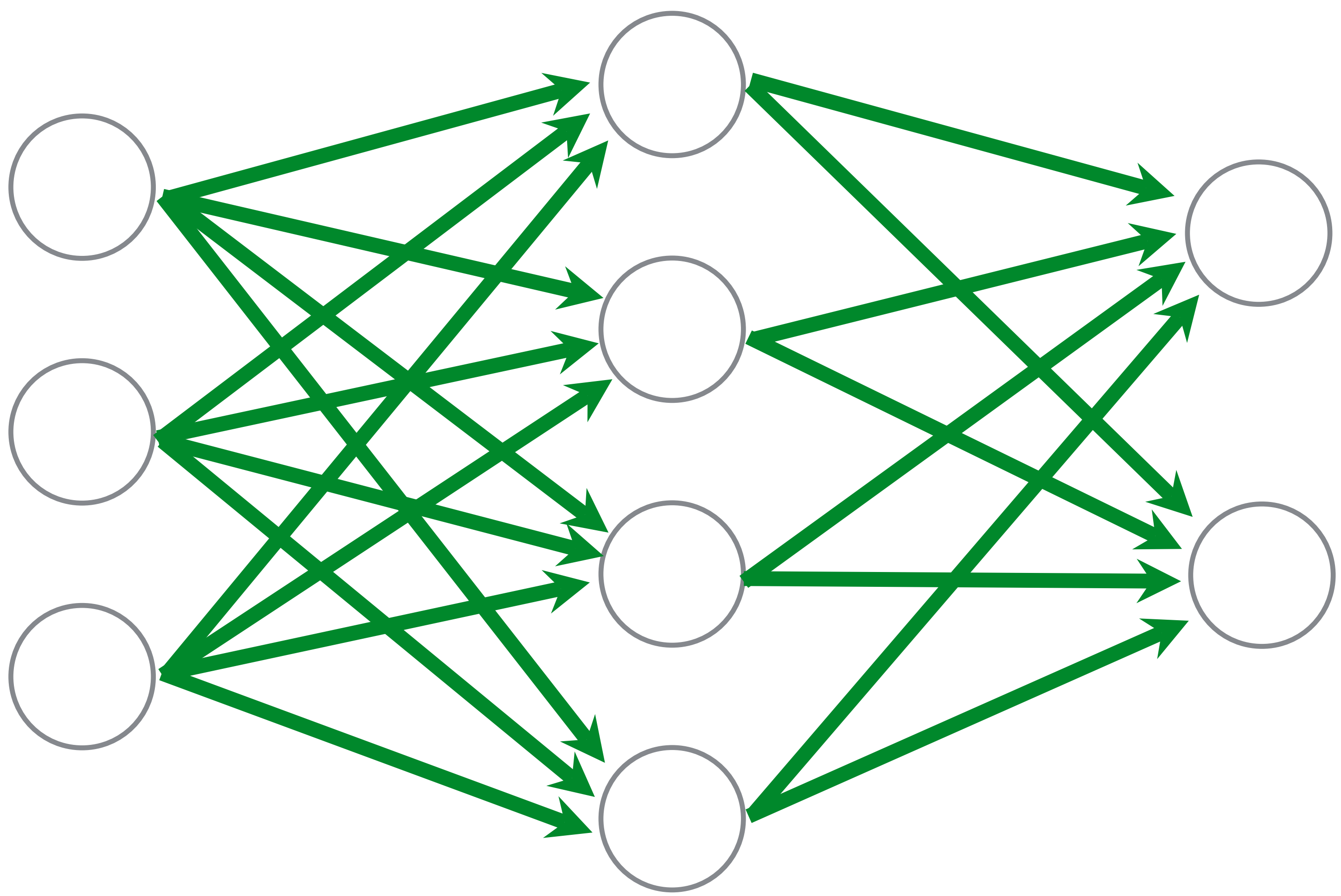
How many weights (edges)?



How many learnable parameters total?

How many neurons (perceptrons)? $4 + 2 = 6$

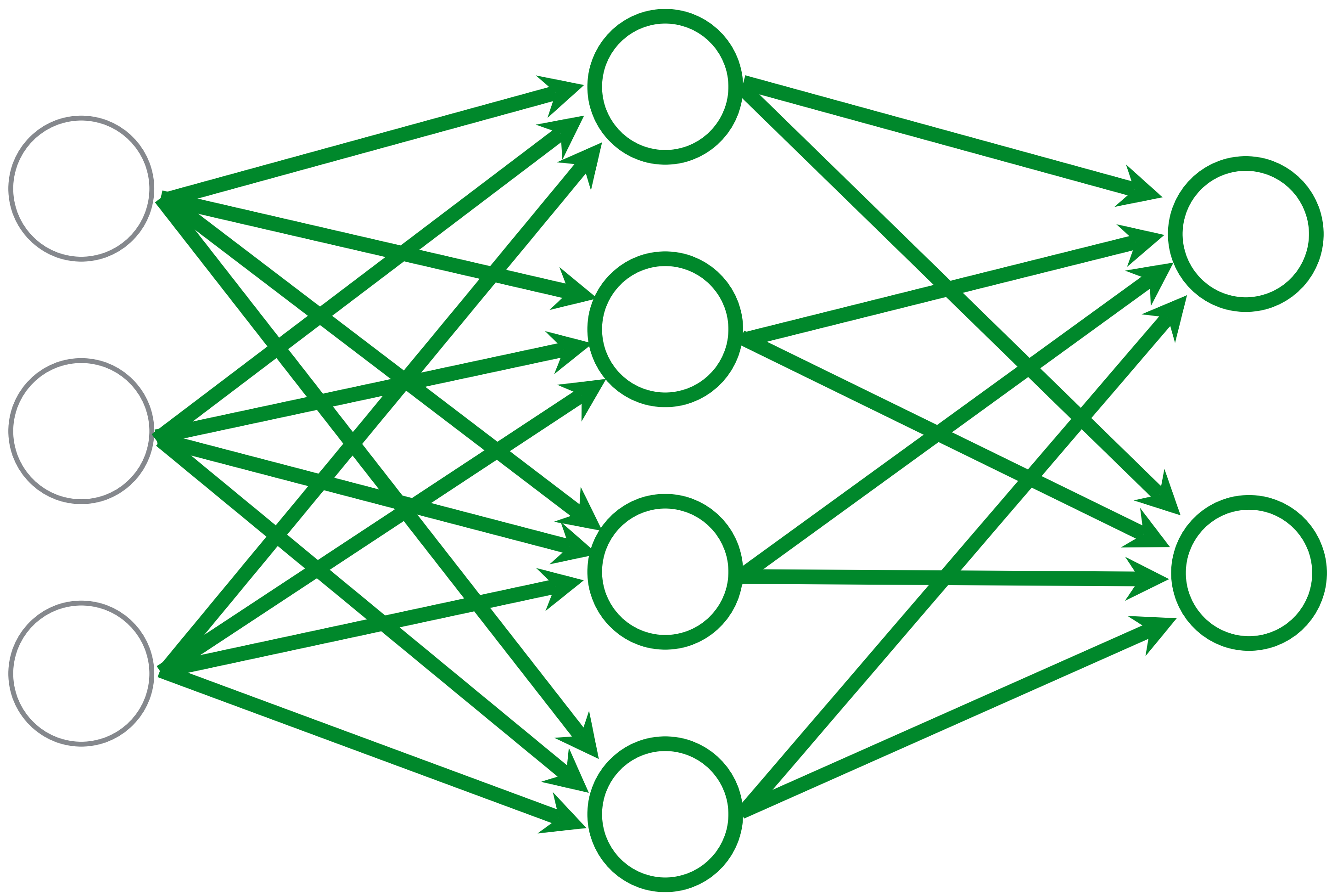
How many weights (edges)? $(3 \times 4) + (4 \times 2) = 20$



How many learnable parameters total?

How many neurons (perceptrons)? 4 + 2 = 6

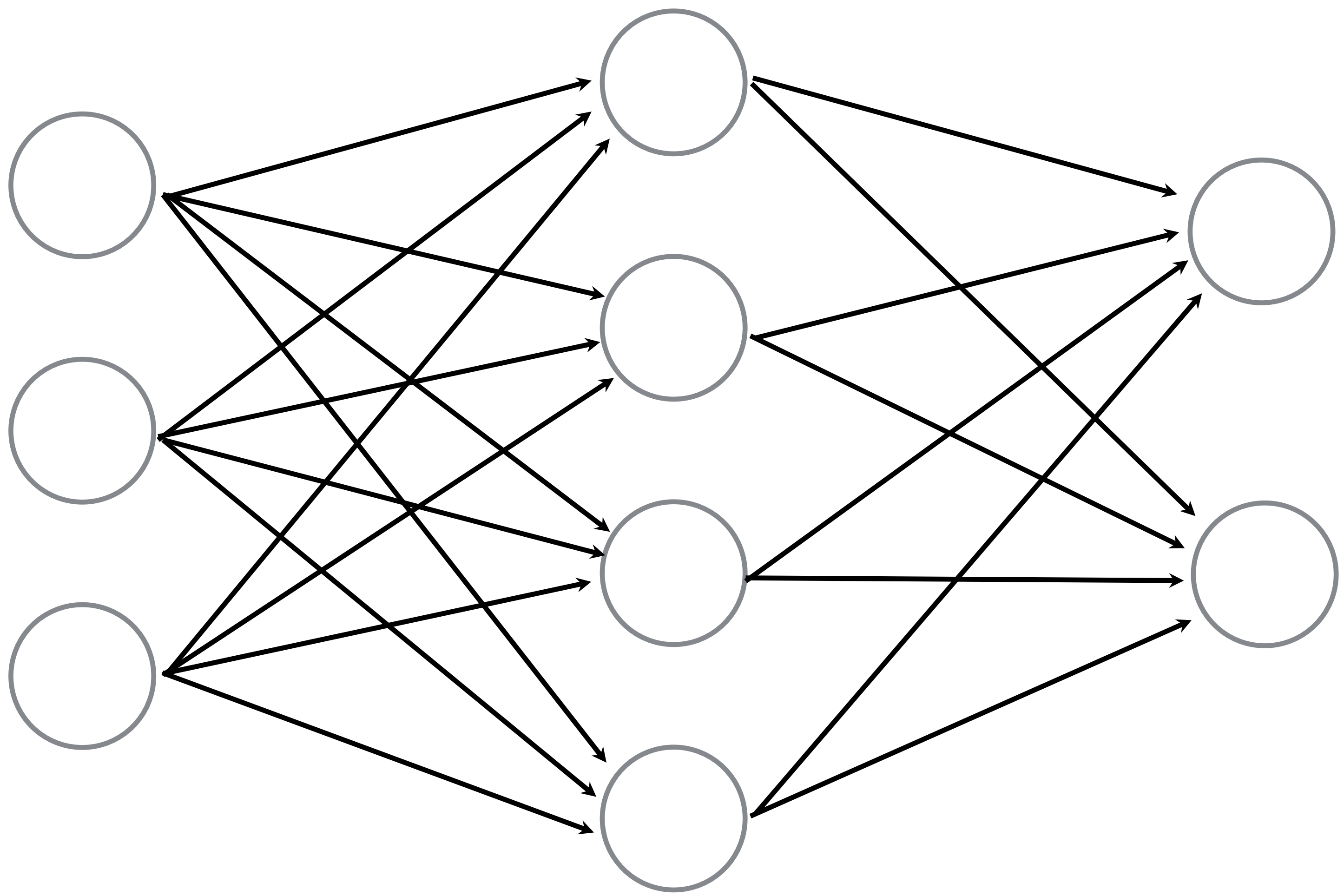
How many weights (edges)? (3 x 4) + (4 x 2) = 20



How many learnable parameters total? 20 + 4 + 2 = 26

bias terms

performance usually tops out at 2-3 layers,
deeper networks don't really improve performance...

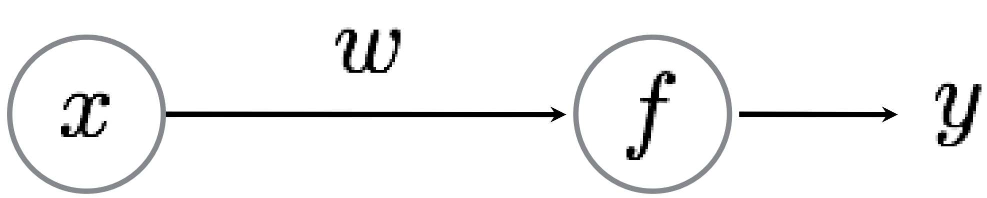


...with the exception of **convolutional** networks for images

Training perceptrons

Let’s start easy

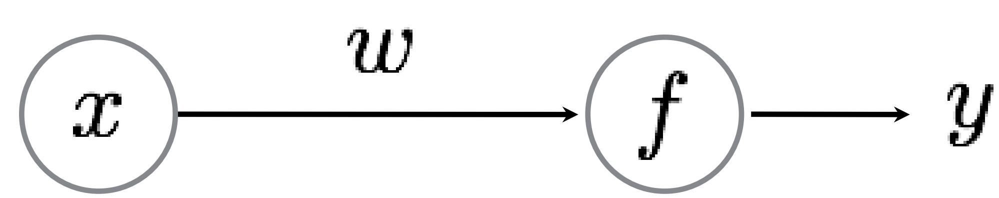
world's smallest perceptron!



$$y = wx$$

What does this look like?

world's smallest perceptron!



$$y = wx$$

(a.k.a. line equation, linear regression)

Learning a Perceptron

Given a set of samples and a Perceptron

$$\{x_i, y_i\}$$

$$y = f_{\text{PER}}(x; w)$$

Estimate the parameters of the Perceptron

$$w$$

Given training data:

x	y
10	10.1
2	1.9
3.5	3.4
1	1.1

What do you think the weight parameter is?

$$y = wx$$

Given training data:

x	y
10	10.1
2	1.9
3.5	3.4
1	1.1

What do you think the weight parameter is?

$$y = wx$$

not so obvious as the network gets more complicated so we use …

An Incremental Learning Strategy

(gradient descent)

Given several examples

$$\{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$$

and a perceptron

$$\hat{y} = wx$$

An Incremental Learning Strategy

(gradient descent)

Given several examples

$$\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$$

and a perceptron

$$\hat{y} = wx$$

Modify weight w such that $\hat{y}$ gets **'closer'** to y

An Incremental Learning Strategy

(gradient descent)

Given several examples

$$\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$$

and a perceptron

$$\hat{y} = wx$$

Modify weight w such that $\hat{y}$ gets **'closer'** to y

perceptron parameter

perceptron output

true label

An Incremental Learning Strategy

(gradient descent)

Given several examples

$$\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$$

and a perceptron

$$\hat{y} = wx$$

Modify weight w such that $\hat{y}$ gets **'closer'** to y

perceptron parameter

perceptron output

what does this mean?

true label

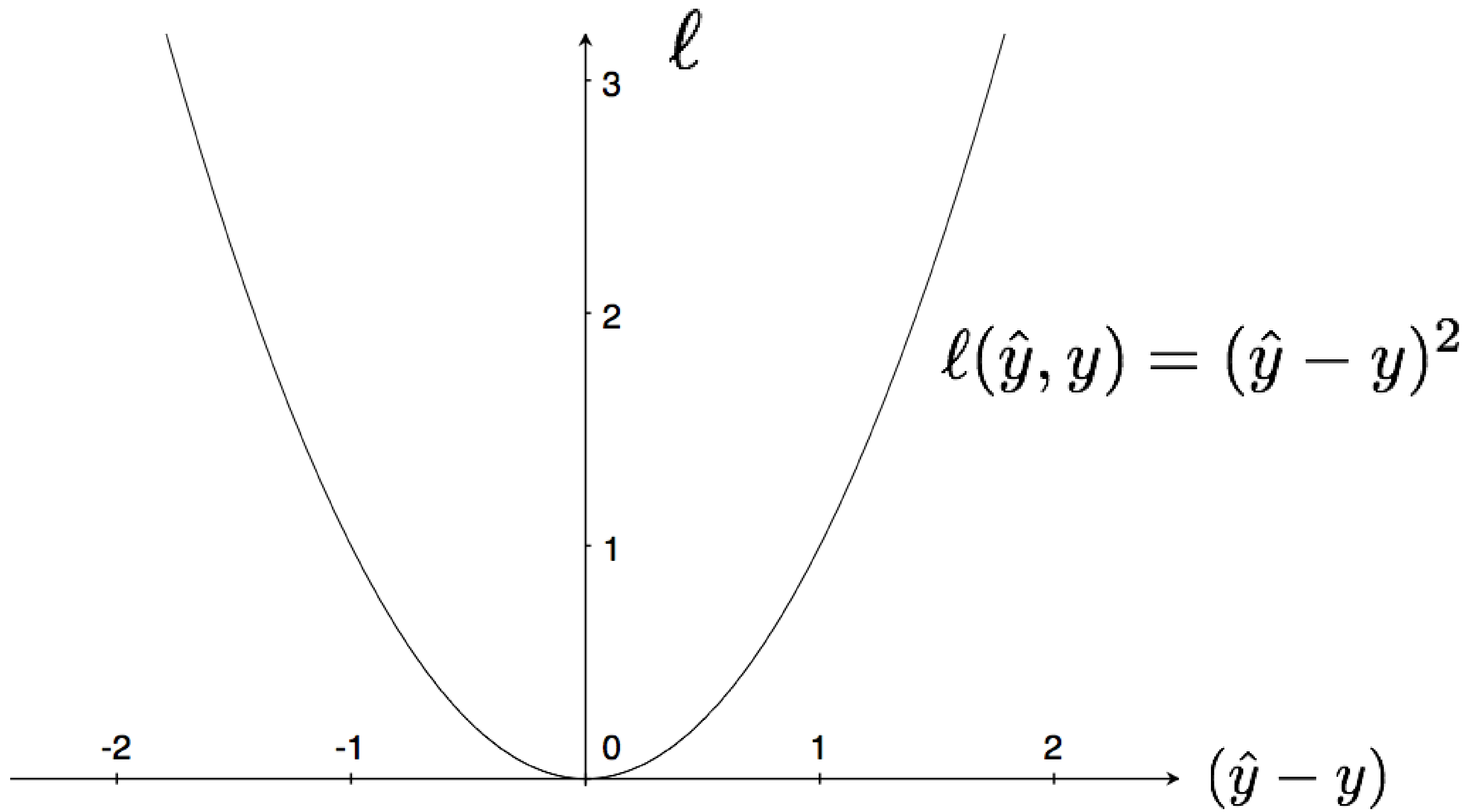
Before diving into gradient descent, we need to understand …

Loss Function
defines what is means to be
close to the true solution

YOU get to chose the loss function!
(some are better than others depending on what you want to do)

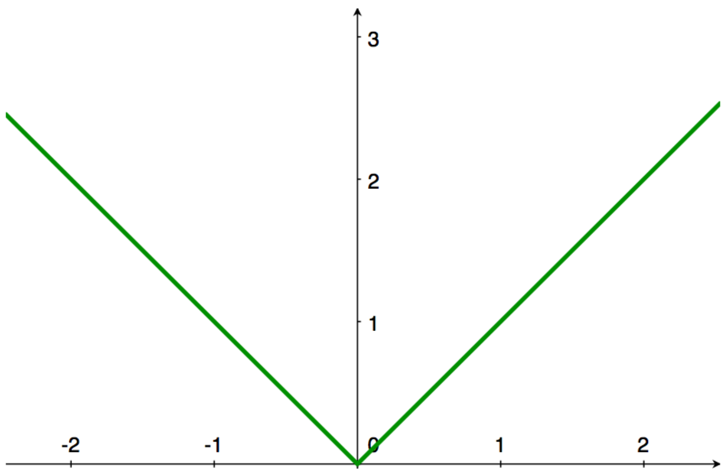
Squared Error (L2)

(a popular loss function) ((why?))



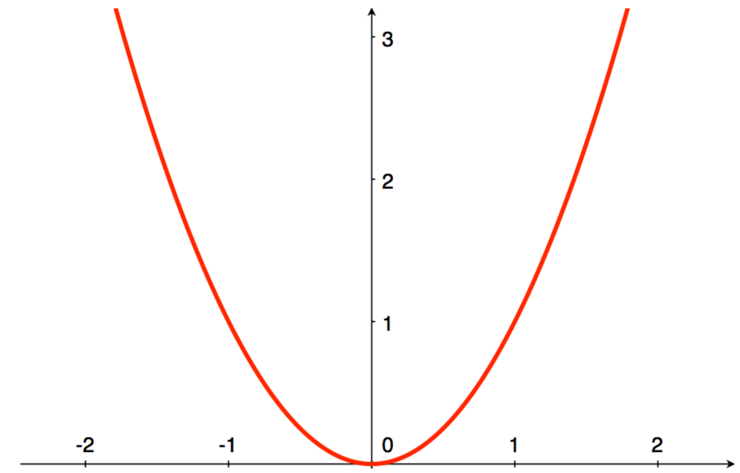
L1 Loss

$$\ell(\hat{y}, y) = |\hat{y} - y|$$



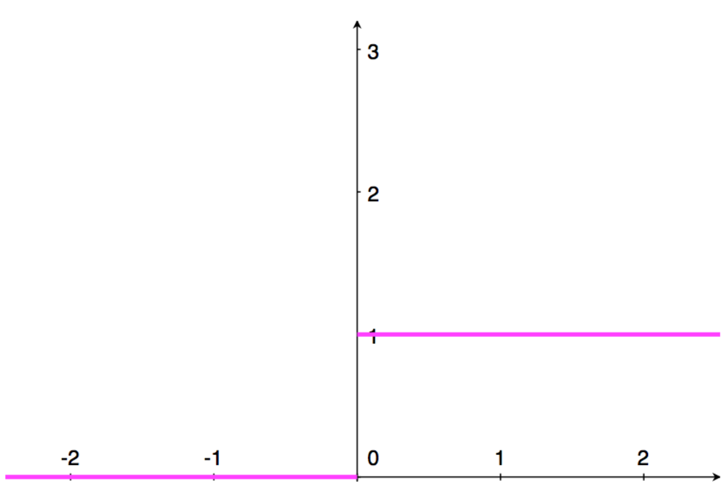
L2 Loss

$$\ell(\hat{y}, y) = (\hat{y} - y)^2$$



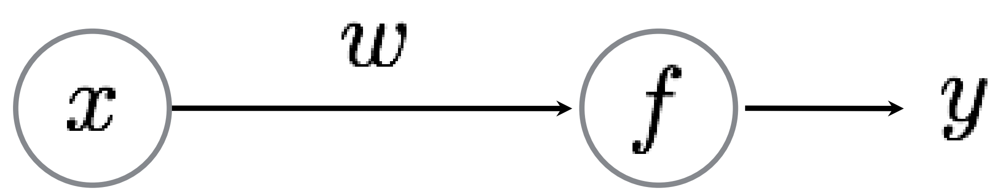
Zero-One Loss

$$\ell(\hat{y}, y) = \mathbf{1}[\hat{y} = y]$$



back to the…

World's Smallest Perceptron!



$$y = wx$$

(a.k.a. line equation, linear regression)

function of **ONE** parameter!

Learning a Perceptron

Given a set of samples and a Perceptron

$$\{x_i, y_i\}$$

$$y = f_{\mathrm{PER}}(x; w)$$

what is this activation function?

Estimate the parameter of the Perceptron

$$w$$

Learning a Perceptron

Given a set of samples and a Perceptron

$$\{x_i, y_i\}$$

$$y = f_{\text{PER}}(x; w)$$

what is this activation function?

linear function! $f(x) = wx$

Estimate the parameter of the Perceptron

$$w$$

Learning Strategy

(gradient descent)

Given several examples

$$\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$$

and a perceptron

$$\hat{y} = wx$$

Modify weight w such that $\hat{y}$ gets **'closer'** to y

perceptron parameter

perceptron output

true label

Let's demystify this process first…

Code to train your perceptron:

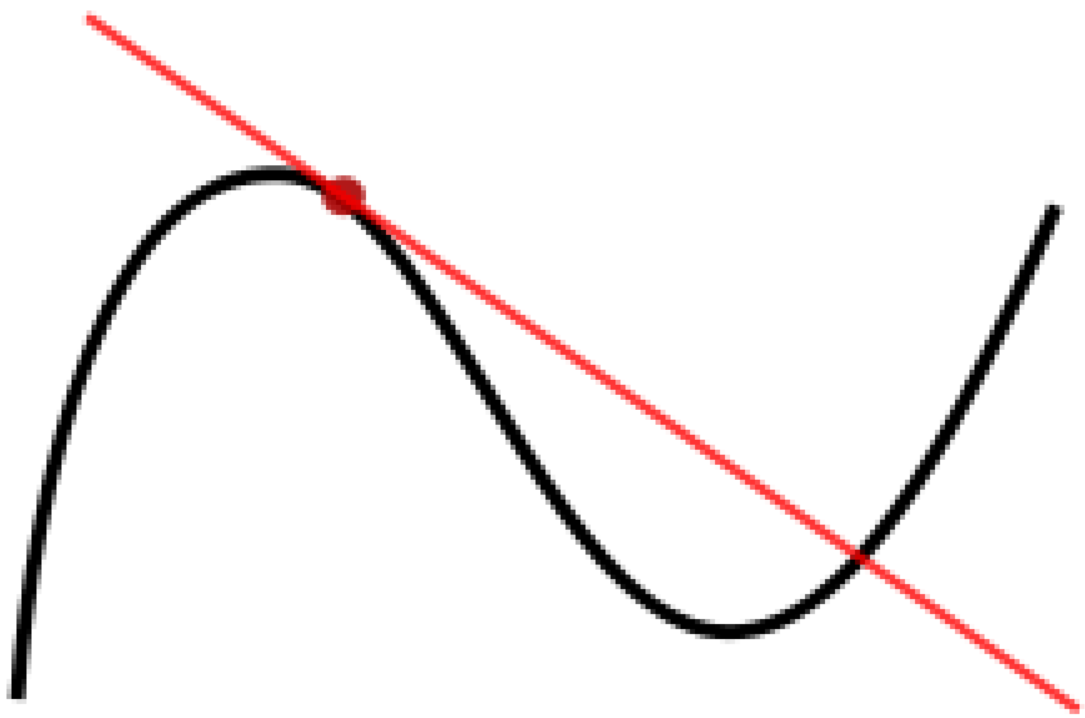
$$\text{for } n = 1 \ldots N$$
$$w = w + (y_n - \hat{y})x_n;$$

just one line of code!

Now where does this come from?

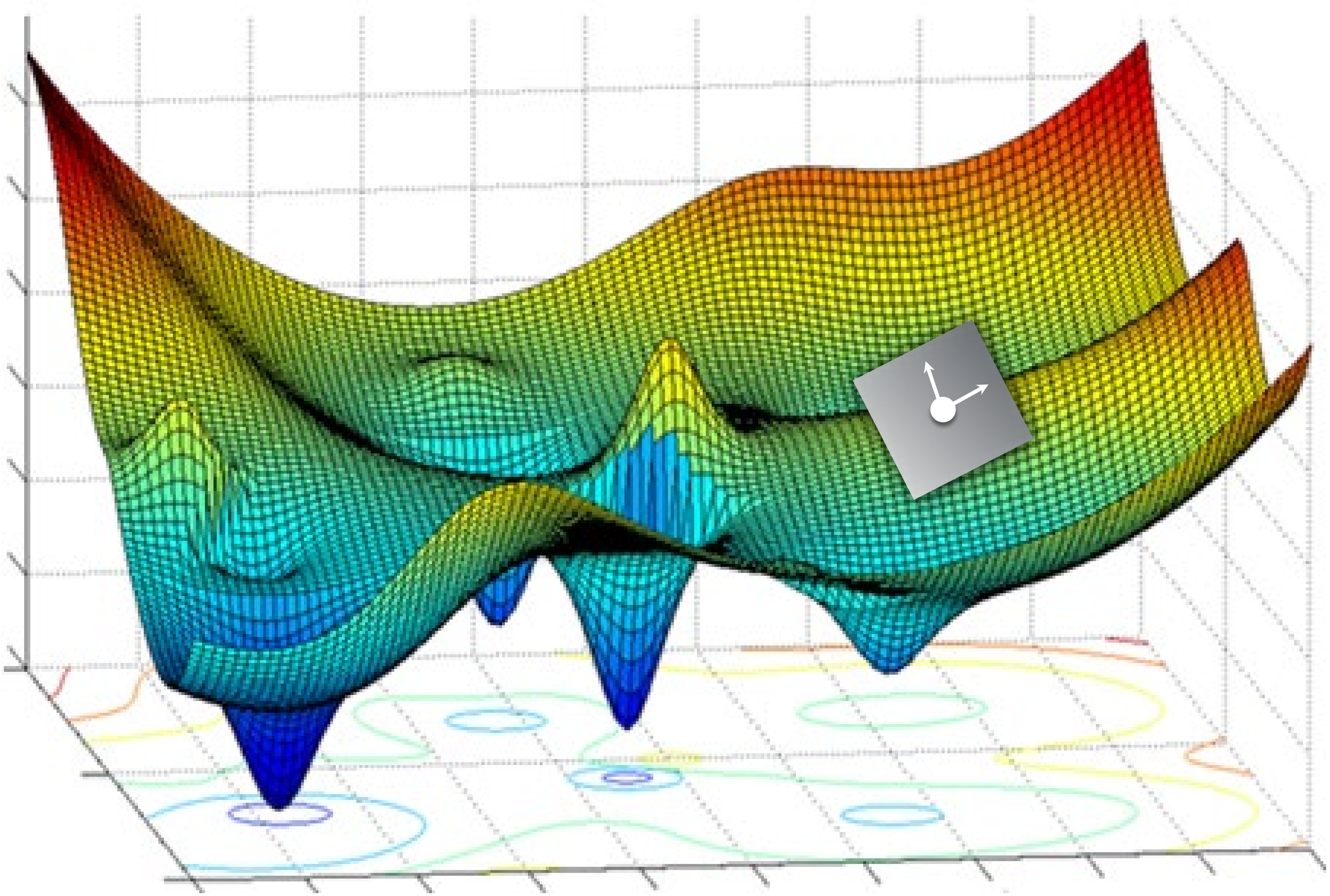
Gradient descent

(partial) derivatives tell us how much one variable affects the function



$$f'(a) = \lim_{h \to 0} \frac{f(a+h) - f(a)}{h}.$$

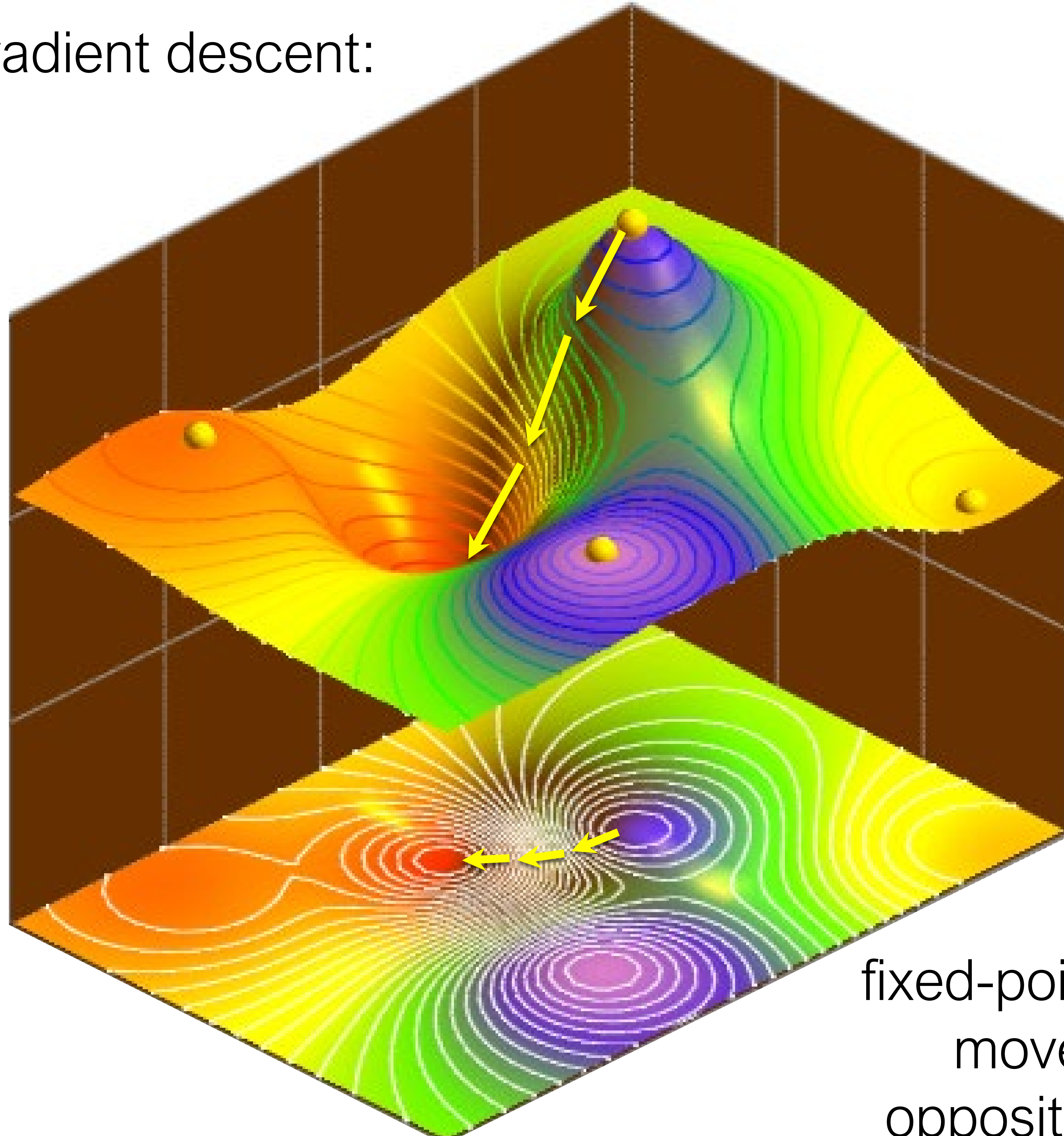
Slope of a function:



$$\frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}} = \left[\frac{\partial f(\boldsymbol{x})}{\partial x}, \frac{\partial f(\boldsymbol{x})}{\partial y}\right]$$

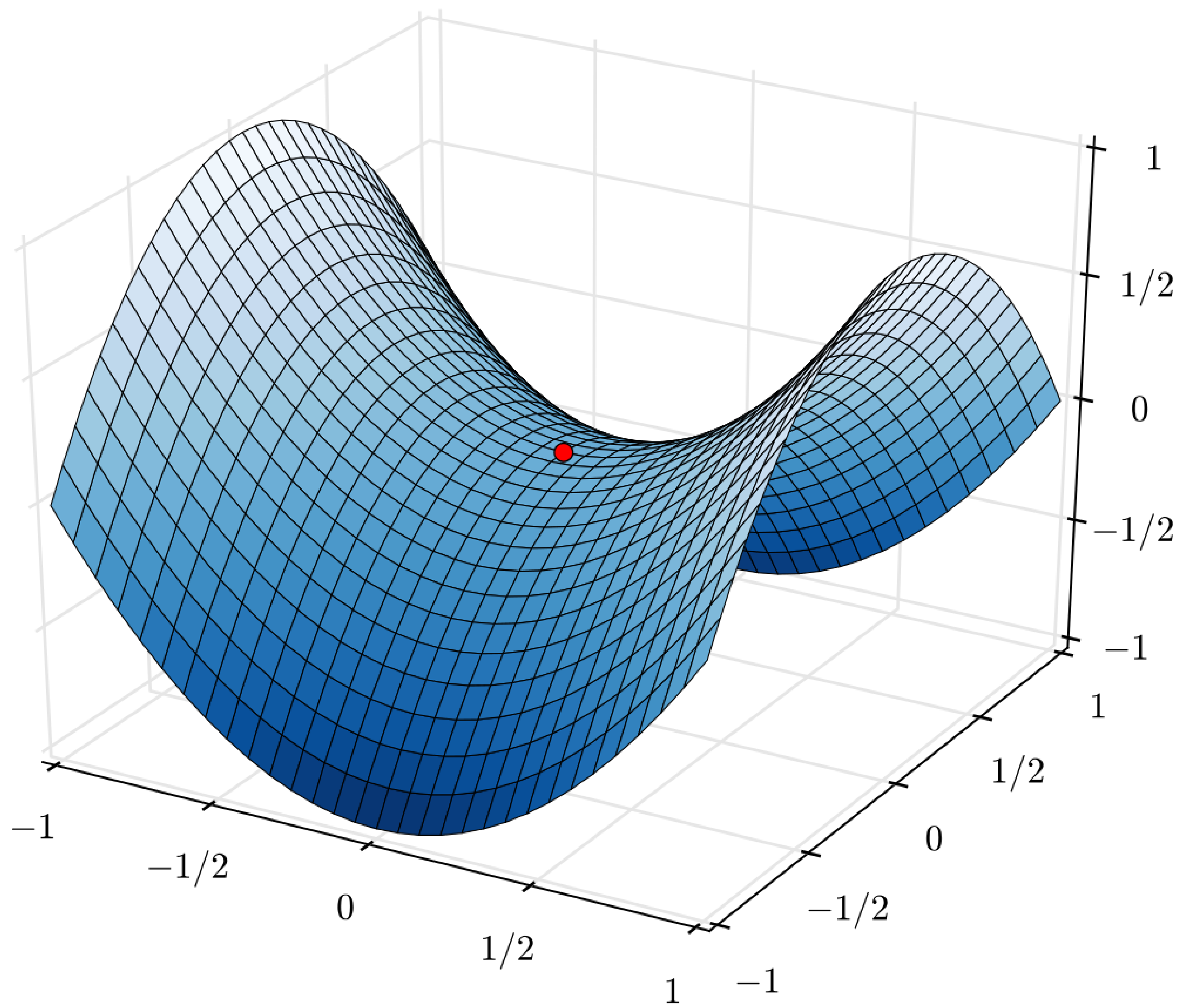
describes the slope around a point

Gradient descent:

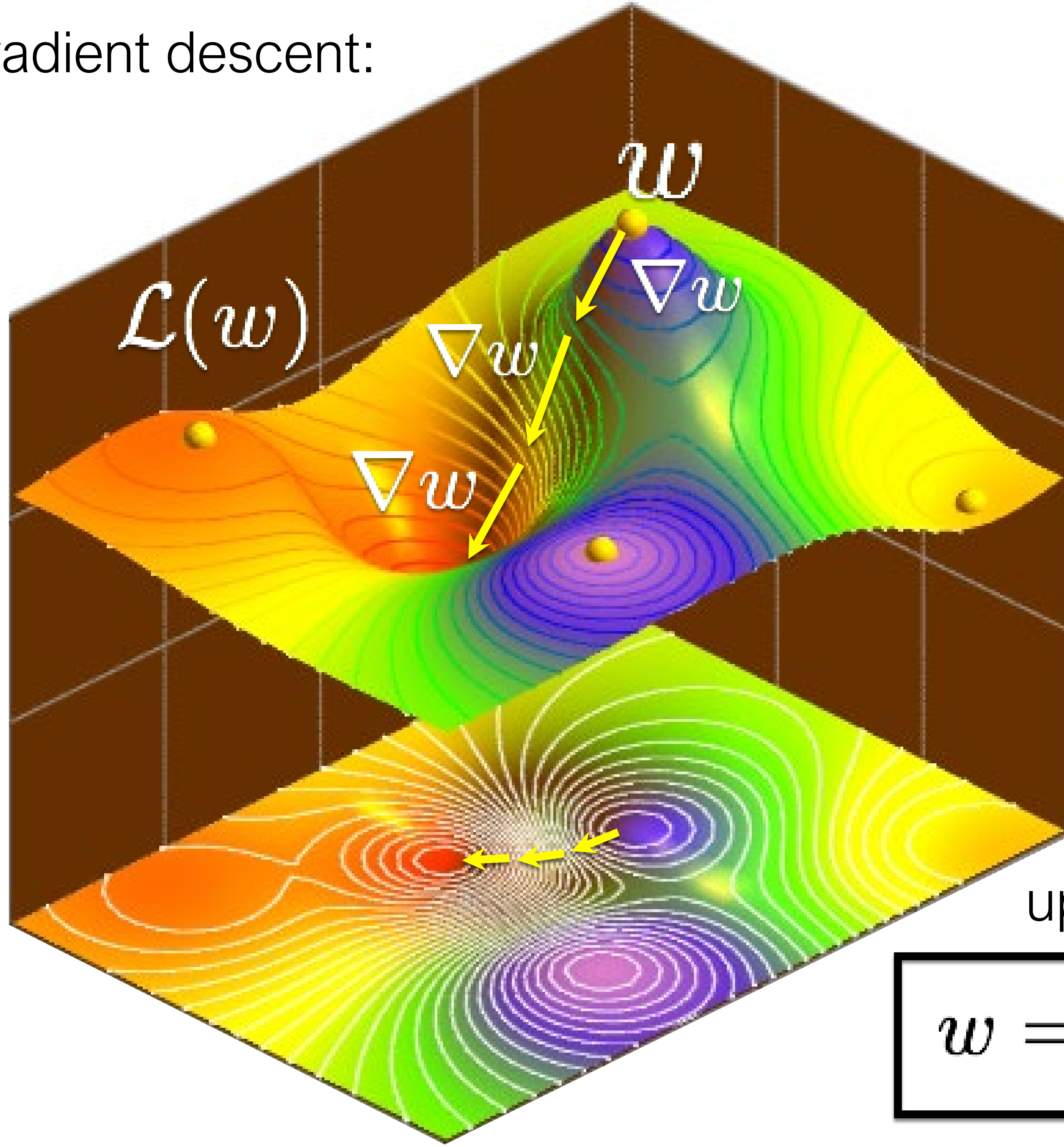


Given a
fixed-point on a function,
move in the direction
opposite of the gradient

Saddle point



Gradient descent:



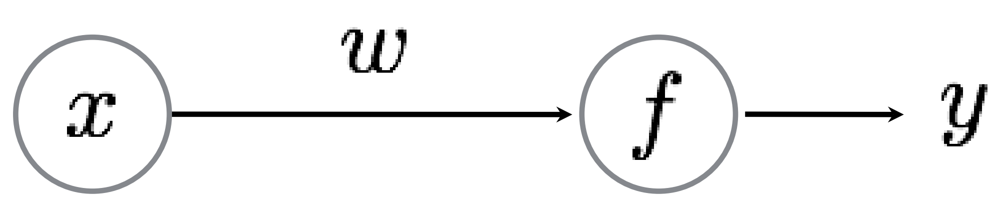
update rule:

$$\boxed{w = w - \nabla w}$$

Backpropagation

back to the…

World's Smallest Perceptron!



$$y = wx$$

(a.k.a. line equation, linear regression)

function of **ONE** parameter!

Training the world's smallest perceptron

$$\text{for } n = 1 \ldots N$$

$$w = w + \underline{(y_n - \hat{y})x_n};$$

This is just gradient descent, that means…

this should be the gradient of the loss function

Now where does this come from?

$$\frac{d\mathcal{L}}{dw}$$

...is the rate at which **this** will change...

$$\mathcal{L} = \frac{1}{2}(y - \hat{y})^2$$

the loss function

... per unit change of **this**

$$y = wx$$

the weight parameter

Let's compute the derivative...

Compute the derivative

$$\frac{d\mathcal{L}}{dw} = \frac{d}{dw}\left\{\frac{1}{2}(y-\hat{y})^2\right\}$$

$$= -(y-\hat{y})\frac{dwx}{dw}$$

$$= -(y-\hat{y})x = \nabla w$$ just shorthand

That means the weight update for **gradient descent** is:

$$w = w - \nabla w$$ move in direction of negative gradient

$$= w + (y-\hat{y})x$$

Gradient Descent (world's smallest perceptron)

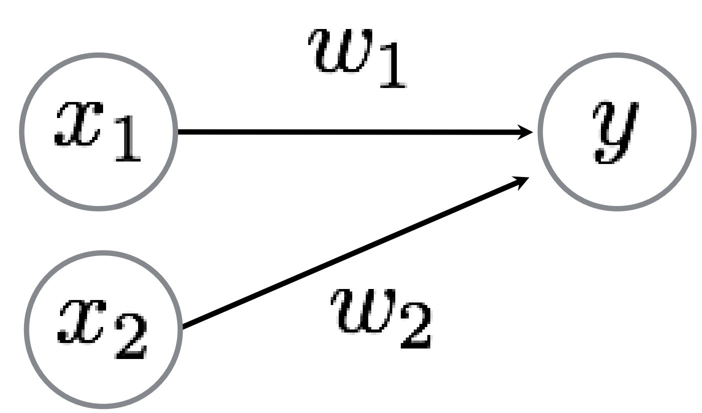
For each sample $\{x_i, y_i\}$

1. Predict
   a. Forward pass $\hat{y} = wx_i$
   b. Compute Loss $\mathcal{L}_i = \frac{1}{2}(y_i - \hat{y})^2$
2. Update
   a. Back Propagation $\frac{d\mathcal{L}_i}{dw} = -(y_i - \hat{y})x_i = \nabla w$
   b. Gradient update $w = w - \nabla w$

Training the world's smallest perceptron

$$\text{for } n = 1 \ldots N$$
$$w = w + (y_n - \hat{y})x_n;$$

world's (second) smallest **perceptron**!



function of **two** parameters!

Gradient Descent

For each sample $\{x_i, y_i\}$

1. Predict
   a. Forward pass
   b. Compute Loss
2. Update
   a. Back Propagation
   b. Gradient update

we just need to compute partial derivatives for this network

Derivative computation

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial w_1} &= \frac{\partial}{\partial w_1}\left\{\frac{1}{2}(y-\hat{y})^2\right\} \\ &= -(y-\hat{y})\frac{\partial \hat{y}}{\partial w_1} \\ &= -(y-\hat{y})\frac{\partial \sum_i w_i x_i}{\partial w_1} \\ &= -(y-\hat{y})\frac{\partial w_1 x_1}{\partial w_1} \\ &\boxed{= -(y-\hat{y})x_1 = \nabla w_1}\end{aligned}$$

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial w_2} &= \frac{\partial}{\partial w_2}\left\{\frac{1}{2}(y-\hat{y})^2\right\} \\ &= -(y-\hat{y})\frac{\partial \hat{y}}{\partial w_2} \\ &= -(y-\hat{y})\frac{\partial \sum_i w_i x_i}{\partial w_1} \\ &= -(y-\hat{y})\frac{\partial w_2 x_2}{\partial w_2} \\ &\boxed{= -(y-\hat{y})x_2 = \nabla w_2}\end{aligned}$$

Why do we have partial derivatives now?

Derivative computation

$$\frac{\partial \mathcal{L}}{\partial w_1} = \frac{\partial}{\partial w_1}\left\{\frac{1}{2}(y-\hat{y})^2\right\}$$
$$= -(y-\hat{y})\frac{\partial \hat{y}}{\partial w_1}$$
$$= -(y-\hat{y})\frac{\partial \sum_i w_i x_i}{\partial w_1}$$
$$= -(y-\hat{y})\frac{\partial w_1 x_1}{\partial w_1}$$
$$= -(y-\hat{y})x_1 = \nabla w_1$$

$$\frac{\partial \mathcal{L}}{\partial w_2} = \frac{\partial}{\partial w_2}\left\{\frac{1}{2}(y-\hat{y})^2\right\}$$
$$= -(y-\hat{y})\frac{\partial \hat{y}}{\partial w_2}$$
$$= -(y-\hat{y})\frac{\partial \sum_i w_i x_i}{\partial w_1}$$
$$= -(y-\hat{y})\frac{\partial w_2 x_2}{\partial w_2}$$
$$= -(y-\hat{y})x_2 = \nabla w_2$$

Gradient Update

$$w_1 = w_1 - \eta \nabla w_1$$
$$= w_1 + \eta(y-\hat{y})x_1$$

$$w_2 = w_2 - \eta \nabla w_2$$
$$= w_2 + \eta(y-\hat{y})x_2$$

Gradient Descent

For each sample $\{x_i, y_i\}$

1. Predict

    a. Forward pass $\hat{y} = f_{\text{MLP}}(x_i; \theta)$

    b. Compute Loss $\mathcal{L}_i = \frac{1}{2}(y_i - \hat{y})^2$

2. Update

    two lines now

    a. Back Propagation

    $$\nabla w_{1i} = -(y_i - \hat{y})x_{1i}$$
    $$\nabla w_{2i} = -(y_i - \hat{y})x_{2i}$$

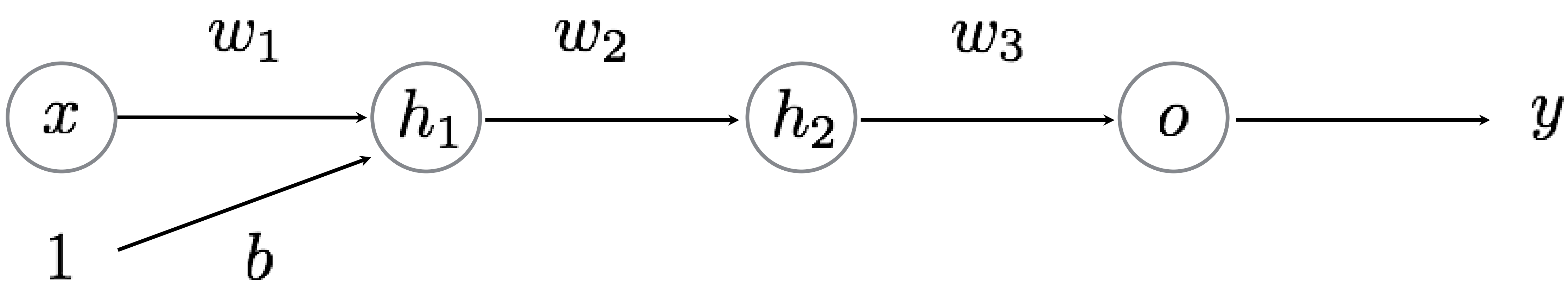
    b. Gradient update

    $$w_{1i} = w_{1i} + \eta(y - \hat{y})x_{1i}$$
    $$w_{2i} = w_{2i} + \eta(y - \hat{y})x_{2i}$$

(adjustable step size)

We haven’t seen a lot of ‘propagation’ yet
because our perceptrons only had <u>one</u> layer…

multi-layer perceptron

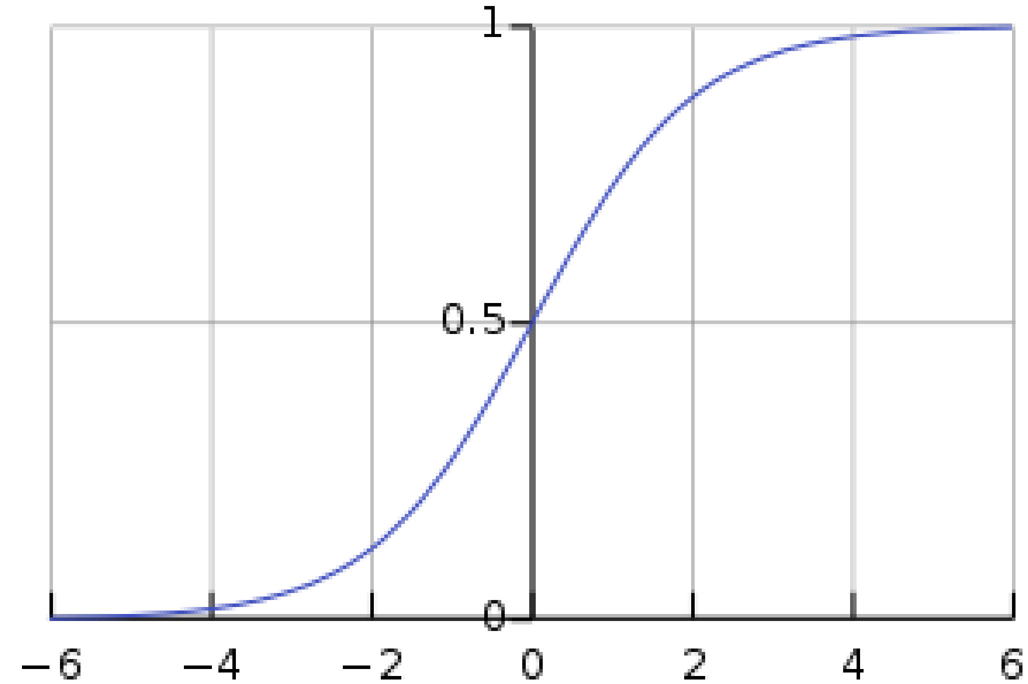


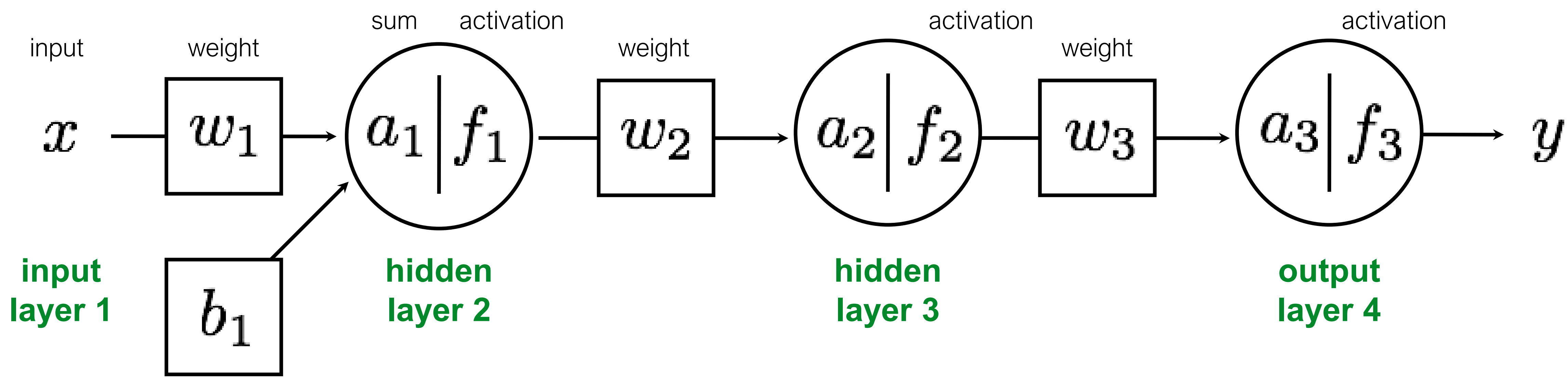
function of **FOUR** parameters and **FOUR** layers!

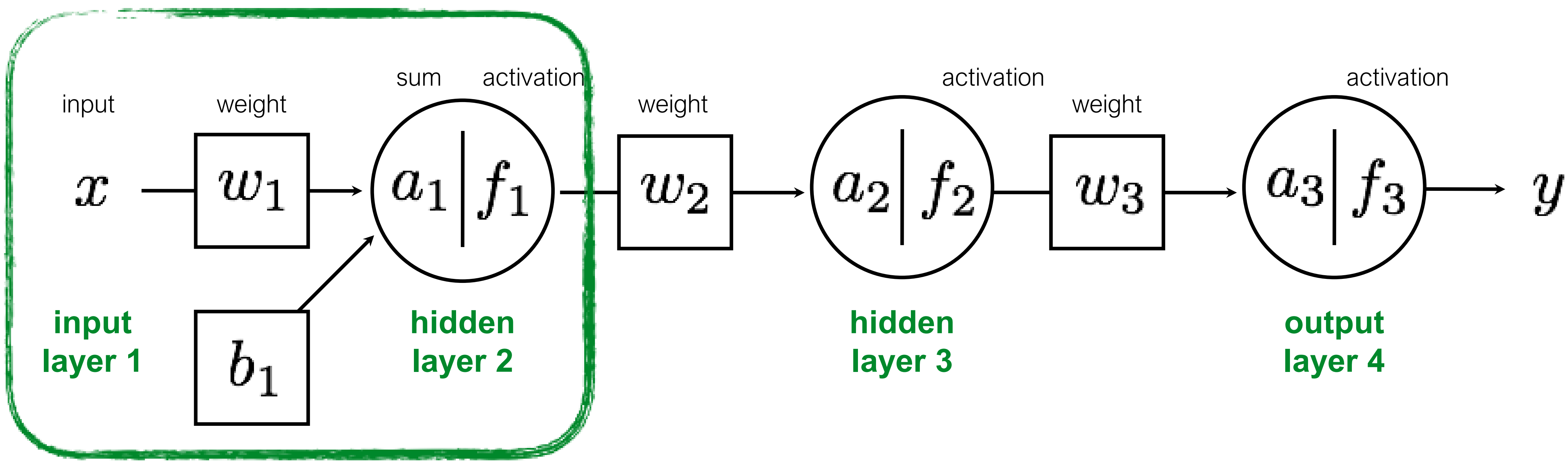
Sigmoid function

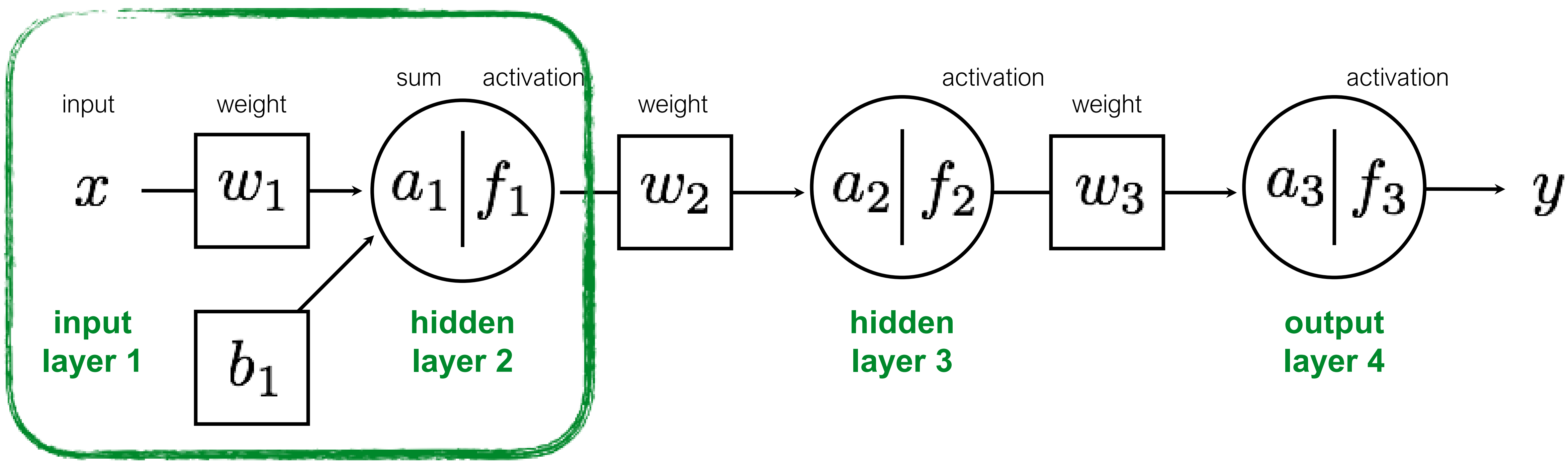
$$S(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1}$$

$$\frac{ds(x)}{dx} = s(x)(1-s(x))$$

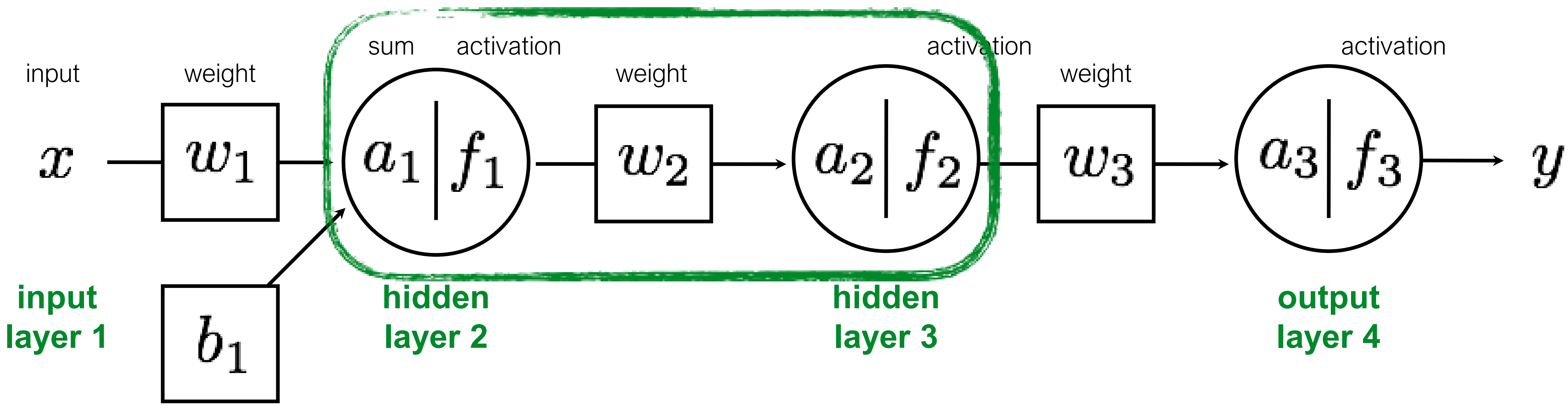



input
weight
sum
activation
weight
activation
weight
activation
x
w_1
$a_1 | f_1$
w_2
$a_2 | f_2$
w_3
$a_3 | f_3$
y
b_1
input
layer 1
hidden
layer 2
hidden
layer 3
output
layer 4

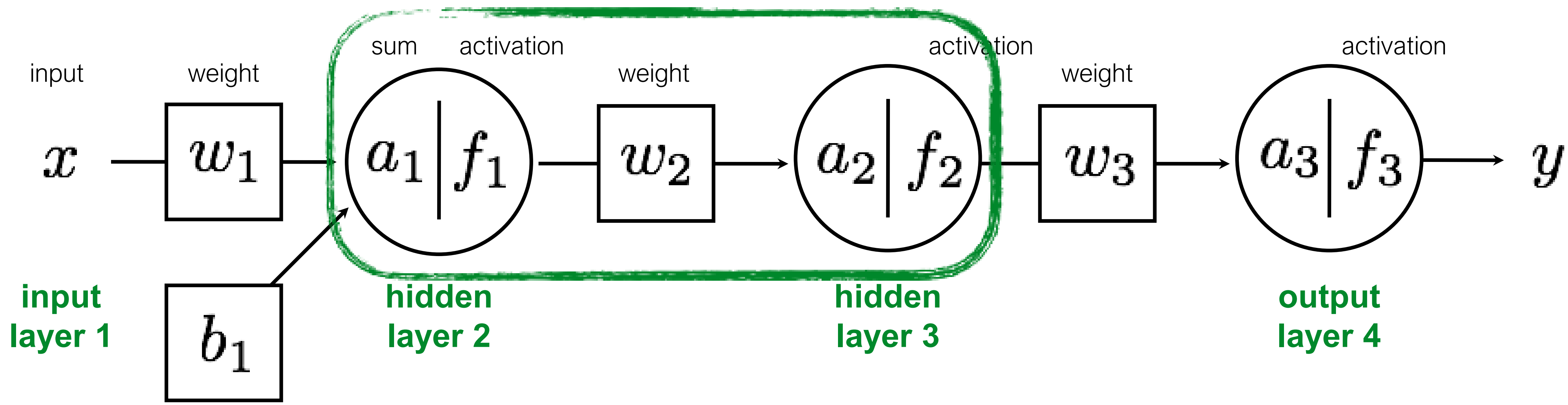

input
weight
sum
activation
weight
activation
weight
activation
x
w_1
b_1
$a_1 | f_1$
w_2
$a_2 | f_2$
w_3
$a_3 | f_3$
y
input
layer 1
hidden
layer 2
hidden
layer 3
output
layer 4



$$a_1 = w_1 \cdot x + b_1$$

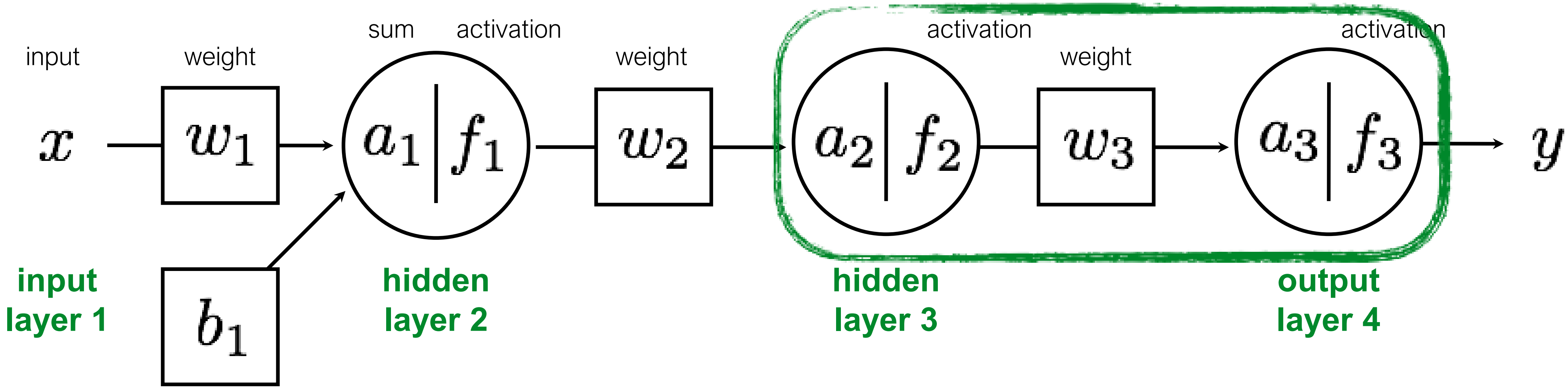


$$a_1 = w_1 \cdot x + b_1$$



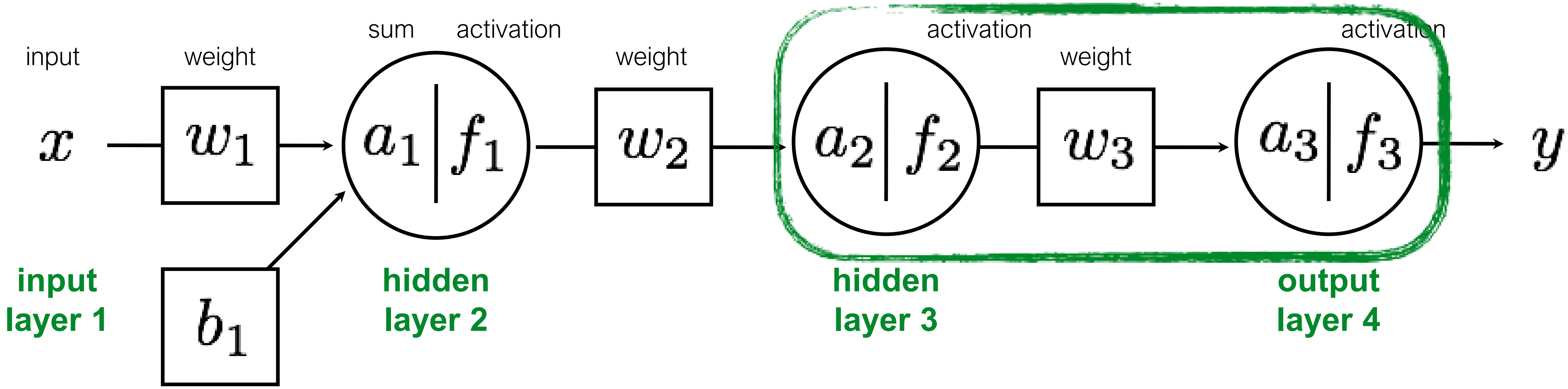
$$a_1 = w_1 \cdot x + b_1$$

$$a_2 = w_2 \cdot f_1(w_1 \cdot x + b_1)$$



$$a_1 = w_1 \cdot x + b_1$$

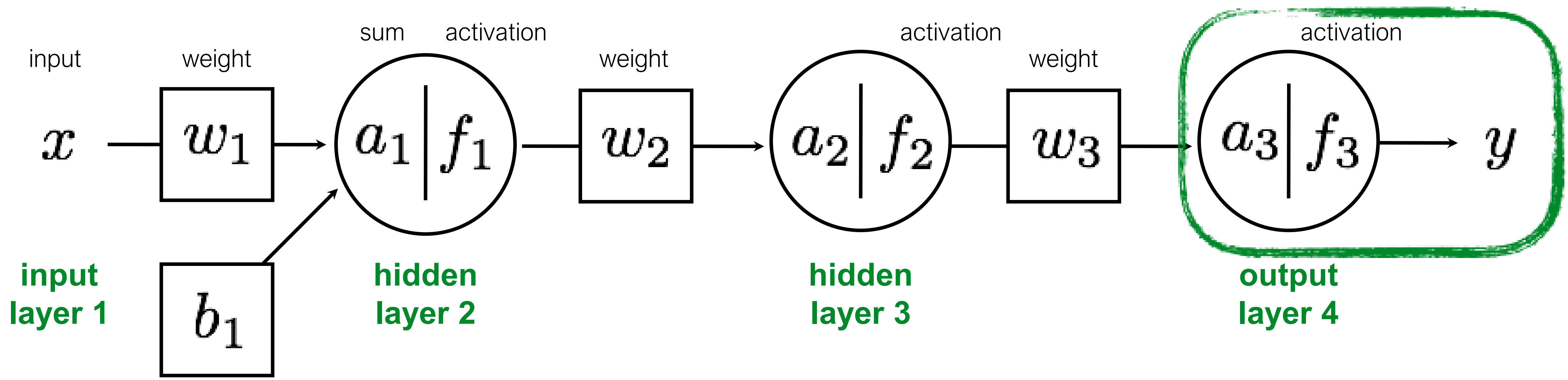
$$a_2 = w_2 \cdot f_1(w_1 \cdot x + b_1)$$



$$a_1 = w_1 \cdot x + b_1$$

$$a_2 = w_2 \cdot f_1(w_1 \cdot x + b_1)$$

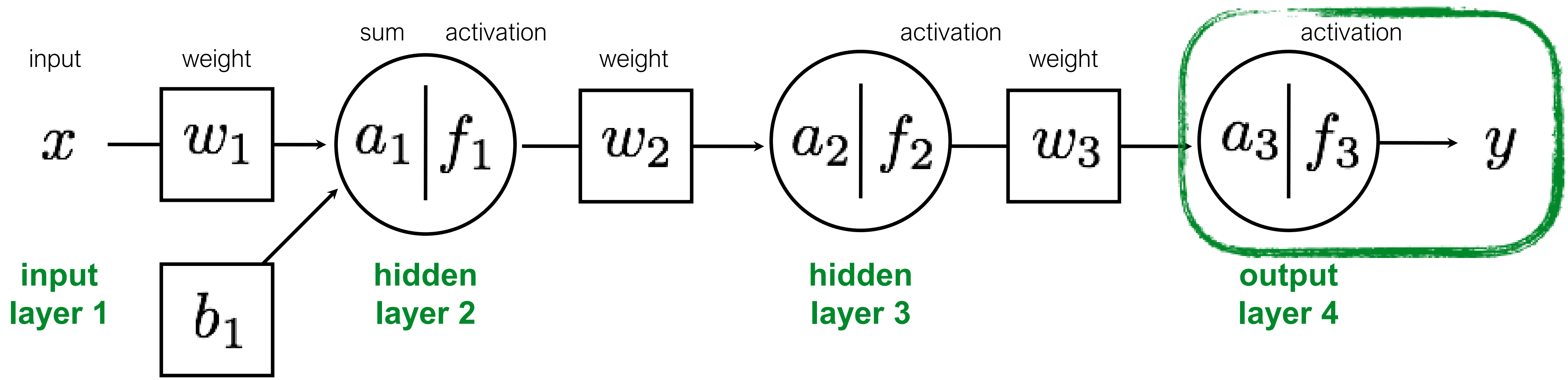
$$a_3 = w_3 \cdot f_2(w_2 \cdot f_1(w_1 \cdot x + b_1))$$



$$a_1 = w_1 \cdot x + b_1$$

$$a_2 = w_2 \cdot f_1(w_1 \cdot x + b_1)$$

$$a_3 = w_3 \cdot f_2(w_2 \cdot f_1(w_1 \cdot x + b_1))$$



$$a_1 = w_1 \cdot x + b_1$$

$$a_2 = w_2 \cdot f_1(w_1 \cdot x + b_1)$$

$$a_3 = w_3 \cdot f_2(w_2 \cdot f_1(w_1 \cdot x + b_1))$$

$$y = f_3(w_3 \cdot f_2(w_2 \cdot f_1(w_1 \cdot x + b_1)))$$

Entire network can be written out as one long equation

$$y = f_3(w_3 \cdot f_2(w_2 \cdot f_1(w_1 \cdot x + b_1)))$$

We need to train the network:

What is known? What is unknown?

Entire network can be written out as a long equation

$$y = f_3(w_3 \cdot f_2(w_2 \cdot f_1(w_1 \cdot x + b_1)))$$

known

We need to train the network:

What is known? What is unknown?

Entire network can be written out as a long equation

$$y = f_3(w_3 \cdot f_2(w_2 \cdot f_1(w_1 \cdot x + b_1)))$$

unknown

We need to train the network:

What is known? What is unknown?

Learning an MLP

Given a set of samples and a MLP

$$\{x_i, y_i\}$$

$$y = f_{\mathrm{MLP}}(x; \theta)$$

Estimate the parameters of the MLP

$$\theta = \quad \{w, b\}$$

Gradient Descent

For each **random** sample $\{x_i, y_i\}$

1. Predict

   a. Forward pass $\hat{y} = f_{\text{MLP}}(x_i; \theta)$

   b. Compute Loss

2. Update

   a. Back Propagation $\frac{\partial \mathcal{L}}{\partial \theta}$ vector of parameter partial derivatives

   b. Gradient update $\theta \leftarrow \theta - \eta \nabla \theta$

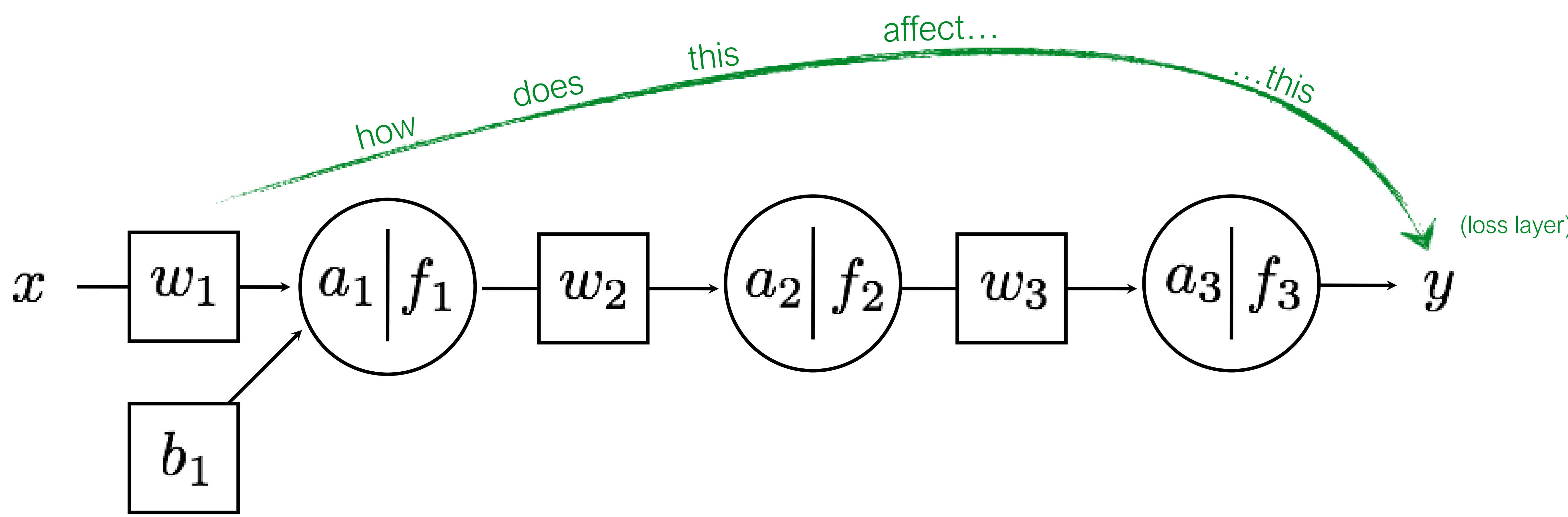
vector of parameter update equations

So we need to compute the partial derivatives

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}} = \left[\frac{\partial \mathcal{L}}{\partial w_3} \frac{\partial \mathcal{L}}{\partial w_2} \frac{\partial \mathcal{L}}{\partial w_1} \frac{\partial \mathcal{L}}{\partial b} \right]$$

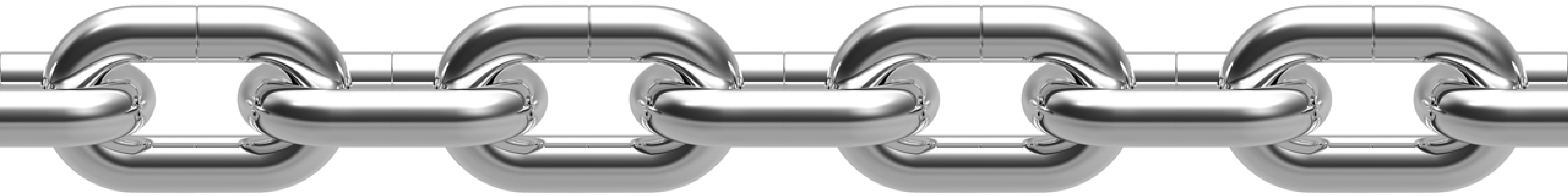
Remember,

Partial derivative $\frac{\partial L}{\partial w_1}$ describes…



So, how do you compute it?

The Chain Rule



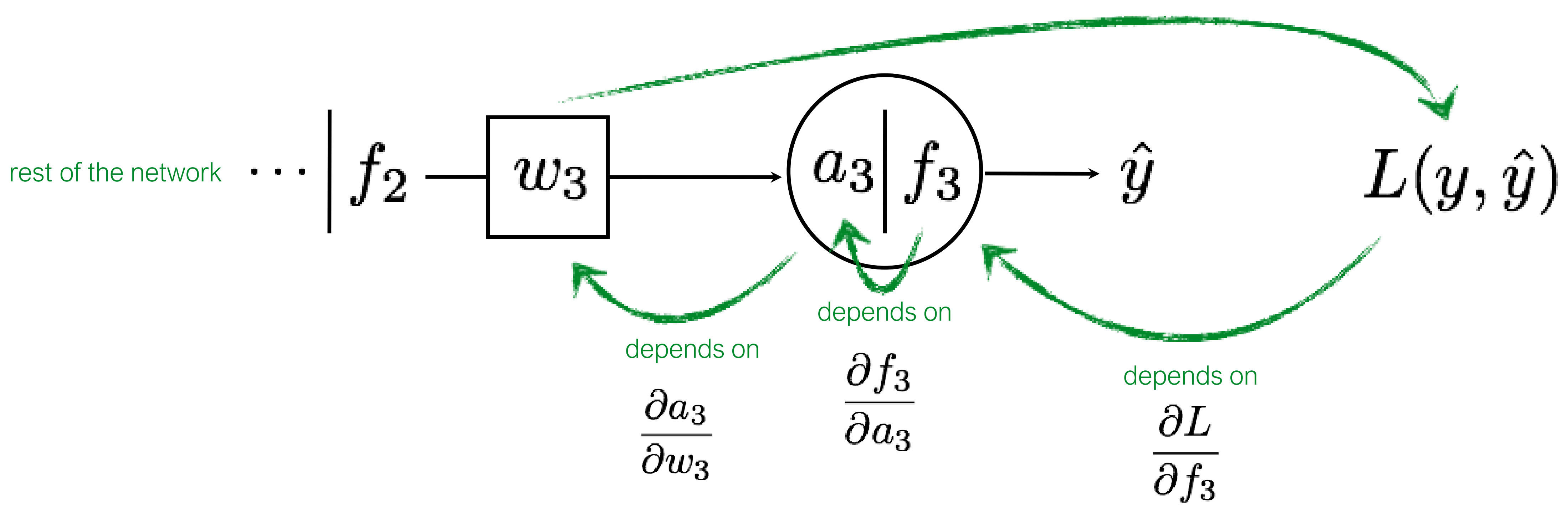
If we have *y=f(u)* and *u=g(x)* then the derivative of *y* w.r.t. *x* is

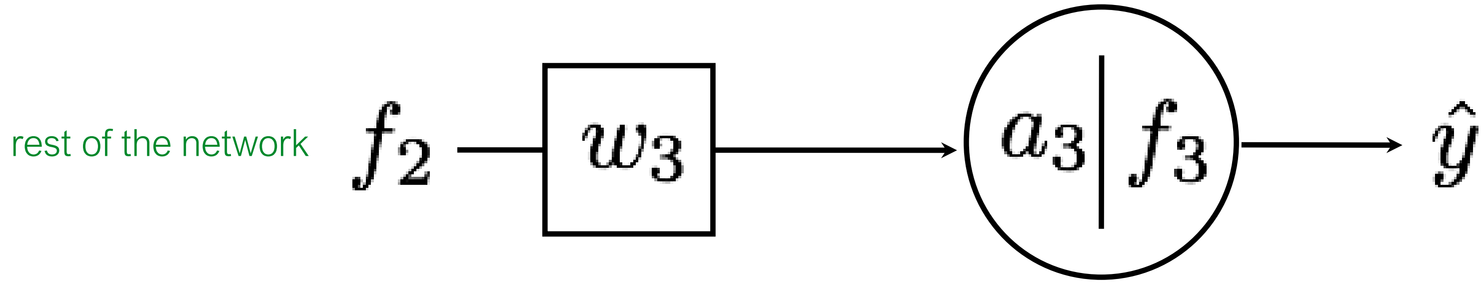
$$\frac{dy}{dx} = \frac{dy}{du}\,\frac{du}{dx}$$

According to the chain rule…

$$\frac{\partial L}{\partial w_3} = \frac{\partial L}{\partial f_3} \frac{\partial f_3}{\partial a_3} \frac{\partial a_3}{\partial w_3}$$

Intuitively, the effect of weight on loss function : $\frac{\partial L}{\partial w_3}$

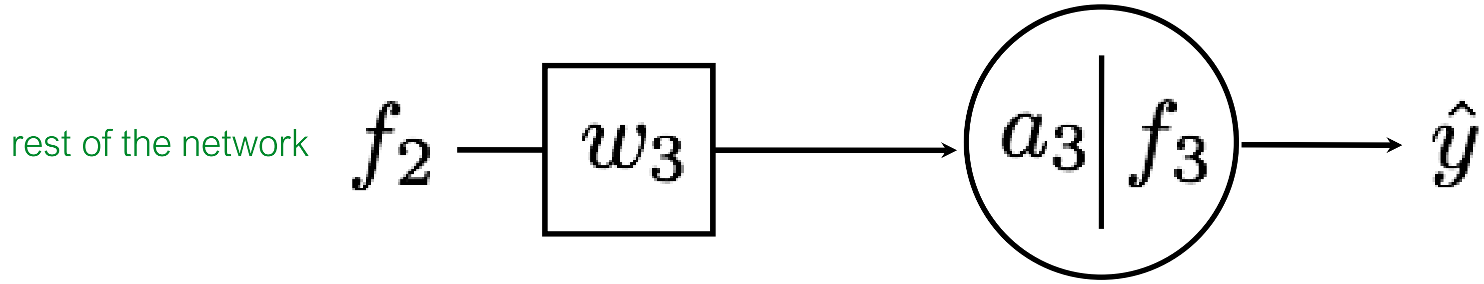




$$L(y, \hat{y})$$

$$\frac{\partial L}{\partial w_3} = \frac{\partial L}{\partial f_3} \frac{\partial f_3}{\partial a_3} \frac{\partial a_3}{\partial w_3}$$

Chain Rule!

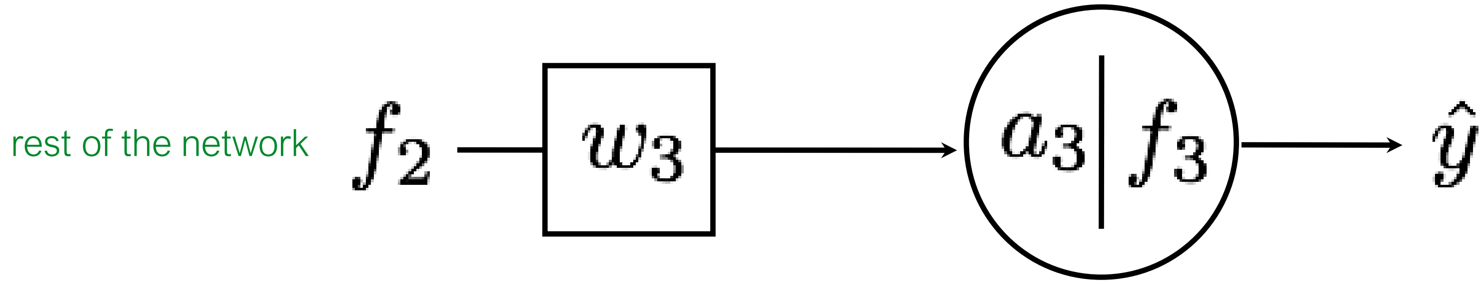


$$L(y, \hat{y})$$

$$\frac{\partial L}{\partial w_3} = \frac{\partial L}{\partial f_3}\frac{\partial f_3}{\partial a_3}\frac{\partial a_3}{\partial w_3}$$

$$= - \quad (y - \hat{y})\frac{\partial f_3}{\partial a_3}\frac{\partial a_3}{\partial w_3}$$

Just the partial
derivative of L2 loss



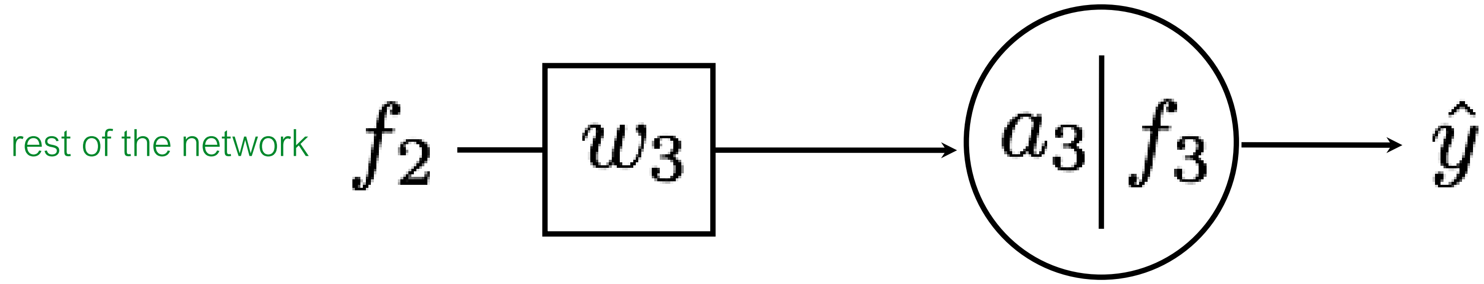
$$L(y, \hat{y})$$

$$\frac{\partial L}{\partial w_3} = \frac{\partial L}{\partial f_3}\frac{\partial f_3}{\partial a_3}\frac{\partial a_3}{\partial w_3}$$

$$= -\ (y - \hat{y})\frac{\partial f_3}{\partial a_3}\frac{\partial a_3}{\partial w_3}$$

Let’s use a Sigmoid function

$$\frac{ds(x)}{dx} = s(x)(1 - s(x))$$



$$L(y, \hat{y})$$

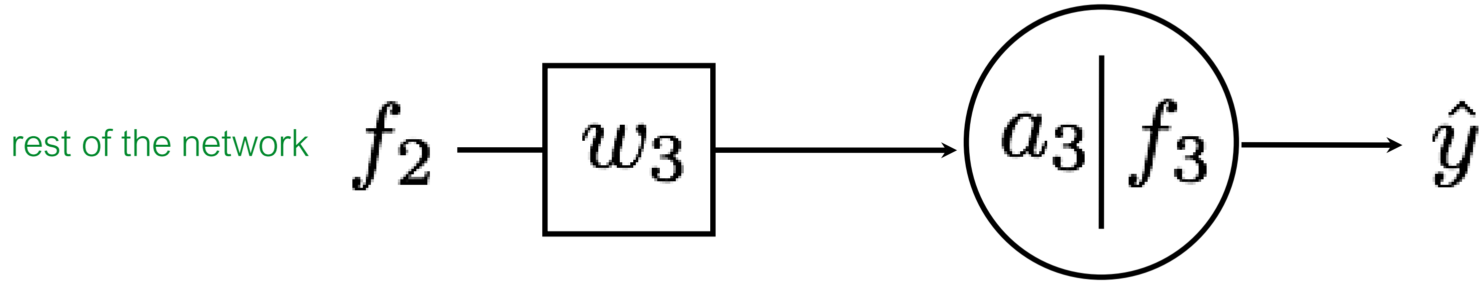
$$\frac{\partial L}{\partial w_3} = \frac{\partial L}{\partial f_3}\frac{\partial f_3}{\partial a_3}\frac{\partial a_3}{\partial w_3}$$

$$= -\ (y - \hat{y})\frac{\partial f_3}{\partial a_3}\frac{\partial a_3}{\partial w_3}$$

$$= -\ (y - \hat{y})f_3(1 - f_3)\frac{\partial a_3}{\partial w_3}$$

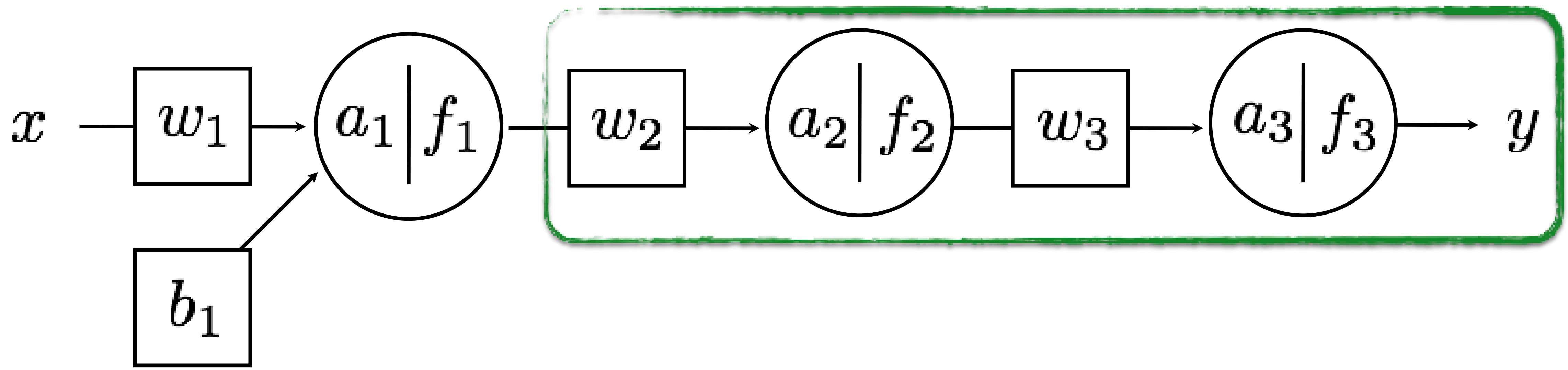
Let's use a Sigmoid function

$$\frac{ds(x)}{dx} = s(x)(1 - s(x))$$

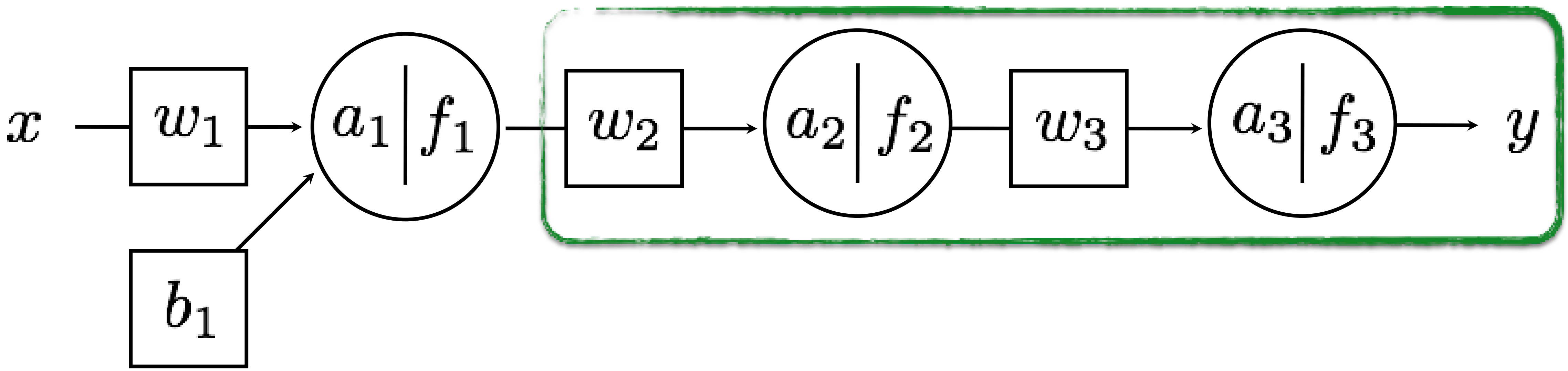


$$L(y, \hat{y})$$

$$\begin{aligned}
\frac{\partial L}{\partial w_3} &= \frac{\partial L}{\partial f_3} \frac{\partial f_3}{\partial a_3} \frac{\partial a_3}{\partial w_3} \\
&= - \; (y - \hat{y}) \frac{\partial f_3}{\partial a_3} \frac{\partial a_3}{\partial w_3} \\
&= - \; (y - \hat{y}) f_3 (1 - f_3) \frac{\partial a_3}{\partial w_3} \\
&= - \; (y - \hat{y}) f_3 (1 - f_3) f_2
\end{aligned}$$



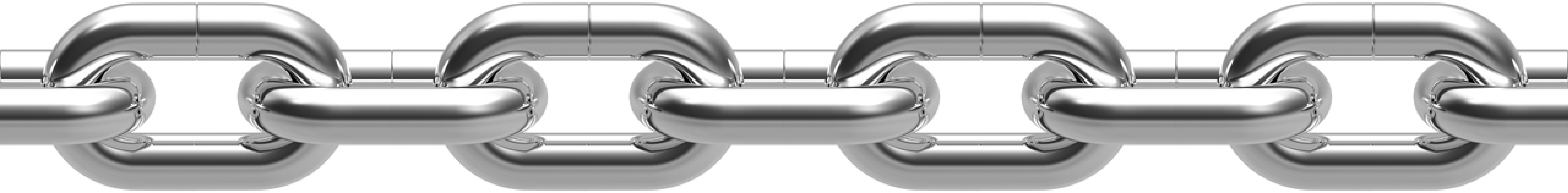
$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial f_3}\frac{\partial f_3}{\partial a_3}\frac{\partial a_3}{\partial f_2}\frac{\partial f_2}{\partial a_2}\frac{\partial a_2}{\partial w_2}$$



$$\frac{\partial L}{\partial w_2} = \boxed{\frac{\partial L}{\partial f_3}\frac{\partial f_3}{\partial a_3}}\frac{\partial a_3}{\partial f_2}\frac{\partial f_2}{\partial a_2}\frac{\partial a_2}{\partial w_2}$$

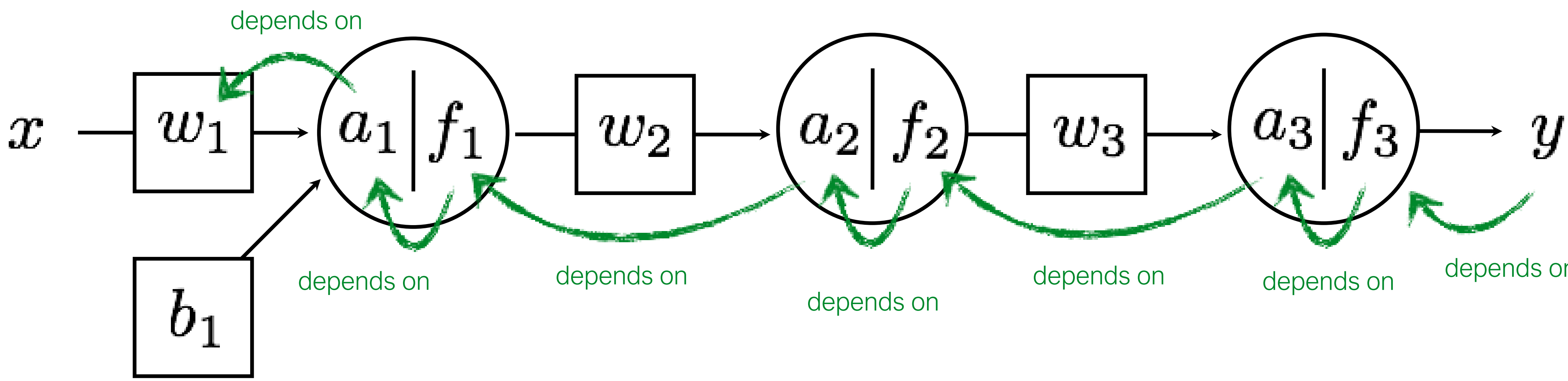
already computed.
re-use (propagate)!

The Chain Rule



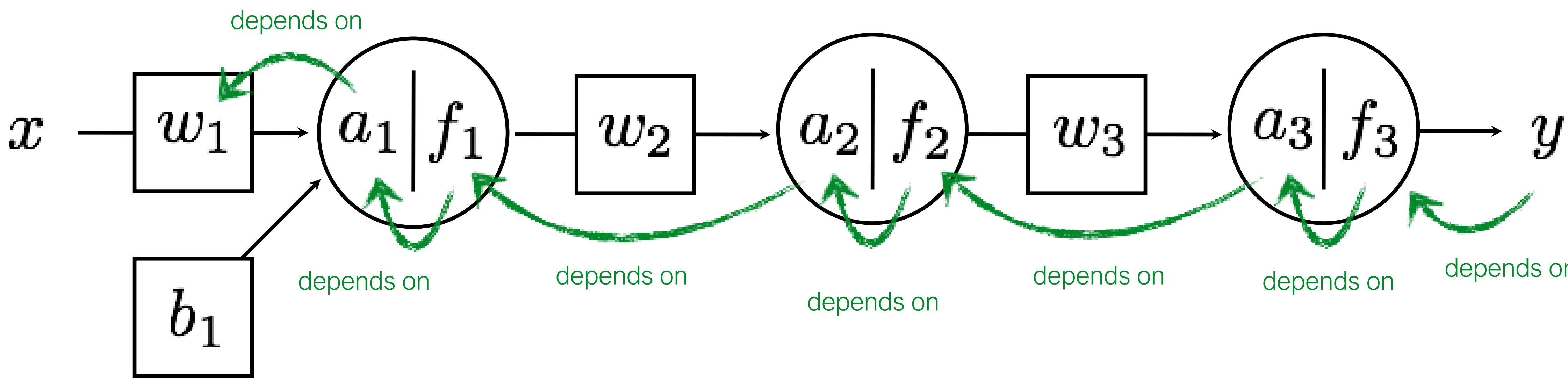
A.K.A. Backpropagation

The chain rule says…



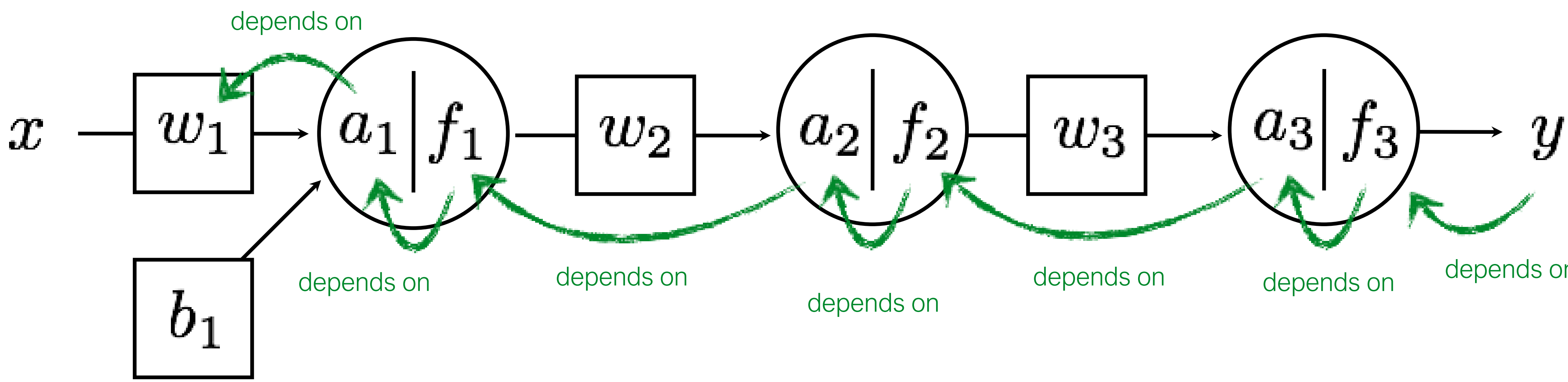
$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial f_3}\frac{\partial f_3}{\partial a_3}\frac{\partial a_3}{\partial f_2}\frac{\partial f_2}{\partial a_2}\frac{\partial a_2}{\partial f_1}\frac{\partial f_1}{\partial a_1}\frac{\partial a_1}{\partial w_1}$$

The chain rule says…



$$\frac{\partial L}{\partial w_1} = \boxed{\frac{\partial L}{\partial f_3}\frac{\partial f_3}{\partial a_3}\frac{\partial a_3}{\partial f_2}\frac{\partial f_2}{\partial a_2}}\frac{\partial a_2}{\partial f_1}\frac{\partial f_1}{\partial a_1}\frac{\partial a_1}{\partial w_1}$$

already computed.
re-use (propagate)!

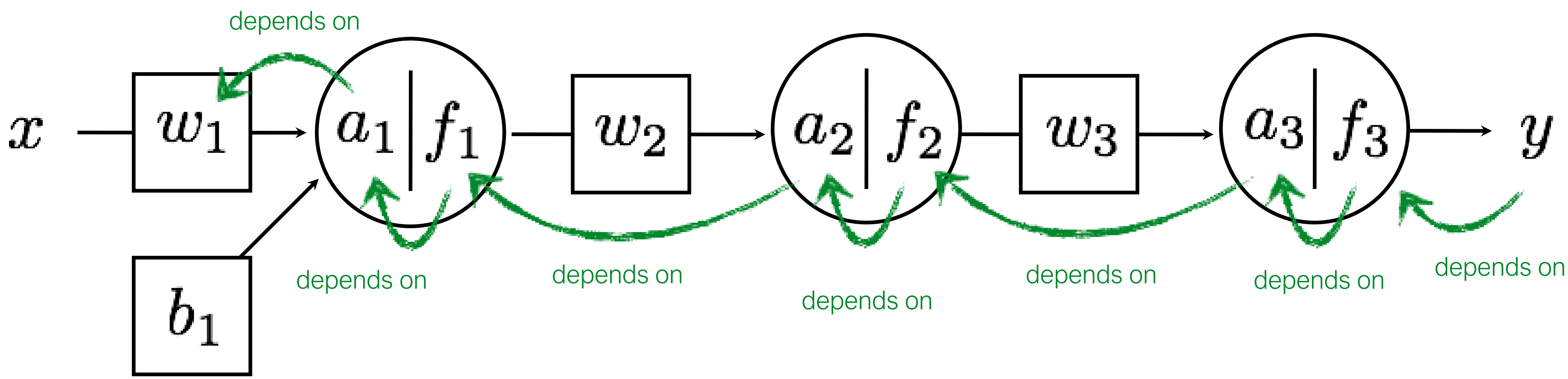


$$\frac{\partial \mathcal{L}}{\partial w_3} = \boxed{\frac{\partial \mathcal{L}}{\partial f_3}\frac{\partial f_3}{\partial a_3}}\frac{\partial a_3}{\partial w_3}$$

$$\frac{\partial \mathcal{L}}{\partial w_2} = \boxed{\frac{\partial \mathcal{L}}{\partial f_3}\frac{\partial f_3}{\partial a_3}}\frac{\partial a_3}{\partial f_2}\frac{\partial f_2}{\partial a_2}\frac{\partial a_2}{\partial w_2}$$

$$\frac{\partial \mathcal{L}}{\partial w_1} = \frac{\partial \mathcal{L}}{\partial f_3}\frac{\partial f_3}{\partial a_3}\frac{\partial a_3}{\partial f_2}\frac{\partial f_2}{\partial a_2}\frac{\partial a_2}{\partial f_1}\frac{\partial f_1}{\partial a_1}\frac{\partial a_1}{\partial w_1}$$

$$\frac{\partial \mathcal{L}}{\partial b} = \frac{\partial \mathcal{L}}{\partial f_3}\frac{\partial f_3}{\partial a_3}\frac{\partial a_3}{\partial f_2}\frac{\partial f_2}{\partial a_2}\frac{\partial a_2}{\partial f_1}\frac{\partial f_1}{\partial a_1}\frac{\partial a_1}{\partial b}$$

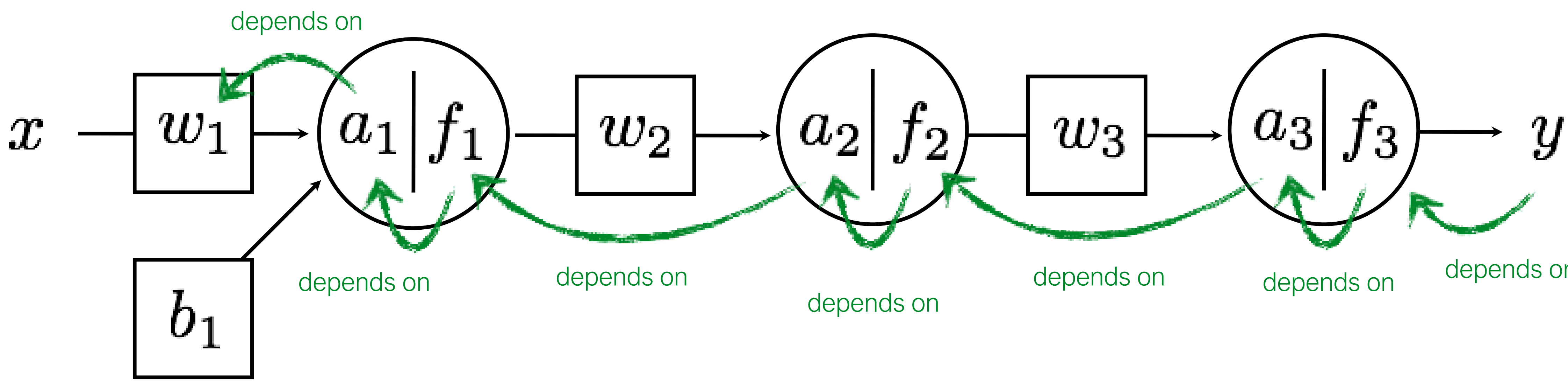


$$\frac{\partial \mathcal{L}}{\partial w_3} = \frac{\partial \mathcal{L}}{\partial f_3}\frac{\partial f_3}{\partial a_3}\frac{\partial a_3}{\partial w_3}$$

$$\frac{\partial \mathcal{L}}{\partial w_2} = \boxed{\frac{\partial \mathcal{L}}{\partial f_3}\frac{\partial f_3}{\partial a_3}\frac{\partial a_3}{\partial f_2}\frac{\partial f_2}{\partial a_2}}\frac{\partial a_2}{\partial w_2}$$

$$\frac{\partial \mathcal{L}}{\partial w_1} = \boxed{\frac{\partial \mathcal{L}}{\partial f_3}\frac{\partial f_3}{\partial a_3}\frac{\partial a_3}{\partial f_2}\frac{\partial f_2}{\partial a_2}}\frac{\partial a_2}{\partial f_1}\frac{\partial f_1}{\partial a_1}\frac{\partial a_1}{\partial w_1}$$

$$\frac{\partial \mathcal{L}}{\partial b} = \frac{\partial \mathcal{L}}{\partial f_3}\frac{\partial f_3}{\partial a_3}\frac{\partial a_3}{\partial f_2}\frac{\partial f_2}{\partial a_2}\frac{\partial a_2}{\partial f_1}\frac{\partial f_1}{\partial a_1}\frac{\partial a_1}{\partial b}$$



$$\frac{\partial \mathcal{L}}{\partial w_3} = \frac{\partial \mathcal{L}}{\partial f_3}\frac{\partial f_3}{\partial a_3}\frac{\partial a_3}{\partial w_3}$$

$$\frac{\partial \mathcal{L}}{\partial w_2} = \frac{\partial \mathcal{L}}{\partial f_3}\frac{\partial f_3}{\partial a_3}\frac{\partial a_3}{\partial f_2}\frac{\partial f_2}{\partial a_2}\frac{\partial a_2}{\partial w_2}$$

$$\frac{\partial \mathcal{L}}{\partial w_1} = \boxed{\frac{\partial \mathcal{L}}{\partial f_3}\frac{\partial f_3}{\partial a_3}\frac{\partial a_3}{\partial f_2}\frac{\partial f_2}{\partial a_2}\frac{\partial a_2}{\partial f_1}\frac{\partial f_1}{\partial a_1}}\frac{\partial a_1}{\partial w_1}$$

$$\frac{\partial \mathcal{L}}{\partial b} = \boxed{\frac{\partial \mathcal{L}}{\partial f_3}\frac{\partial f_3}{\partial a_3}\frac{\partial a_3}{\partial f_2}\frac{\partial f_2}{\partial a_2}\frac{\partial a_2}{\partial f_1}\frac{\partial f_1}{\partial a_1}}\frac{\partial a_1}{\partial b}$$

Gradient Descent

For each example sample $\{x_i, y_i\}$

1. Predict

   a. Forward pass $\hat{y} = f_{\text{MLP}}(x_i; \theta)$

   b. Compute Loss $\mathcal{L}_i$

2. Update

   a. Back Propagation

$$\frac{\partial \mathcal{L}}{\partial w_3} = \frac{\partial \mathcal{L}}{\partial f_3}\frac{\partial f_3}{\partial a_3}\frac{\partial a_3}{\partial w_3}$$
$$\frac{\partial \mathcal{L}}{\partial w_2} = \frac{\partial \mathcal{L}}{\partial f_3}\frac{\partial f_3}{\partial a_3}\frac{\partial a_3}{\partial f_2}\frac{\partial f_2}{\partial a_2}\frac{\partial a_2}{\partial w_2}$$
$$\frac{\partial \mathcal{L}}{\partial w_1} = \frac{\partial \mathcal{L}}{\partial f_3}\frac{\partial f_3}{\partial a_3}\frac{\partial a_3}{\partial f_2}\frac{\partial f_2}{\partial a_2}\frac{\partial a_2}{\partial f_1}\frac{\partial f_1}{\partial a_1}\frac{\partial a_1}{\partial w_1}$$
$$\frac{\partial \mathcal{L}}{\partial b} = \frac{\partial \mathcal{L}}{\partial f_3}\frac{\partial f_3}{\partial a_3}\frac{\partial a_3}{\partial f_2}\frac{\partial f_2}{\partial a_2}\frac{\partial a_2}{\partial f_1}\frac{\partial f_1}{\partial a_1}\frac{\partial a_1}{\partial b}$$

   b. Gradient update

$$w_3 = w_3 - \eta \nabla w_3$$
$$w_2 = w_2 - \eta \nabla w_2$$
$$w_1 = w_1 - \eta \nabla w_1$$
$$b = b - \eta \nabla b$$

Gradient Descent

For each example sample $\{x_i, y_i\}$

1. Predict

   a. Forward pass $\hat{y} = f_{\text{MLP}}(x_i; \theta)$

   b. Compute Loss $\mathcal{L}_i$

2. Update

   a. Back Propagation $\frac{\partial \mathcal{L}}{\partial \theta}$

   vector of parameter partial derivatives

   b. Gradient update $\theta \leftarrow \theta - \eta \frac{\partial \mathcal{L}}{\partial \theta}$

   vector of parameter update equations

Stochastic gradient descent

What we are truly minimizing:

$$\min_{\theta} \sum_{i=1}^{N} L(y_i, f_{MLP}(x_i))$$

The gradient is:

What we are truly minimizing:

$$\min_{\theta} \sum_{i=1}^{N} L(y_i, f_{MLP}(x_i))$$

The gradient is:

$$\sum_{i=1}^{N} \frac{\partial L(y_i, f_{MLP}(x_i))}{\partial \theta}$$

What we use for gradient update is:

What we are truly minimizing:

$$\min_{\theta} \sum_{i=1}^{N} L(y_i, f_{MLP}(x_i))$$

The gradient is:

$$\sum_{i=1}^{N} \frac{\partial L(y_i, f_{MLP}(x_i))}{\partial \theta}$$

What we use for gradient update is:

$$\frac{\partial L(y_i, f_{MLP}(x_i))}{\partial \theta} \quad \text{for some i}$$

How do we select which sample?

How do we select which sample?

- Select randomly!

Do we need to use only one sample?

How do we select which sample?

- Select randomly!

Do we need to use only one sample?

- You can use a *minibatch* of size $B < N$.

Why not do gradient descent with all samples?

How do we select which sample?

- Select randomly!

Do we need to use only one sample?

- You can use a *minibatch* of size $B < N$.

Why not do gradient descent with all samples?

- It’s very expensive when N is large (big data).

Do I lose anything by using stochastic GD?

How do we select which sample?

- Select randomly!

Do we need to use only one sample?

- You can use a *minibatch* of size $B < N$.

Why not do gradient descent with all samples?

- It's very expensive when N is large (big data).

Do I lose anything by using stochastic GD?

- Same convergence guarantees and complexity!
- Better generalization.

Are back-propagation and (stochastic) gradient descent the same thing?

Iteration versus **Epoch**

How many iterations per epoch?