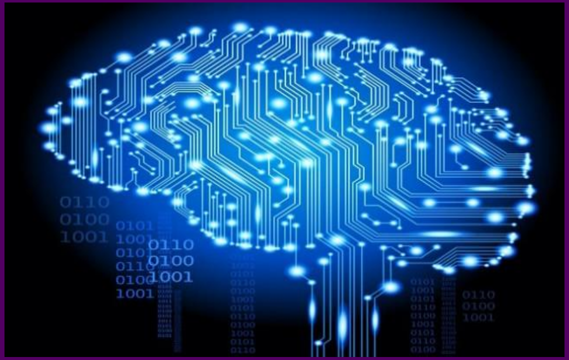
Deep Learning
MSiA 490-30



Theory and Applications

NORTHWESTERN
UNIVERSITY

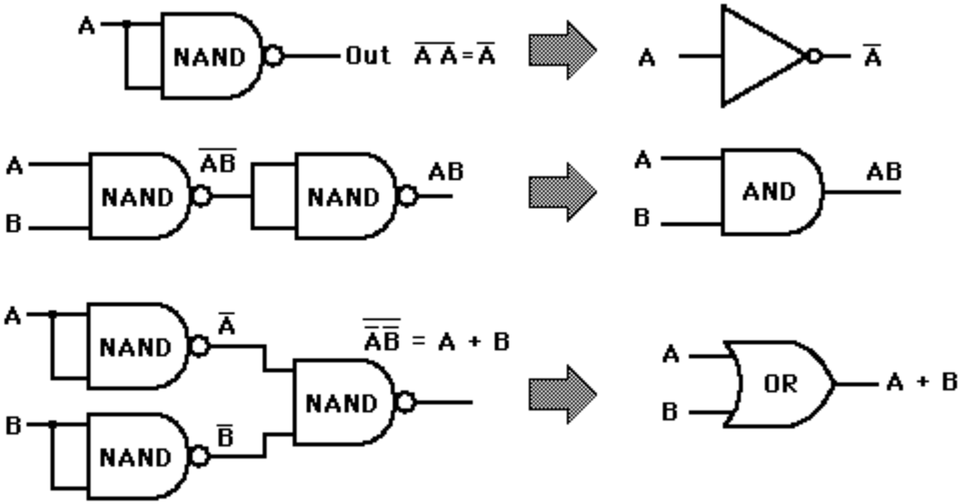
Preliminaries

- Class photo?
- GPU/AWS?
- Heatmaps
- Groups
- Projects

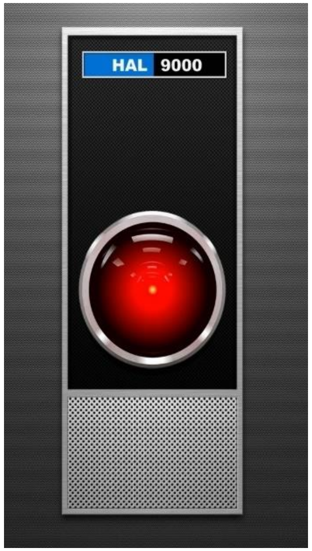
Feedback: News and Ideas

~5-10 min to share ideas/news

NAND gates universal for computation



=



Most modern computers, phones, tablets

http://hyperphysics.phy-astr.gsu.edu/hbase/electronic/nand.html

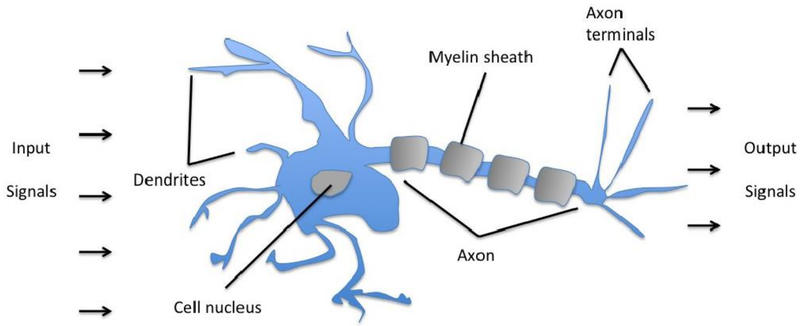
Last time…

- Python tutorial
- Linear classifier identifying 3’s

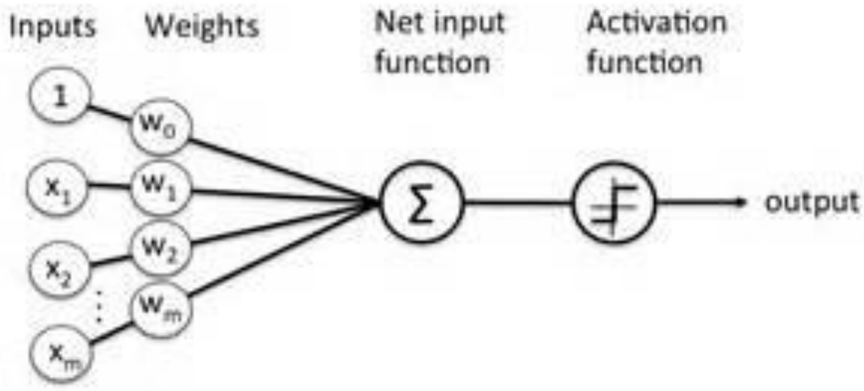
→ This time:

- How to improve classification performance
- NN in “depth”

History: Perceptrons



Schematic of a biological neuron.

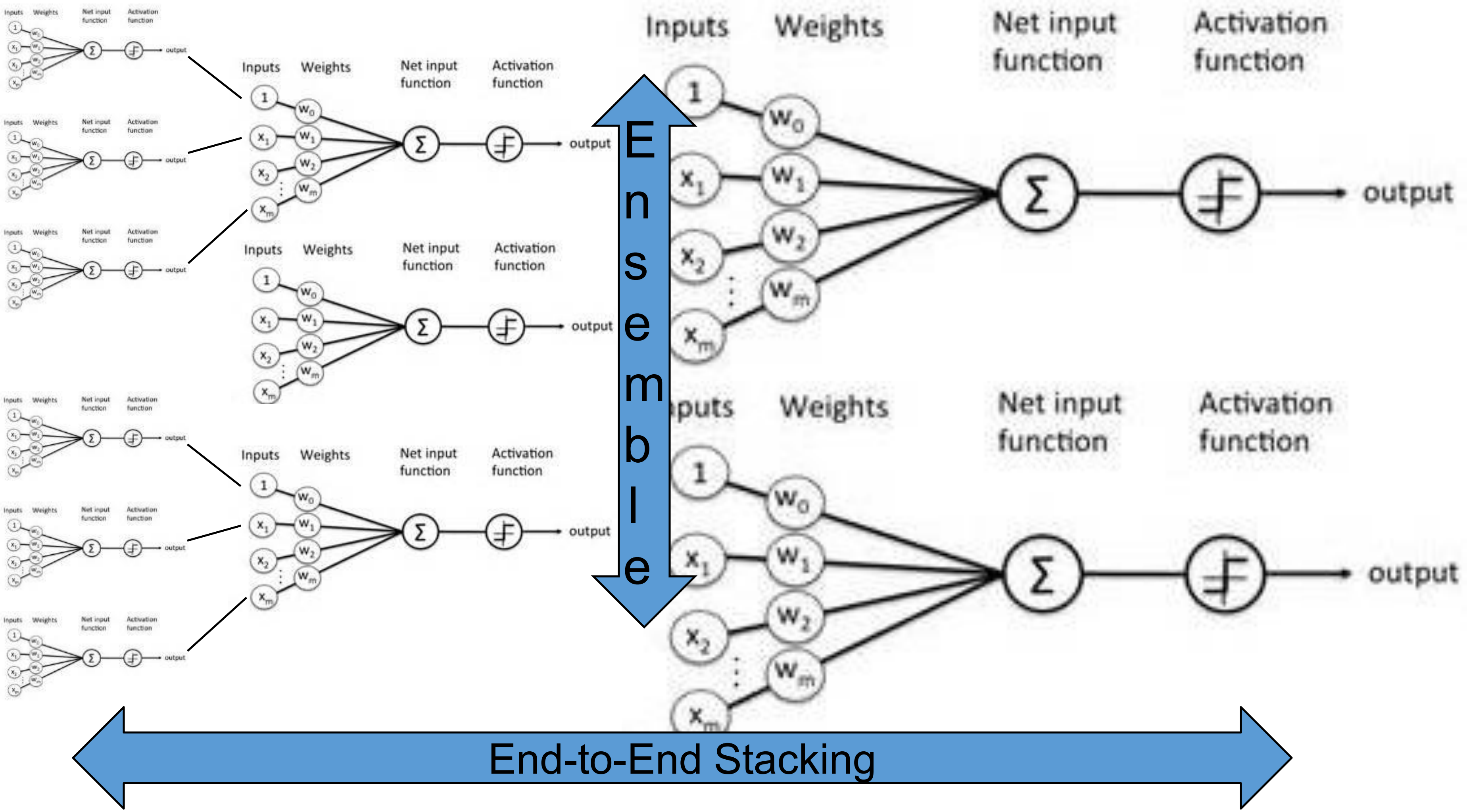


Schematic of Rosenblatt's perceptron.

Rosenblatt, 1957

Think of NN as statistical generalization machines with priors - not as brain models

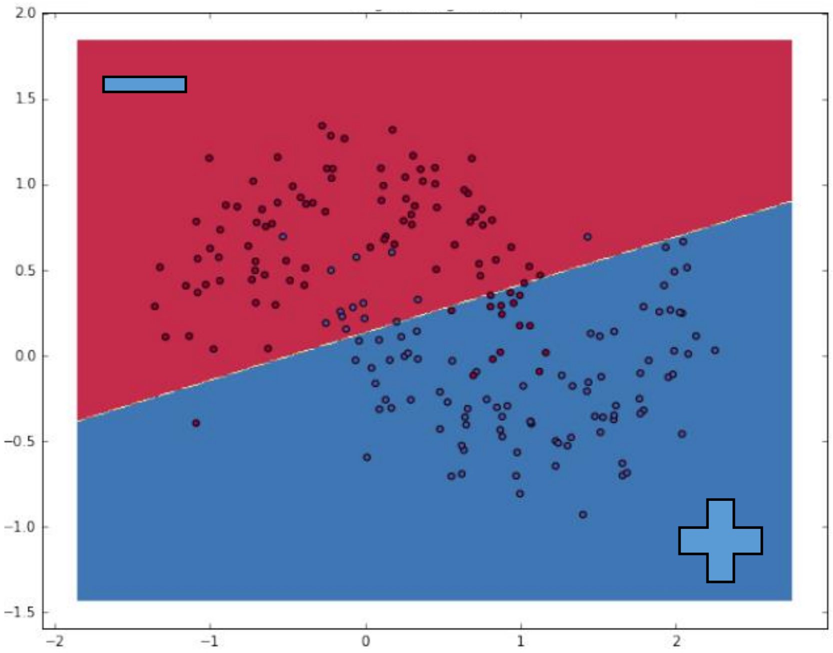
“Layered” Logistic Regression



Recall…

- Simple Linear classifier
- Creates a separating hyperplane defined by:

$$f_w(x) = w_0 + w_1 x_1 + \cdots + w_p x_p$$



Improving predictions

- To “improve” predictions, first define an objective function
- Any ideas?

- Hint: Reward for correct answers, penalize for wrong

Improving predictions

- To "improve" predictions, first define an objective function
- Simple penalty: Mean Square Error (MSE)

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - t_i)^2$$

over n examples

Correct Label (t)	Prediction (y)	Penalty
0	0	0
0	1	1
1	0	1
1	1	0

Improving predictions

- $MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - t_i)^2$
- Recall simple linear classifier:

$$f_w(x) = 1 * w_0 + w_1 x_1 + \cdots + w_p x_p$$

$$= \sum_{j=0}^{p} w_j x_j = w \cdot x$$

- $MSE = \frac{1}{n}\sum_{i=1}^{n}(f_w(x_i) - t_i)^2$
- $MSE = \frac{1}{n}\sum_{i=1}^{n}\left(\sum_{j=0}^{p} w_j x_j - t_i\right)^2$

Any ideas how to minimize MSE?

Improving predictions

- $MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - t_i)^2$
- Recall simple linear classifier:

$$f_w(x) = 1 * w_0 + w_1 x_1 + \cdots + w_p x_p$$

$$= \sum_{j=0}^{p} w_j x_j \;\; = w \cdot x$$

Remember that x_0=1

This is called the "dot product"

- $MSE = \frac{1}{n}\sum_{i=1}^{n}(f_w(x_i) - t_i\,)^2$
- $MSE = \frac{1}{n}\sum_{i=1}^{n}\left(\sum_{j=0}^{p} w_j x_j \; - t_i\right)^2$

Any ideas how to minimize MSE?

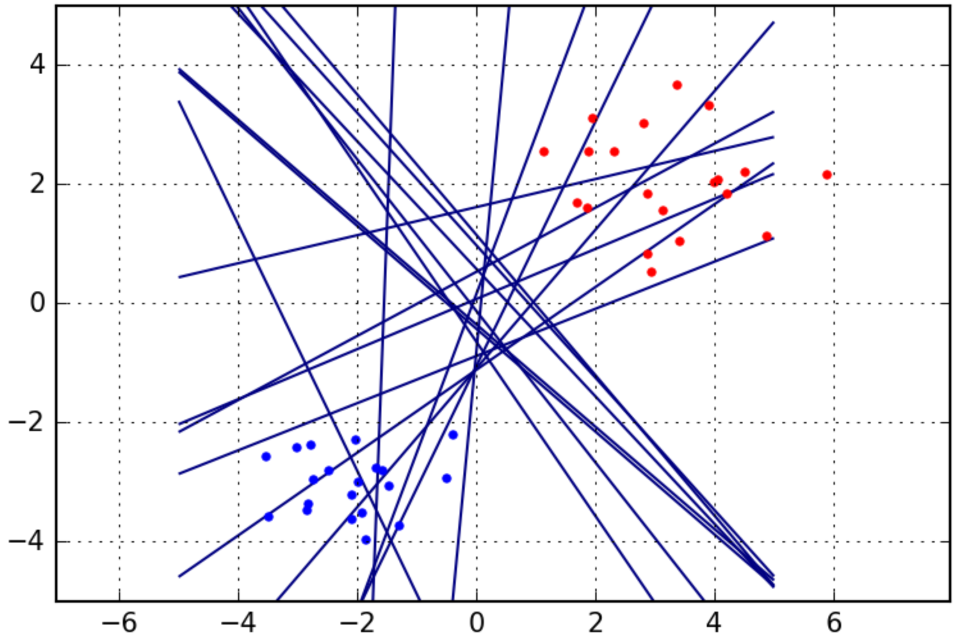
Simple example

- $y = f(w,x)$
  $= w_0 + w_1x_1 + w_2x_2$

x	*t*	y=f(x)
~(3,2)	1	~1
~(-2,-3)	0	~0

- Lots of variance in guessing, few solutions are good

Random guessing again



MSE < 0.25

Simple example

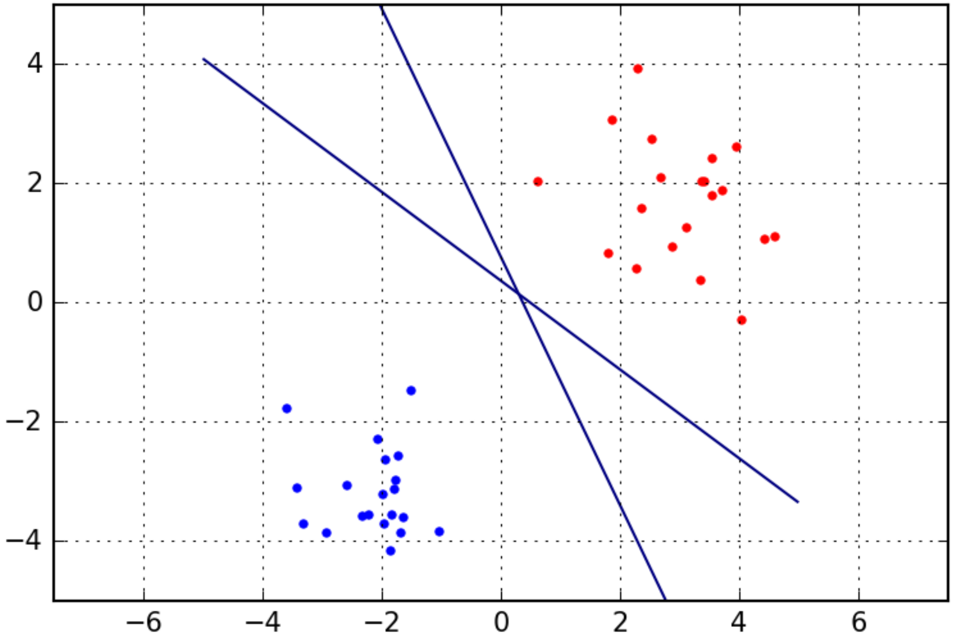
- $y = f(w, x)$
- $= w_0 + w_1 x_1 + w_2 x_2$

x	t	y=f(x)
~(3,2)	1	~1
~(-2,-3)	0	~0

- Doesn't scale well, suppose 10 possible values in each dimension:
  - 100 in 2D
  - 1000 in 3D
  - 1,000,000 in 6D…

w1=-0.60, w2=-0.80, b=0.29
w1=-0.91, w2=-0.44, b=0.34

Random guessing again

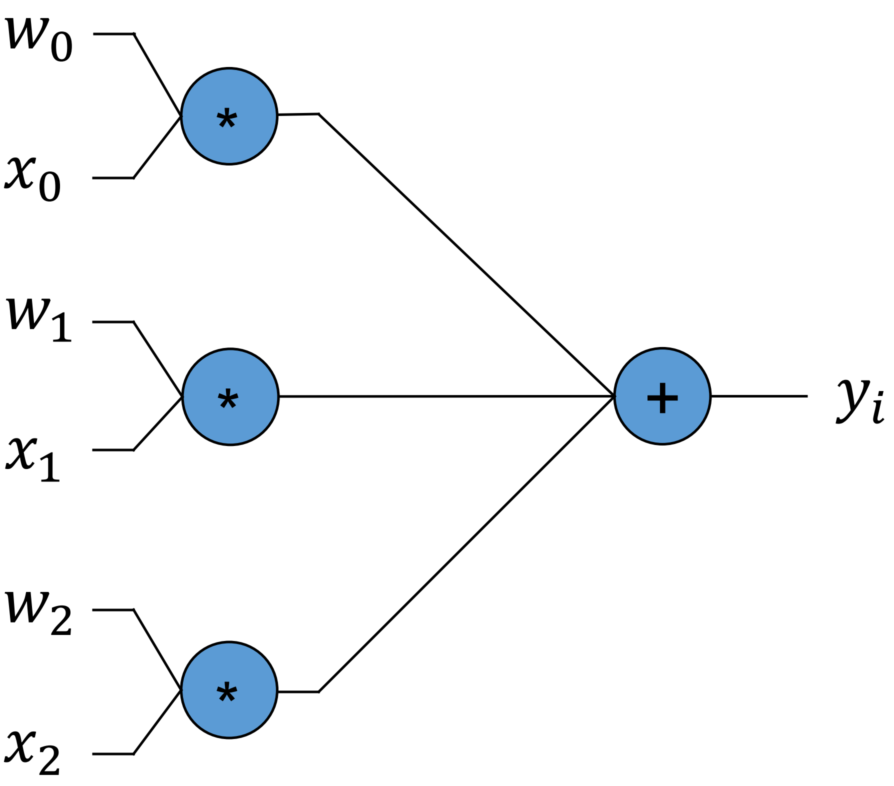


MSE < 0.005

Function as a graph

- $y = f(w, x) = w_0 x_0 + w_1 x_1 + w_2 x_2$

x	*t*
~(3,2)	1
~(-2,-3)	0

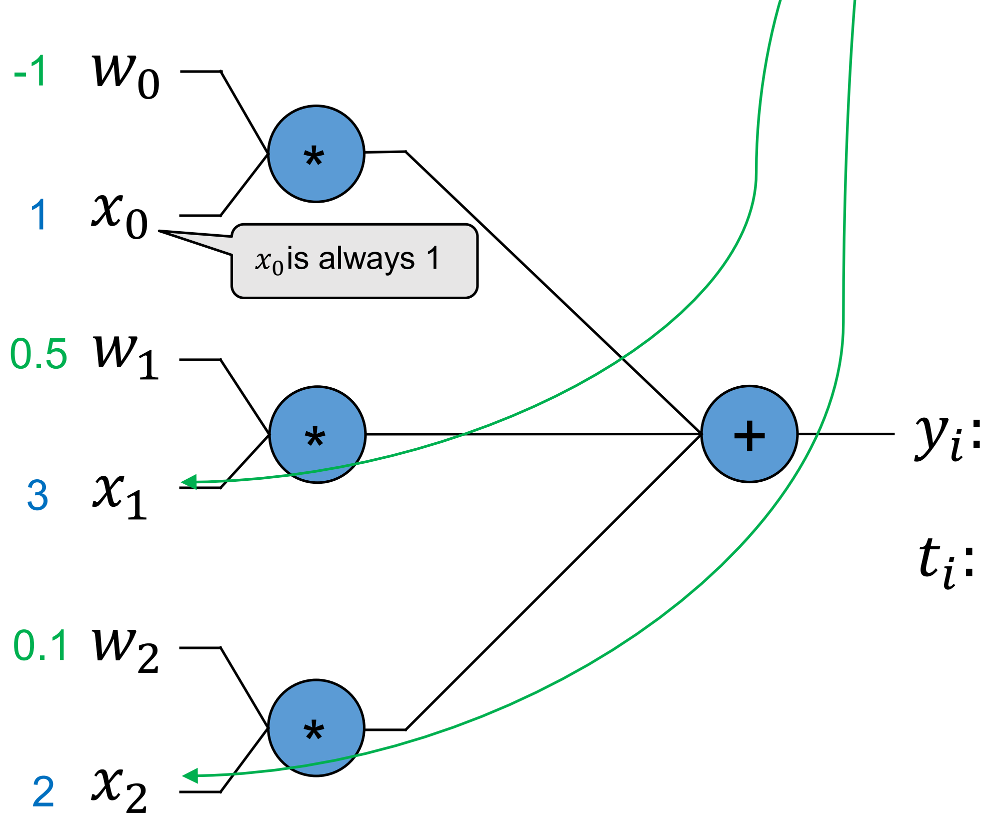


Simple example

- $y = f(w, x) = w_0 x_0 + w_1 x_1 + w_2 x_2$

x	*t*
~(3,2)	1
~(-2,-3)	0

$w = [-1, 0.5, 0.1]$



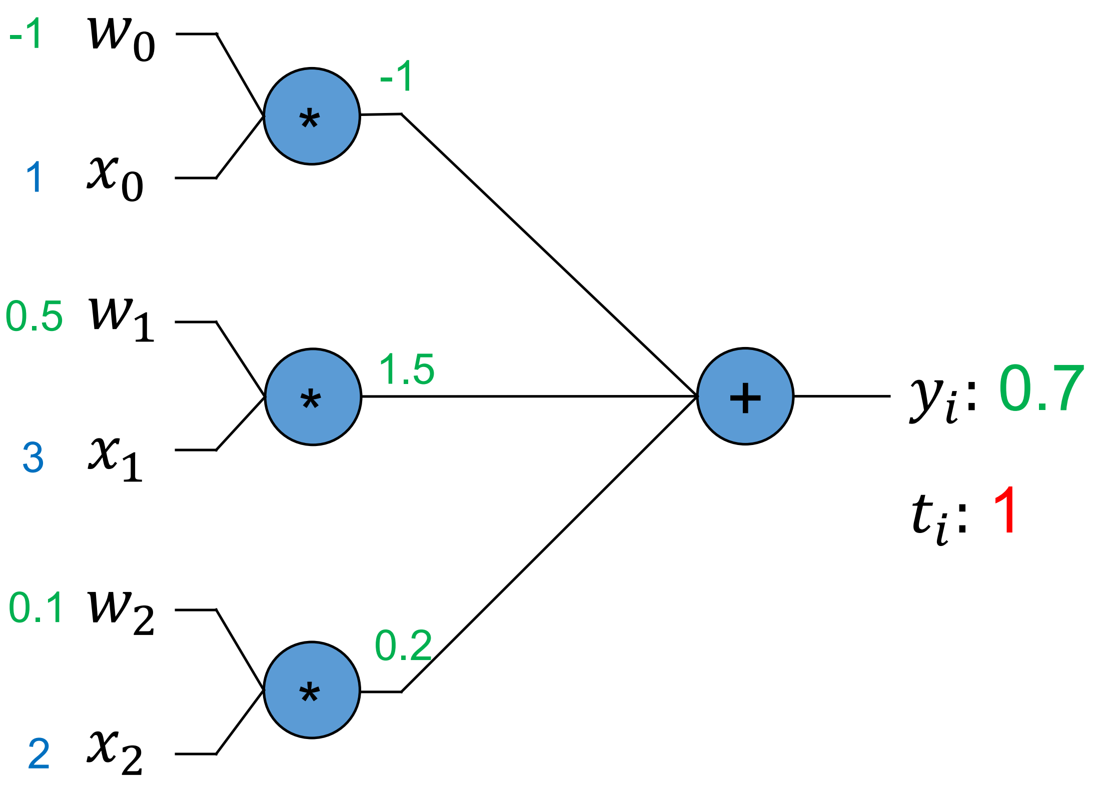
x_0 is always 1

Simple example

- $y = f(w, x) = w_0 x_0 + w_1 x_1 + w_2 x_2$

x	*t*
~(3,2)	1
~(-2,-3)	0

$w = [-1, 0.5, 0.1]$

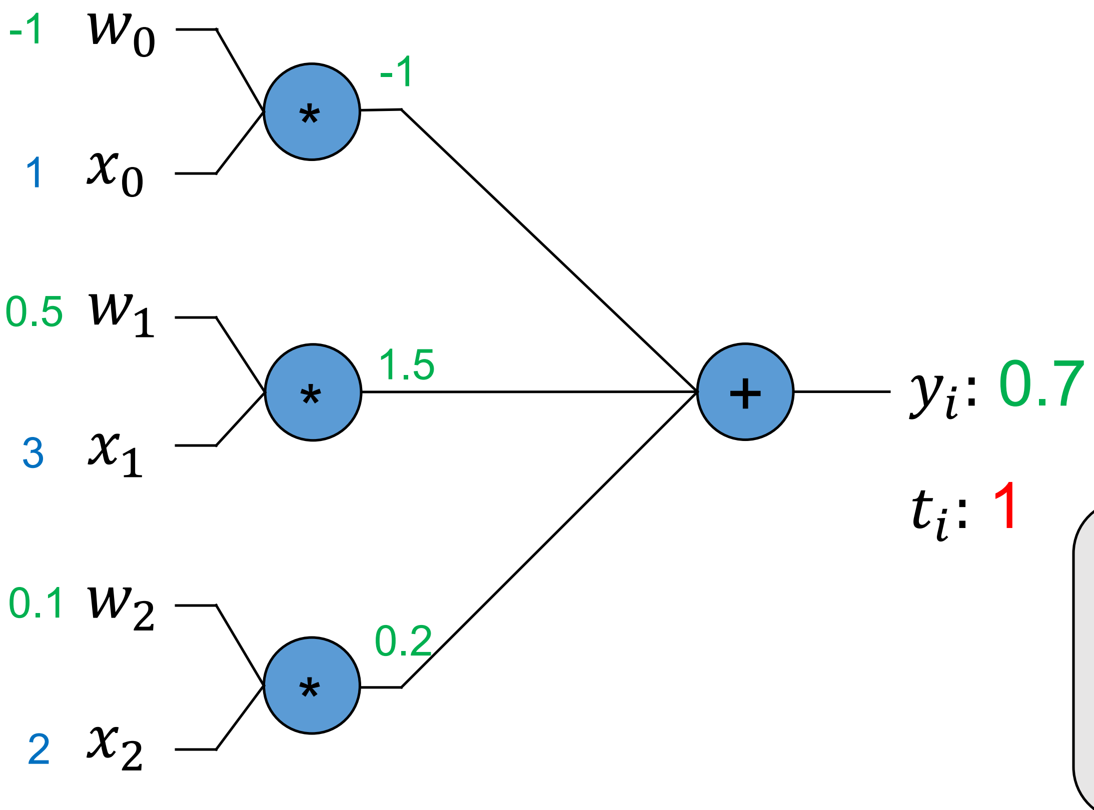


Simple example

- $y = f(w, x) = w_0 x_0 + w_1 x_1 + w_2 x_2$

x	*t*
~(3,2)	1
~(-2,-3)	0

$w = [-1, 0.5, 0.1]$



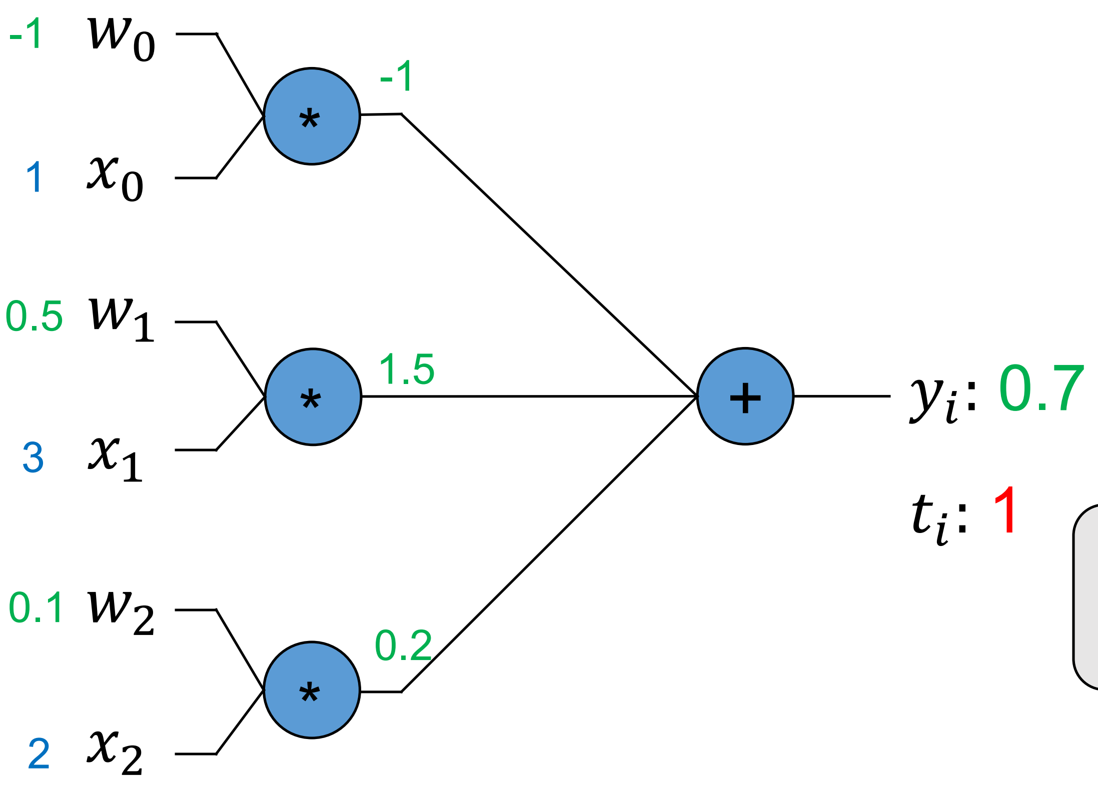
Make y_i look like t_i. More formally, this means minimize $(y_i - t_i)^2$

Simple example

- $y = f(w,x) = w_0x_0 + w_1x_1 + w_2x_2$

x	*t*
~(3,2)	1
~(-2,-3)	0

$w = [-1{,}0.5{,}0.1]$



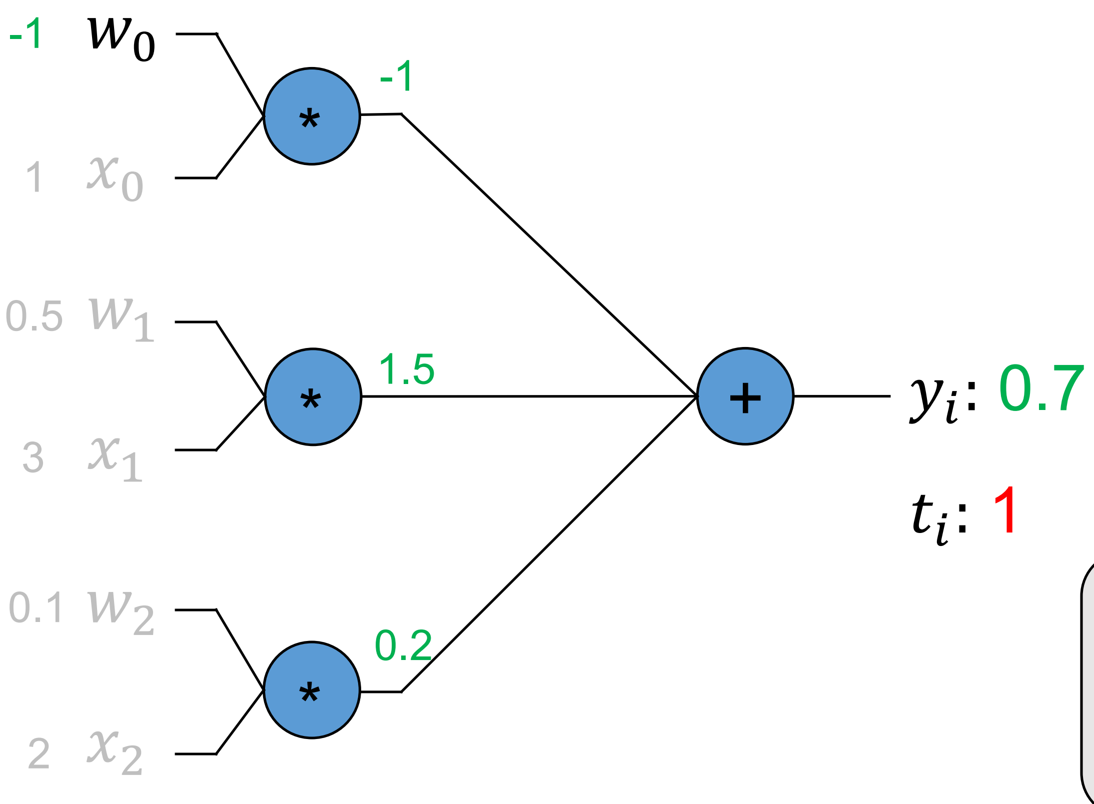
How do we improve our guess?

Simple example

- $y = f(w, x) = w_0 x_0 + w_1 x_1 + w_2 x_2$

x	*t*
~(3,2)	1
~(-2,-3)	0

$w = [-1, 0.5, 0.1]$



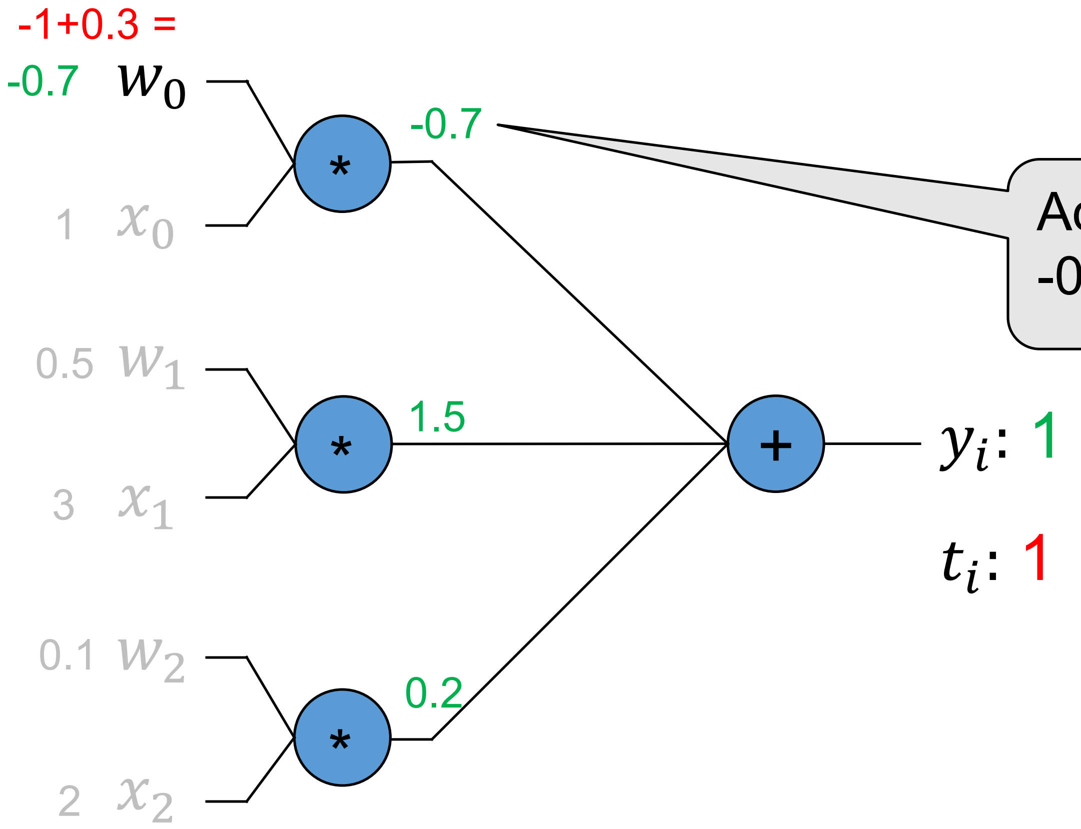
Method 1: Fix everything but one variable and adjust single variable

Simple example

- $y = f(w, x) = w_0x_0 + w_1x_1 + w_2x_2$

x	*t*
~(3,2)	1
~(-2,-3)	0

$w = [-0.7, 0.5, 0.1]$



Add 0.3 to w_0 makes w_0 -0.7 and y_i exactly t_i

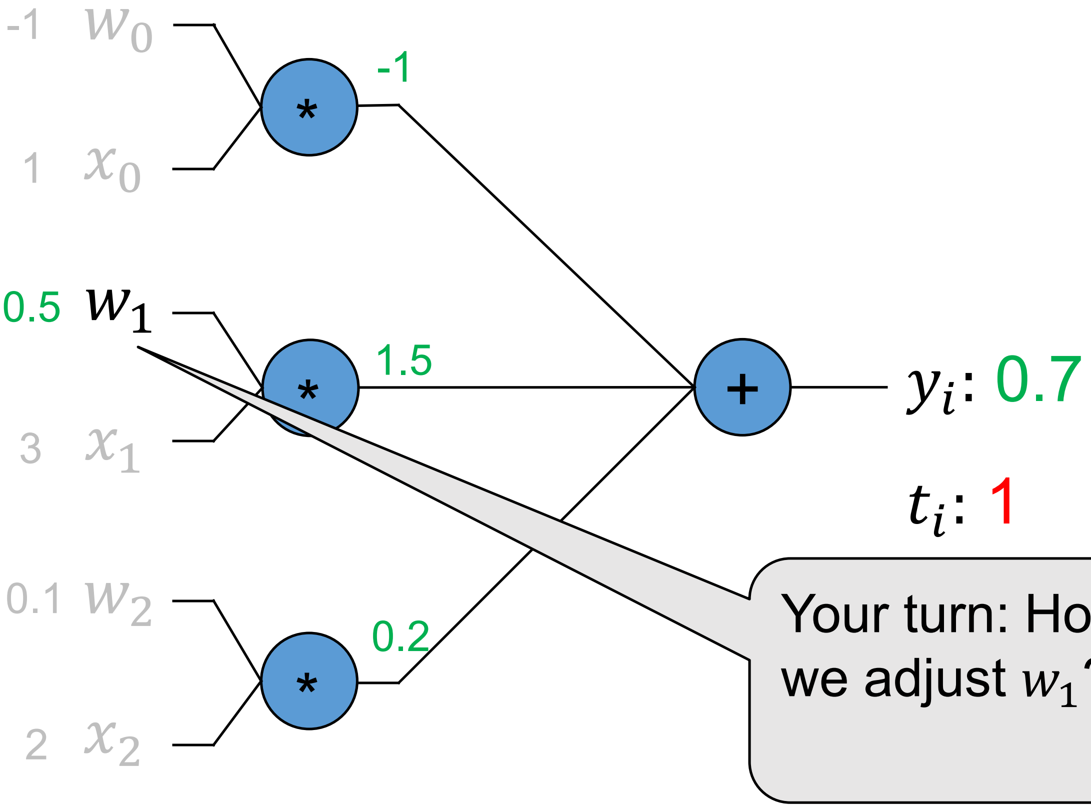
Method 1: Fix everything but one variable and adjust single variable

Simple example

- $y = f(w, x) = w_0 x_0 + w_1 x_1 + w_2 x_2$

x	*t*
~(3,2)	1
~(-2,-3)	0

$w = [-1, 0.5, 0.1]$



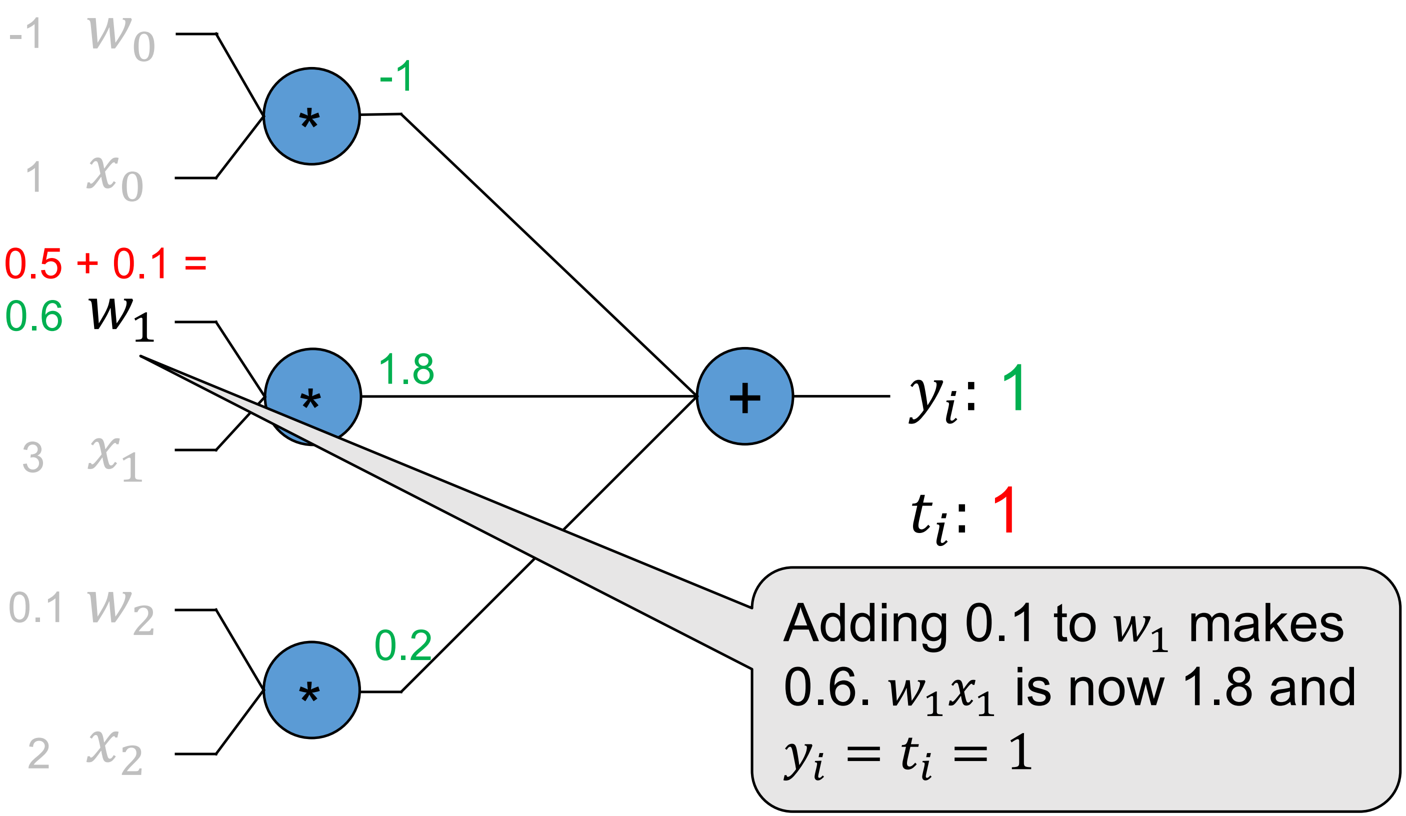
Your turn: How should we adjust w_1?

Simple example

- $y = f(w, x) = w_0 x_0 + w_1 x_1 + w_2 x_2$

x	*t*
~(3,2)	1
~(-2,-3)	0

$w = [-1, 0.6, 0.1]$



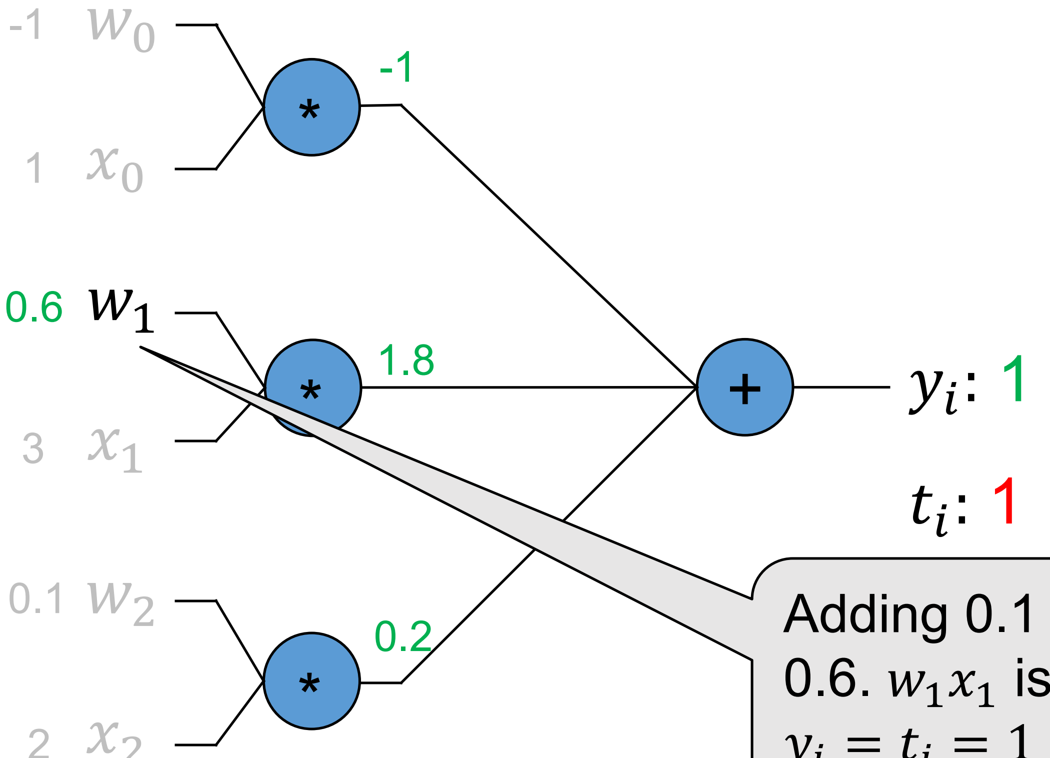
Adding 0.1 to w_1 makes 0.6. $w_1 x_1$ is now 1.8 and $y_i = t_i = 1$

Simple example

- $y = f(w, x) = w_0 x_0 + w_1 x_1 + w_2 x_2$

x	*t*
~(3,2)	1
~(-2,-3)	0

$w = [-1, 0.6, 0.1]$

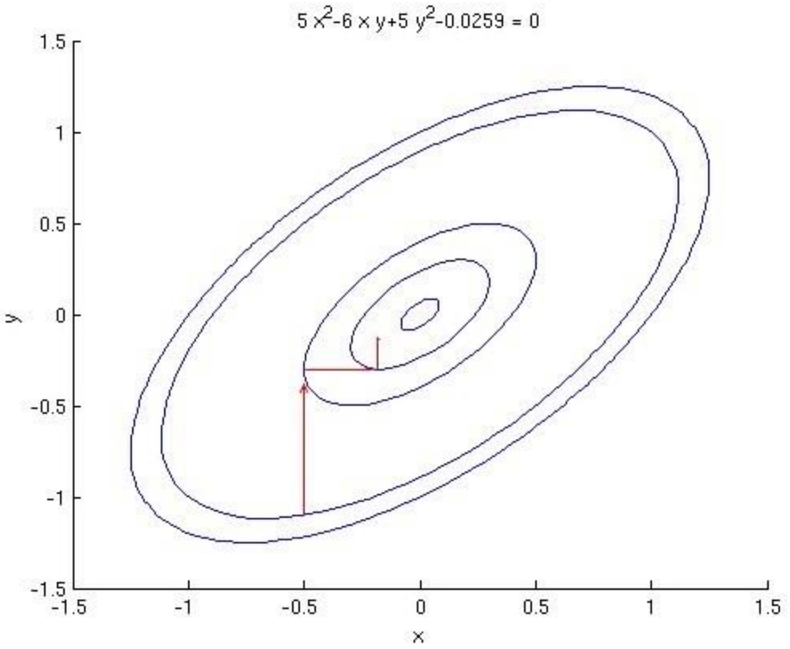


We've basically found an optimization method called *"coordinate descent"*

Adding 0.1 to w_1 makes 0.6. $w_1 x_1$ is now 1.8 and $y_i = t_i = 1$

Coordinate Descent

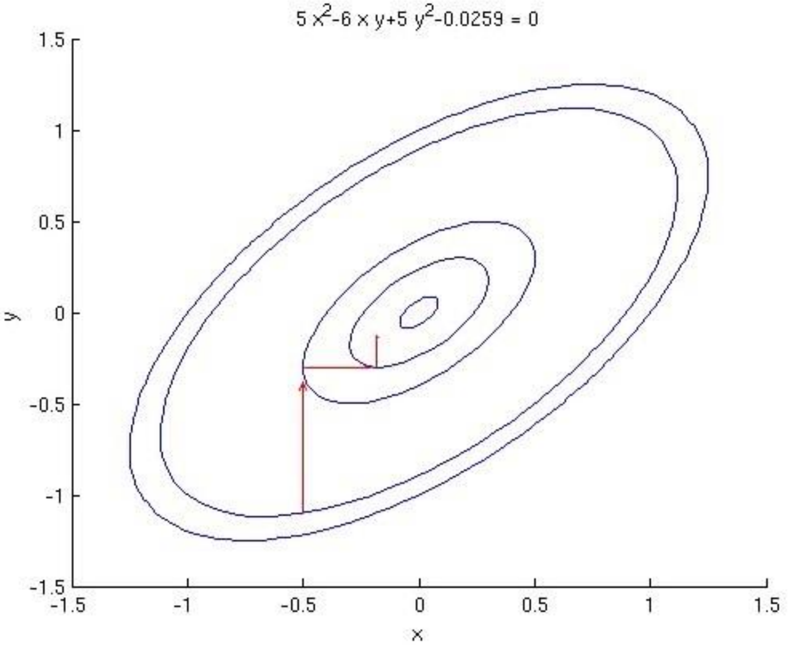
- Repeat until convergence
  - Choose a coordinate (randomly)
  - Step towards minimum along an axis
  - Update solution vector



Wikipedia

But how do we figure out what’s “Best”?

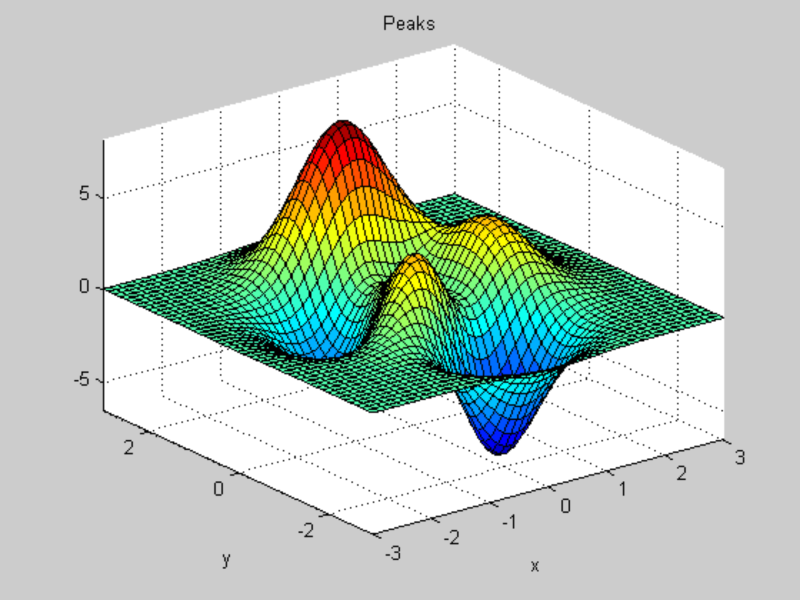
- Pros:
  - Simple
- Cons:
  - Slow (updates only one direction at a time)
  - Stepping in one direction can “undo” gains in another



Issues with Coordinate descent

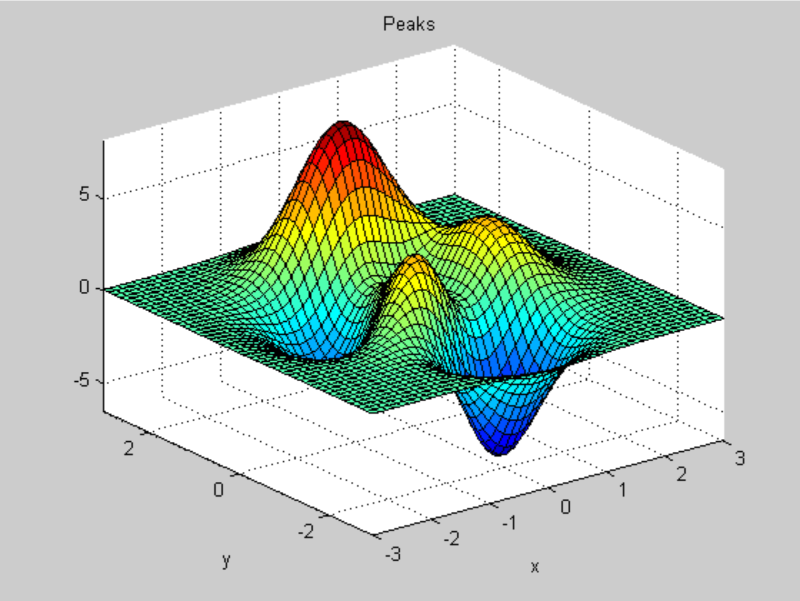
- Updating one dimension at a time is slow
- We might hit a local minima
- Evaluating at lots of points is slow

- Can we do better?
  - Any ideas?



Issues with Coordinate descent

- Updating one dimension at a time is slow
- We might hit a local minima
- Evaluating at lots of points is slow



- Can we do better?
  - Any ideas?

Use calculus to minimize!

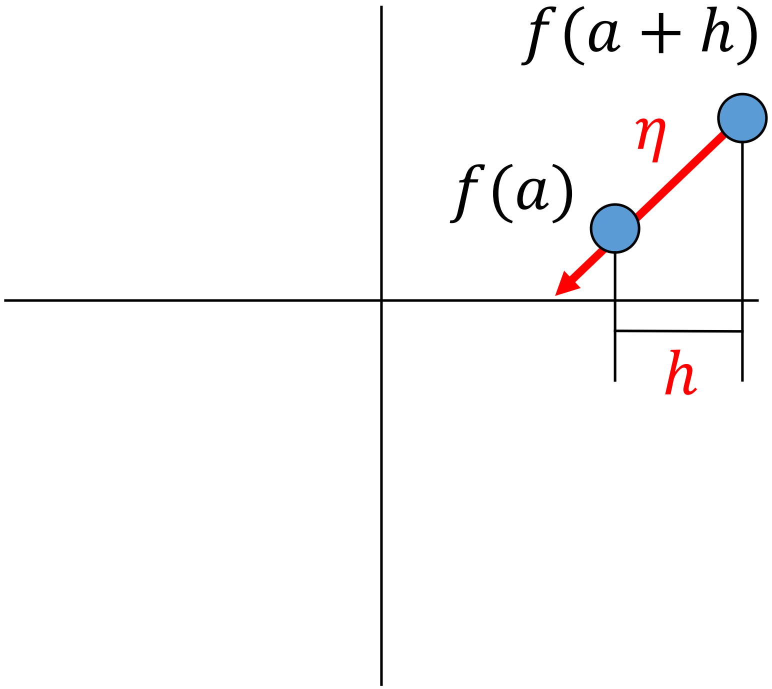
NORTHWESTERN UNIVERSITY

But how do we figure out what’s “Best”?

- Pros:
  - Simple
- Cons:
  - Slow (updates only one direction at a time)
  - Requires lots of evaluation (2x per slope)
  - Stepping in one direction can “undo” gains in another

Improvement:
Take a step η in the direction of steepest descent (slope)

$$slope \approx \lim_{h \to 0} \frac{f(a+h) - f(a)}{h}$$



But how do we figure out what's "Best"?

- Pros:
  - Simple
- Cons:
  - Slow (updates only one direction at a time)
  - Requires lots of evaluation (2x per slope)
  - Stepping in one direction can "undo" gains in another
  - Finite difference unbalanced

(so use centered difference)

Improvement:
Take a step η in the direction of steepest descent (slope)

$$slope \approx \lim_{h \to 0} \frac{f(a+h) - f(a-h)}{2h}$$

Why?

Finite difference error:

O(Δh)

Centered difference error:

O(Δh^2)

http://www.mathematik.uni-dortmund.de/~kuzmin/cfdintro/lecture4.pdf

Quick review of derivatives

- $f(x,y) = xy \rightarrow \boxed{\frac{\partial f}{\partial x} = ?,}\boxed{\frac{\partial f}{\partial y} = ?}$
- $\frac{\partial f(x)}{\partial x} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$
- $f(x+h) = f(x) + h\frac{\partial f(x)}{\partial x}$

Example: x=5, y=-2 → f(x,y)=x*y=?

$$\boxed{\frac{\partial f}{\partial x} = ?} \quad \boxed{\frac{\partial f}{\partial y} = ?} \qquad \boxed{\nabla f = [\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}]}$$

Partial derivatives

Gradient

Quick review of derivatives

- $f(x,y) = xy \;\rightarrow\; \boxed{\frac{\partial f}{\partial x} = y,}\boxed{\frac{\partial f}{\partial y} = x}$
- $\frac{\partial f(x)}{\partial x} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$
- $f(x+h) = f(x) + h\frac{\partial f(x)}{\partial x}$

Example: x=5, y=-2 → f(x,y)=x*y=-10

$$\boxed{\frac{\partial f}{\partial x} = -2} \quad \boxed{\frac{\partial f}{\partial y} = 5} \qquad \boxed{\nabla f = [\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}]}$$

Partial derivatives

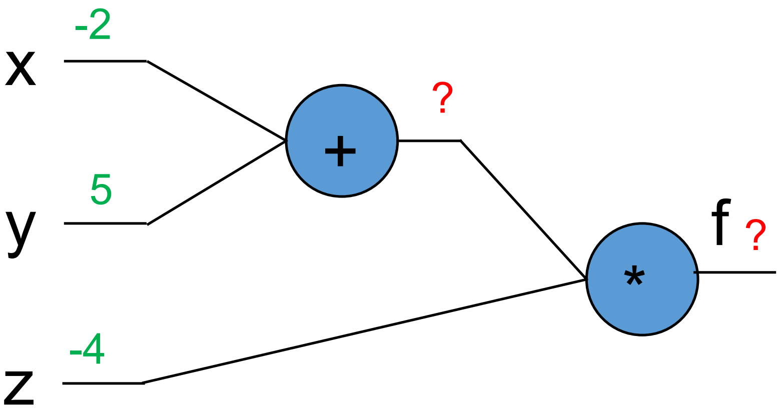
Gradient

Compound expressions

$$f(x, y, z) = (x + y)z$$

$$q = x + y$$

$$f = qz$$



Forward pass:

```
q = x + y → q = ?
f = q * z → f = ?
```

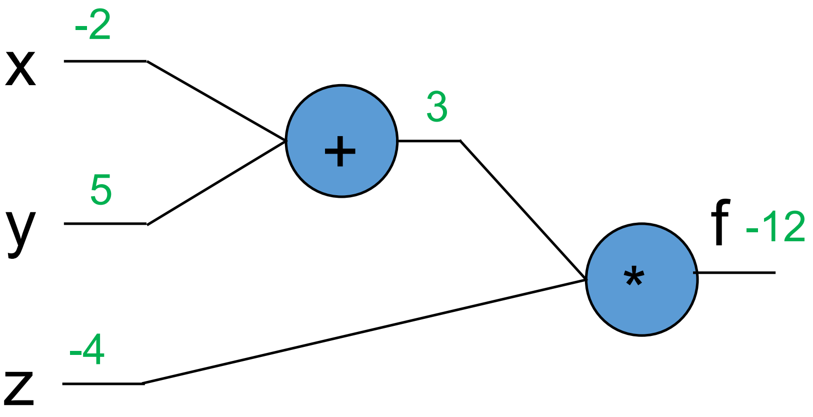
x=-2, y=5, z=-4

Compound expressions

$$f(x,y,z) = (x+y)z$$

$$q = x + y$$

$$f = qz$$



x=-2, y=5, z=-4

Forward pass:

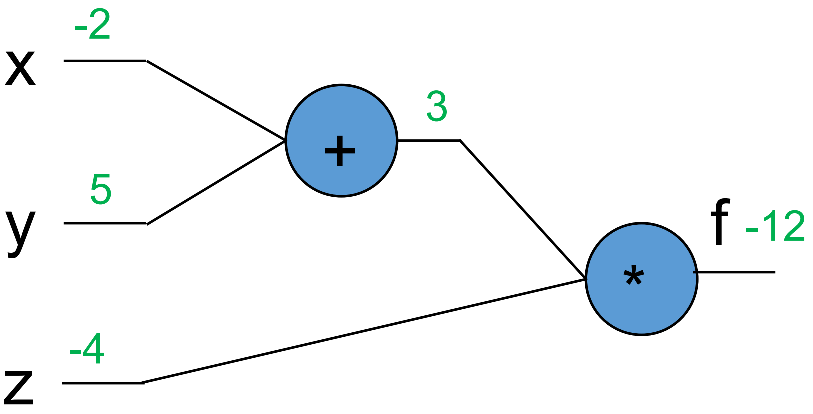
```
q = x + y → q = 3
f = q * z → f = -12
```

Compound expressions

$$f(x,y,z) = (x+y)z = qz$$

$q = x + y \qquad \frac{\partial q}{\partial x} = ?, \frac{\partial q}{\partial y} = ?$

$\frac{\partial f}{\partial q} = ?, \frac{\partial f}{\partial z} = ?$



x=-2, y=5, z=-4

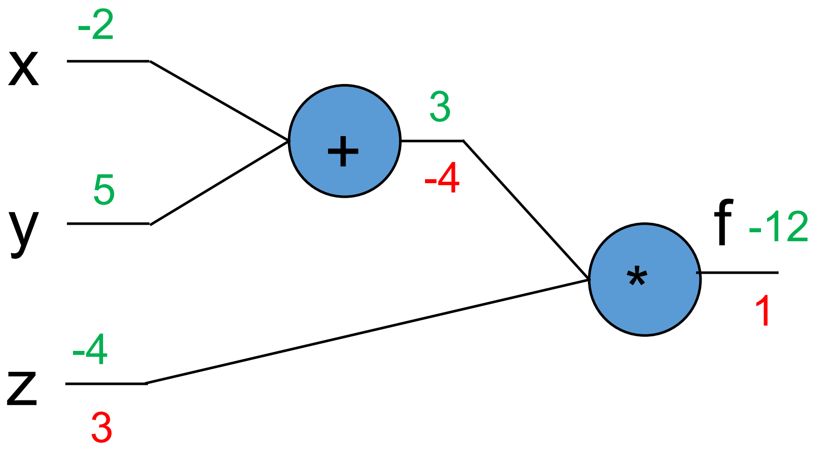
```
Forward pass:
q = x + y → q = 3
f = q * z → f = -12
```

Compound expressions

$$f(x,y,z) = (x+y)z = qz$$

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$\frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$



x=-2, y=5, z=-4

```
Forward pass:
q = x + y → q = 3
f = q * z → f = -12

Backward pass:
dfdz = q → 3
dfdq = z → -4
```

NORTHWESTERN UNIVERSITY

Compound expressions

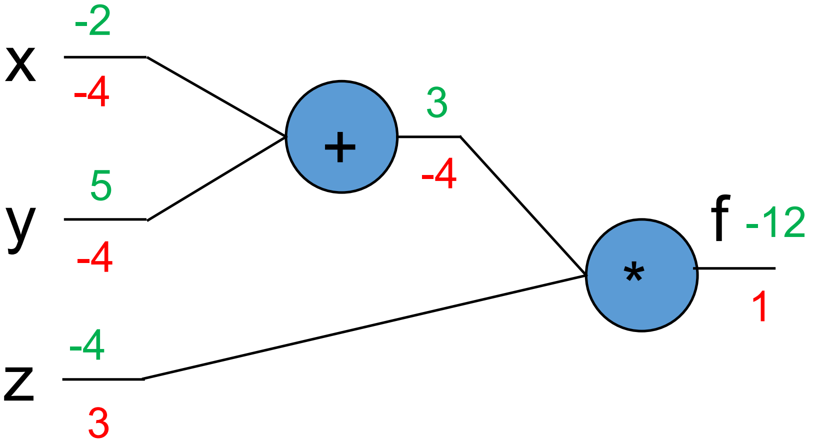
$$f(x,y,z) = (x+y)z = qz$$

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$\frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Chain rule:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial x}$$



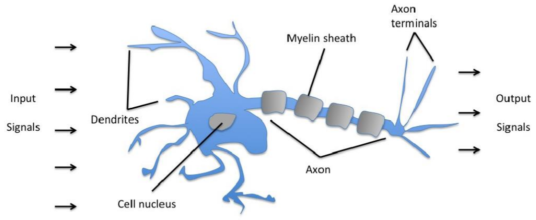
x=-2, y=5, z=-4

```
Forward pass:
q = x + y → q = 3
f = q * z → f = -12

Backward pass:
dfdz = q → 3
dfdq = z → -4

Back one more level:
dfdx = dqdx*dfdq = 1.0*dfdq →-4
dfdy = dqdy*dfdq = 1.0*dfdq →-4
```

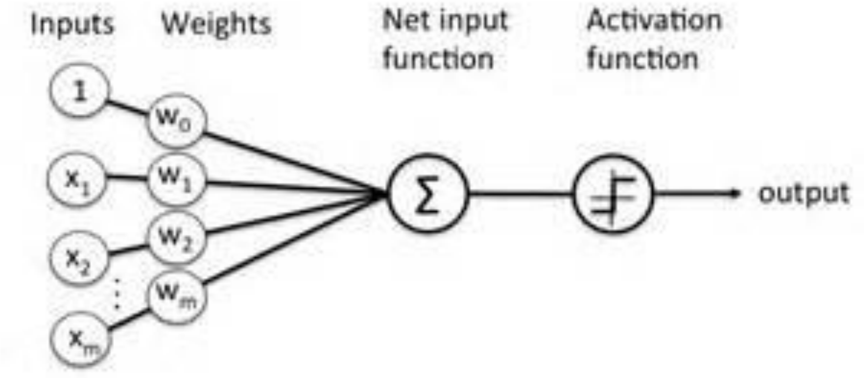
Congratulations!

- We've now discovered the basics of learning weights in a NN take a step along direction of steepest descent (gradient)
- NN consists of elementary operations such as: +, *, /, e^x
- Neural network can be expressed in terms of these basic operations

→ And we've learned how to optimize one part of these



Schematic of a biological neuron.

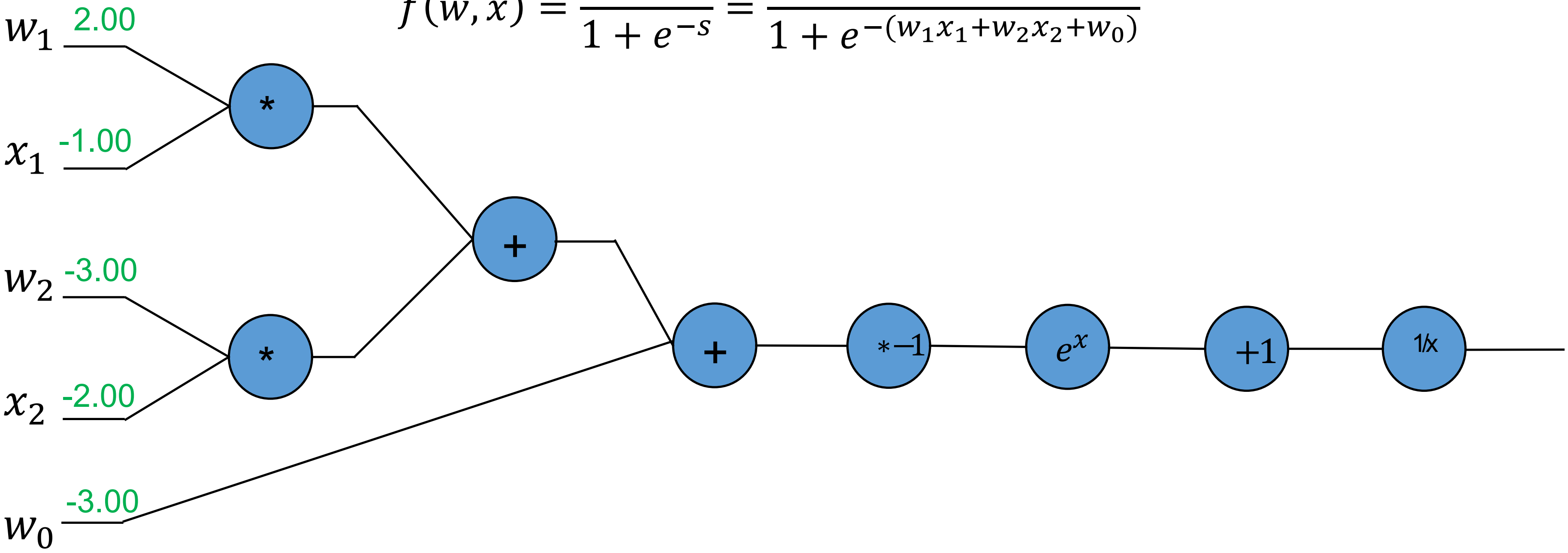


Schematic of Rosenblatt's perceptron.

$$f(w,x) = \frac{1}{1+e^{-s}}$$
$$= \frac{1}{1+e^{-(w_0+w_1x_1+w_2x_2)}}$$

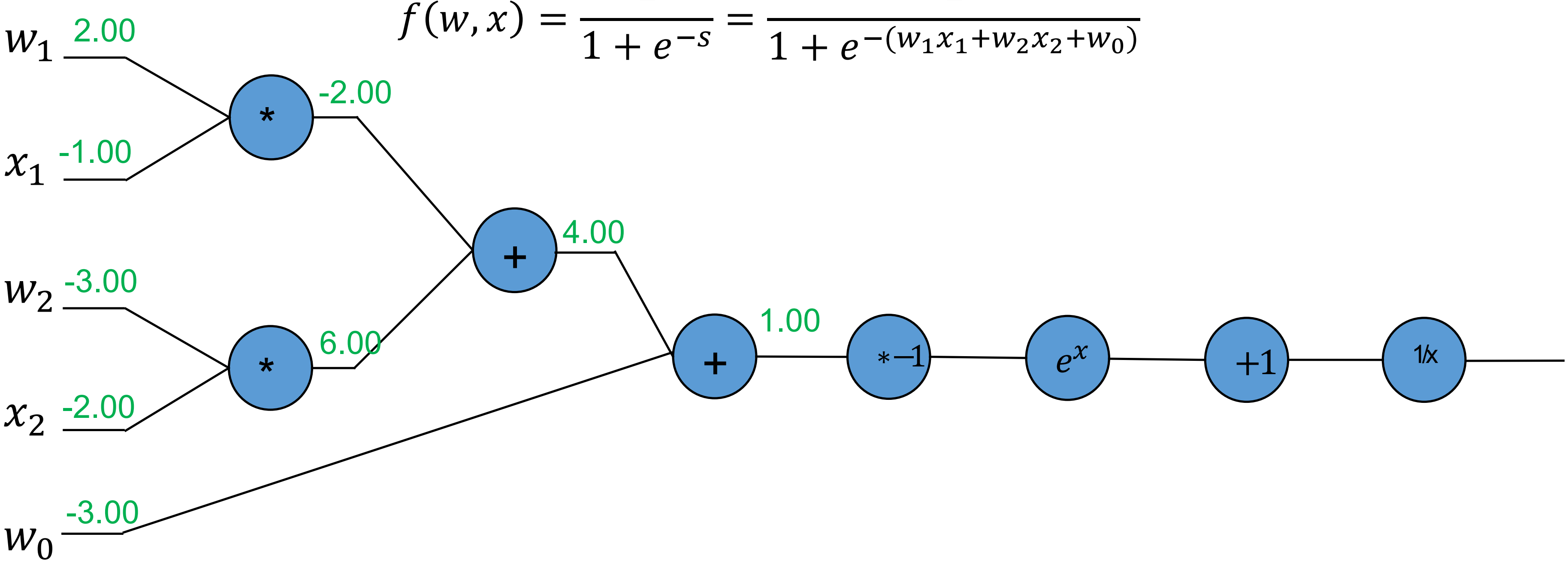
Sigmoid example

$$f(w,x) = \frac{1}{1+e^{-s}} = \frac{1}{1+e^{-(w_1x_1+w_2x_2+w_0)}}$$



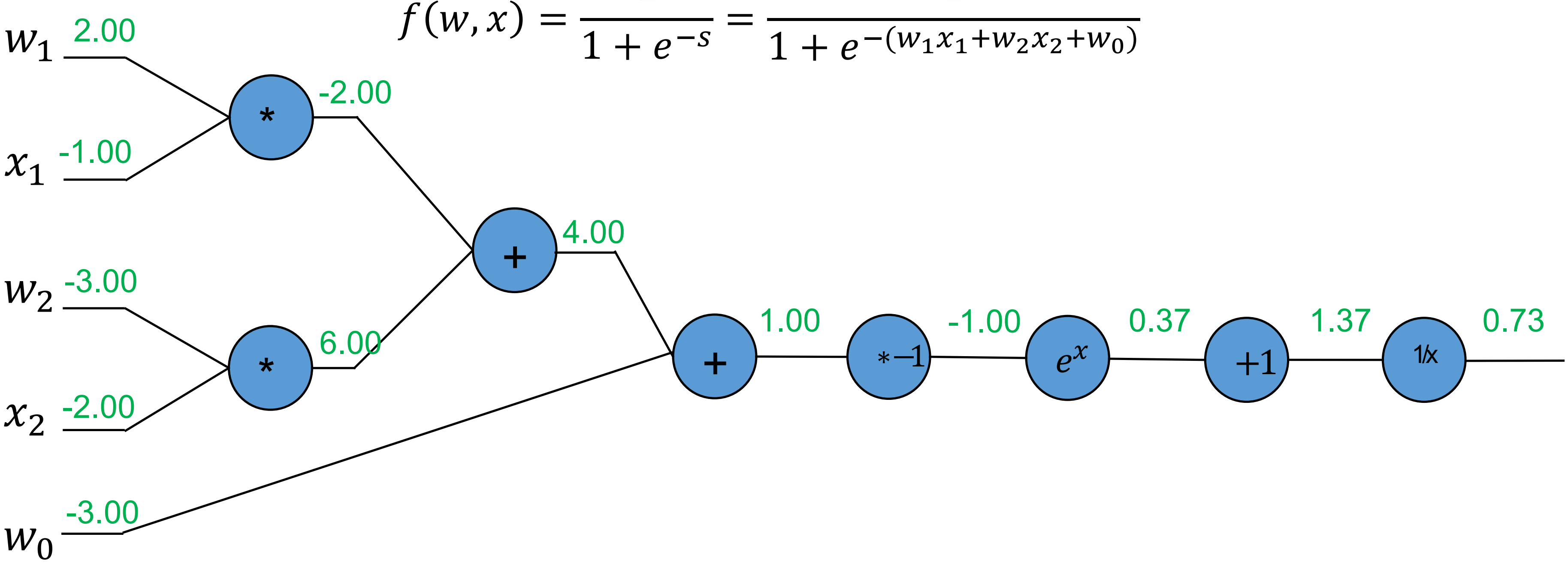
Sigmoid example

$$f(w,x) = \frac{1}{1+e^{-s}} = \frac{1}{1+e^{-(w_1 x_1 + w_2 x_2 + w_0)}}$$



Sigmoid example

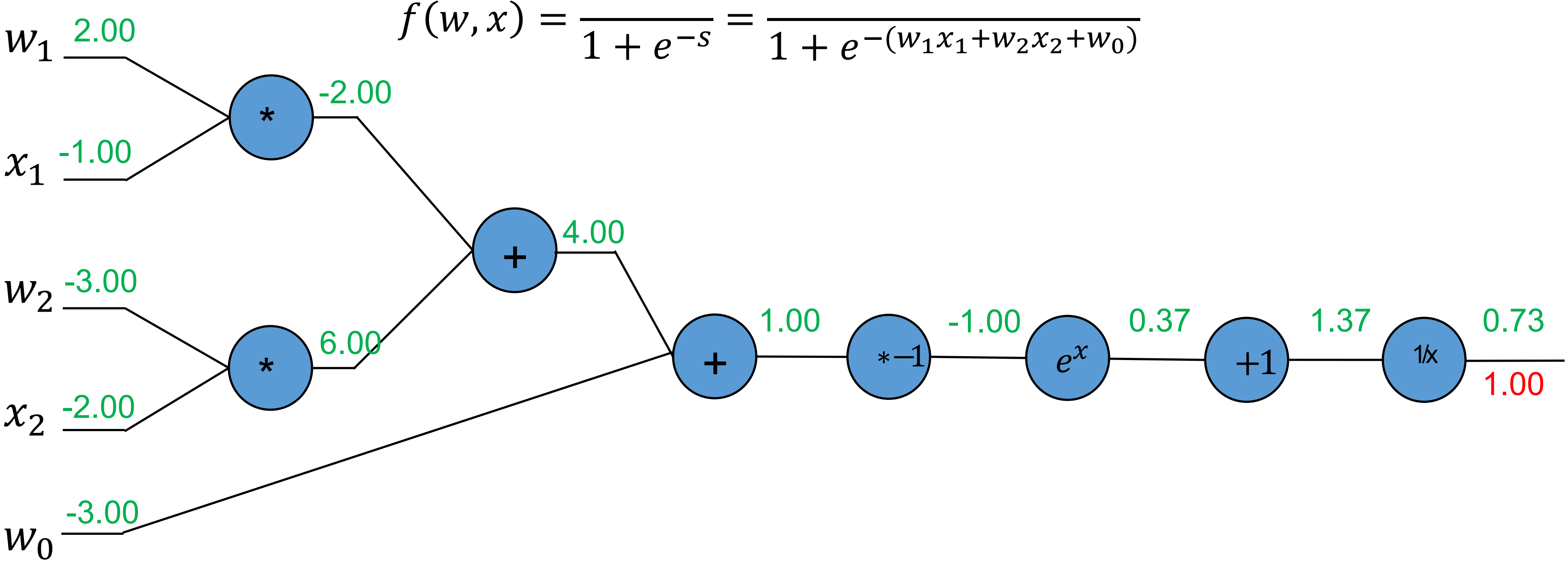
$$f(w,x)=\frac{1}{1+e^{-s}}=\frac{1}{1+e^{-(w_1x_1+w_2x_2+w_0)}}$$



NORTHWESTERN UNIVERSITY

Sigmoid example

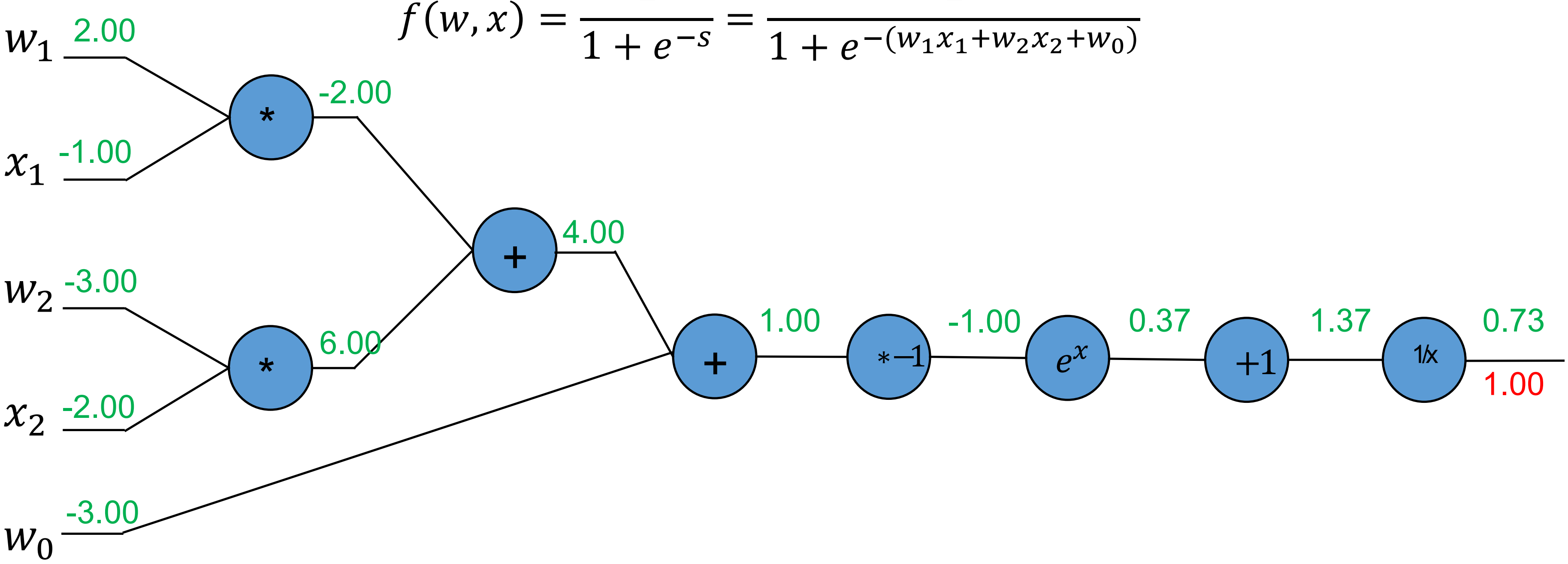
$$f(w,x) = \frac{1}{1+e^{-s}} = \frac{1}{1+e^{-(w_1x_1+w_2x_2+w_0)}}$$



$f(x) = e^x$	→	$\frac{\partial f}{\partial x} =$?	$f(x) = \frac{1}{x}$	→	$\frac{\partial f}{\partial x} =$?
$f_a(x) = ax$	→	$\frac{\partial f}{\partial x} =$?	$f_c(x) = c + x$	→	$\frac{\partial f}{\partial x} =$?

Sigmoid example

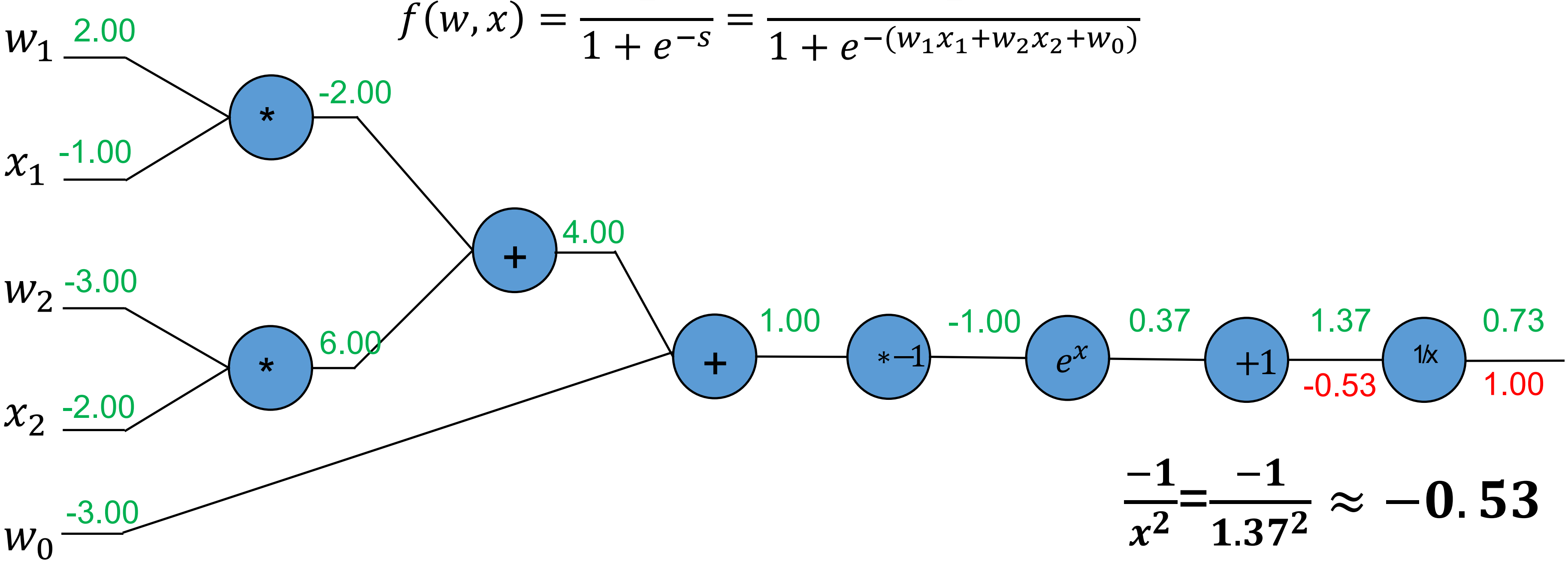
$$f(w,x) = \frac{1}{1+e^{-s}} = \frac{1}{1+e^{-(w_1x_1+w_2x_2+w_0)}}$$



$f(x) = e^x$ →	$\frac{\partial f}{\partial x} = e^x$	$f(x) = \frac{1}{x}$ →	$\frac{\partial f}{\partial x} = \frac{-1}{x^2}$
$f_a(x) = ax$ →	$\frac{\partial f}{\partial x} = a$	$f_c(x) = c + x$ →	$\frac{\partial f}{\partial x} = 1$

Sigmoid example

$$f(w,x) = \frac{1}{1+e^{-s}} = \frac{1}{1+e^{-(w_1x_1+w_2x_2+w_0)}}$$

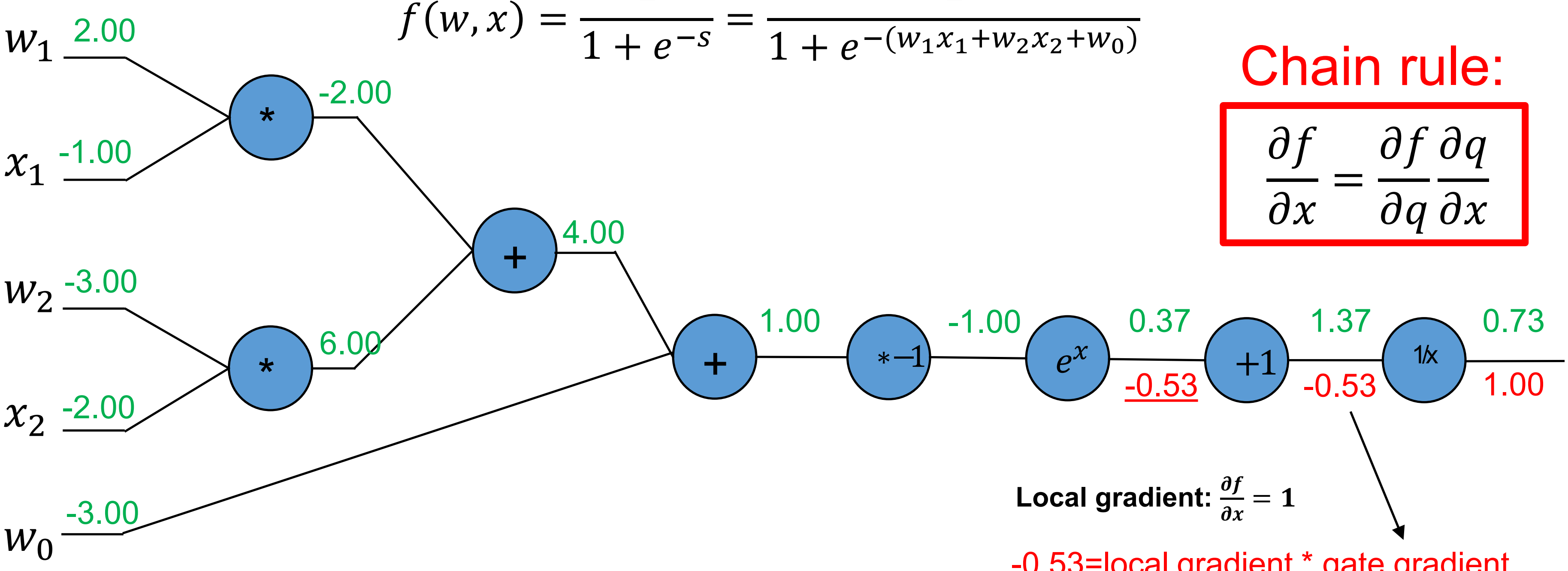


$$\frac{-1}{x^2} = \frac{-1}{1.37^2} \approx -0.53$$

$f(x) = e^x \quad \rightarrow \quad \frac{\partial f}{\partial x} = e^x$	$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{\partial f}{\partial x} = \frac{-1}{x^2}$
$f_a(x) = ax \quad \rightarrow \quad \frac{\partial f}{\partial x} = a$	$f_c(x) = c + x \quad \rightarrow \quad \frac{\partial f}{\partial x} = 1$

Sigmoid example

$$f(w,x) = \frac{1}{1+e^{-s}} = \frac{1}{1+e^{-(w_1x_1+w_2x_2+w_0)}}$$



Chain rule:

$$\boxed{\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial x}}$$

Local gradient: $\frac{\partial f}{\partial x} = 1$

-0.53=local gradient * gate gradient

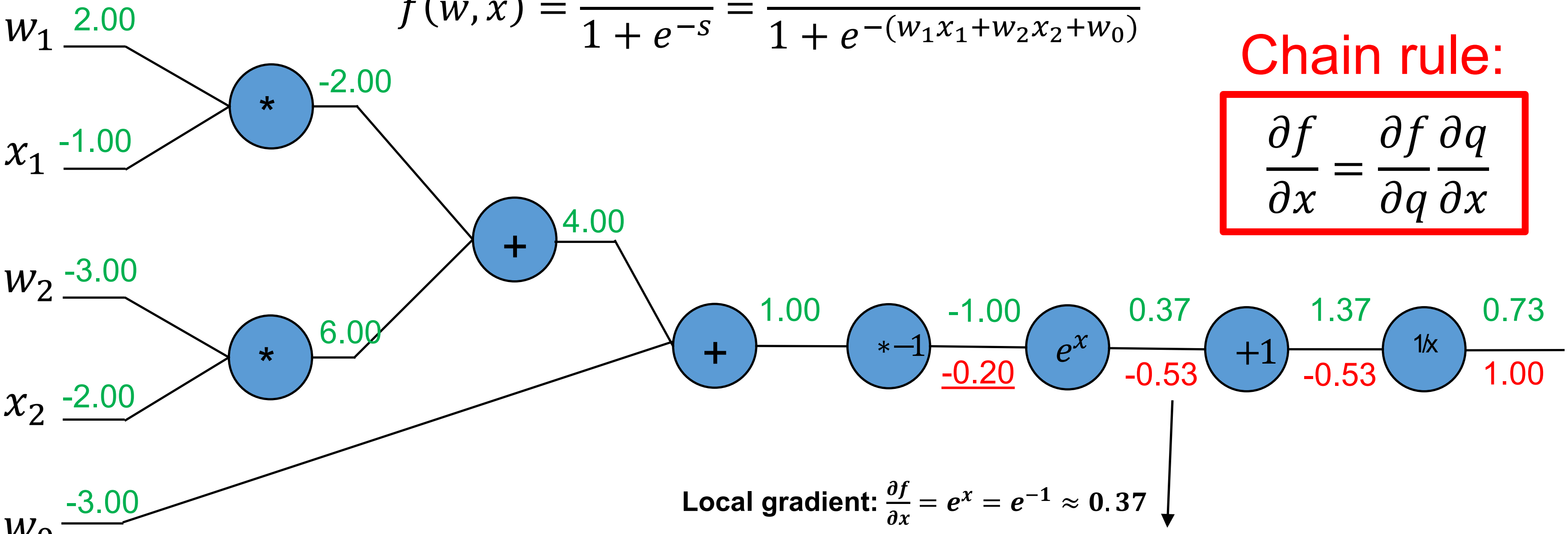
$f(x) = e^x \quad \rightarrow$	$\frac{\partial f}{\partial x} = e^x$	$f(x) = \frac{1}{x} \quad \rightarrow$	$\frac{\partial f}{\partial x} = \frac{-1}{x^2}$
$f_a(x) = ax \quad \rightarrow$	$\frac{\partial f}{\partial x} = a$	$f_c(x) = c + x \quad \rightarrow$	$\frac{\partial f}{\partial x} = 1$

Sigmoid example

$$f(w,x)=\frac{1}{1+e^{-s}}=\frac{1}{1+e^{-(w_1x_1+w_2x_2+w_0)}}$$

Chain rule:

$$\boxed{\frac{\partial f}{\partial x}=\frac{\partial f}{\partial q}\frac{\partial q}{\partial x}}$$



Local gradient: $\frac{\partial f}{\partial x}=e^x=e^{-1}\approx 0.37$

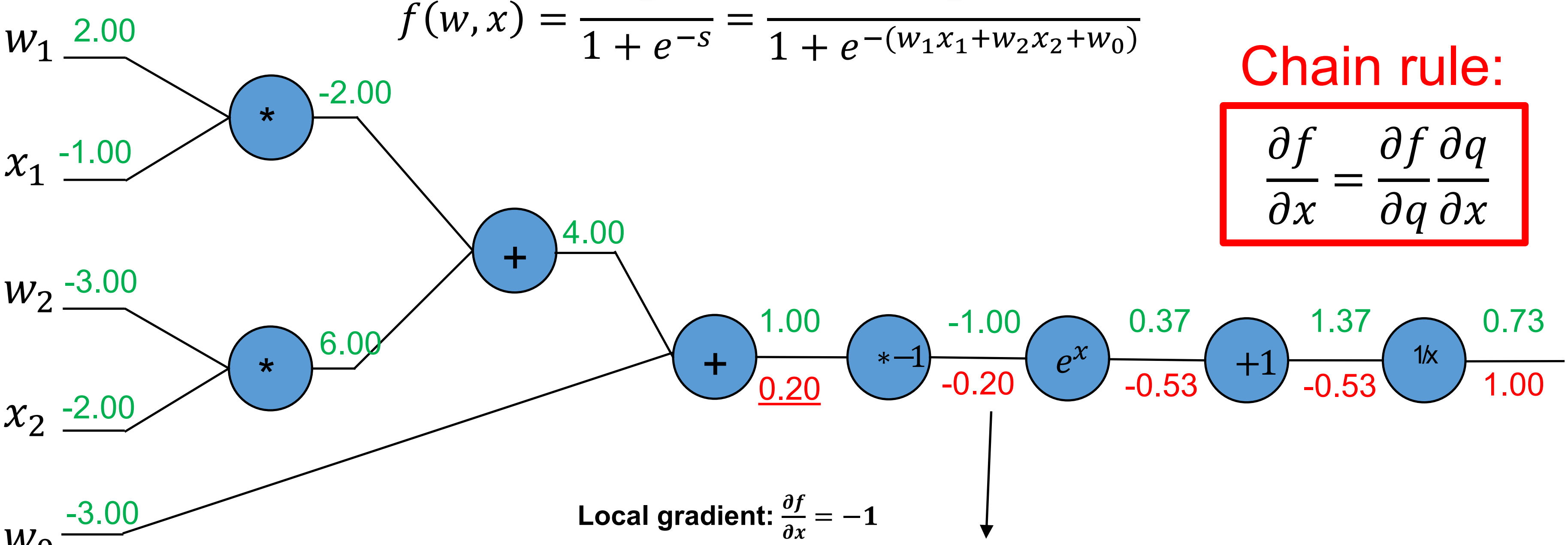
-0.20=local gradient * gate gradient=0.37*-0.53

$\boxed{f(x)=e^x \quad \rightarrow \quad \frac{\partial f}{\partial x}=e^x}$	$f(x)=\frac{1}{x} \quad \rightarrow \quad \frac{\partial f}{\partial x}=\frac{-1}{x^2}$
$f_a(x)=ax \quad \rightarrow \quad \frac{\partial f}{\partial x}=a$	$f_c(x)=c+x \quad \rightarrow \quad \frac{\partial f}{\partial x}=1$

NORTHWESTERN
UNIVERSITY

Sigmoid example

$$f(w,x) = \frac{1}{1+e^{-s}} = \frac{1}{1+e^{-(w_1x_1+w_2x_2+w_0)}}$$



Chain rule:

$$\boxed{\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial x}}$$

Local gradient: $\frac{\partial f}{\partial x} = -1$

0.20=local gradient * gate gradient=-1*-0.20

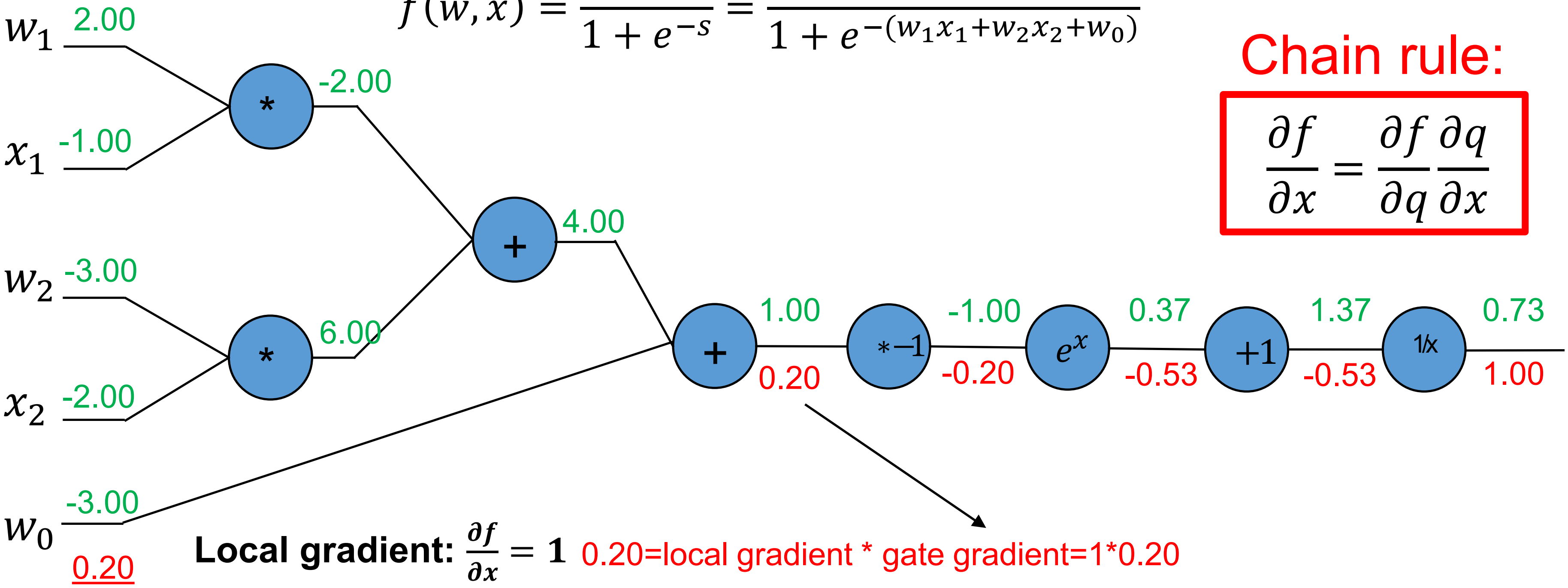
$f(x) = e^x$	→	$\frac{\partial f}{\partial x} = e^x$	$f(x) = \frac{1}{x}$	→	$\frac{\partial f}{\partial x} = \frac{-1}{x^2}$
$f_a(x) = ax$	→	$\frac{\partial f}{\partial x} = a$	$f_c(x) = c + x$	→	$\frac{\partial f}{\partial x} = 1$

Sigmoid example

$$f(w,x) = \frac{1}{1+e^{-s}} = \frac{1}{1+e^{-(w_1x_1+w_2x_2+w_0)}}$$

Chain rule:

$$\boxed{\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial x}}$$

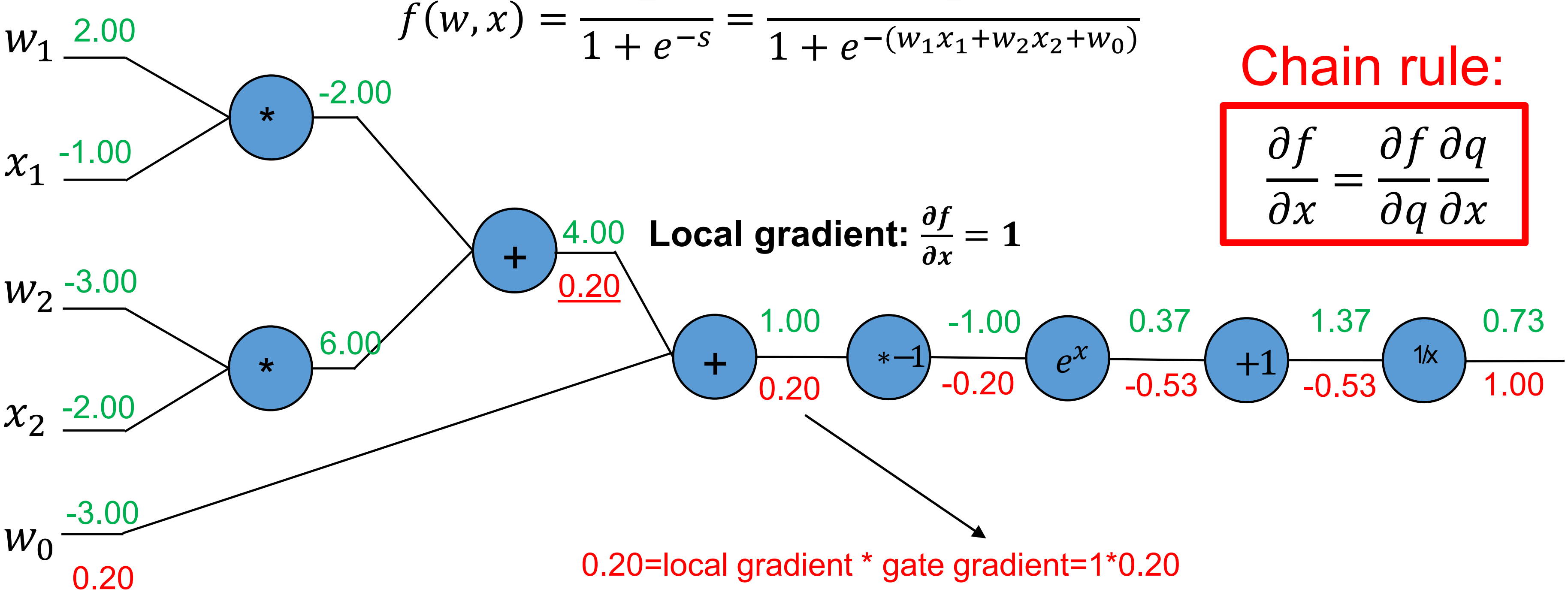


Local gradient: $\frac{\partial f}{\partial x} = 1$ 0.20=local gradient * gate gradient=1*0.20

$f(x) = e^x \quad \rightarrow \quad \frac{\partial f}{\partial x} = e^x$	$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{\partial f}{\partial x} = \frac{-1}{x^2}$
$f_a(x) = ax \quad \rightarrow \quad \frac{\partial f}{\partial x} = a$	$\boxed{f_c(x) = c + x \quad \rightarrow \quad \frac{\partial f}{\partial x} = 1}$

Sigmoid example

$$f(w,x) = \frac{1}{1+e^{-s}} = \frac{1}{1+e^{-(w_1x_1+w_2x_2+w_0)}}$$



Local gradient: $\frac{\partial f}{\partial x} = 1$

0.20=local gradient * gate gradient=1*0.20

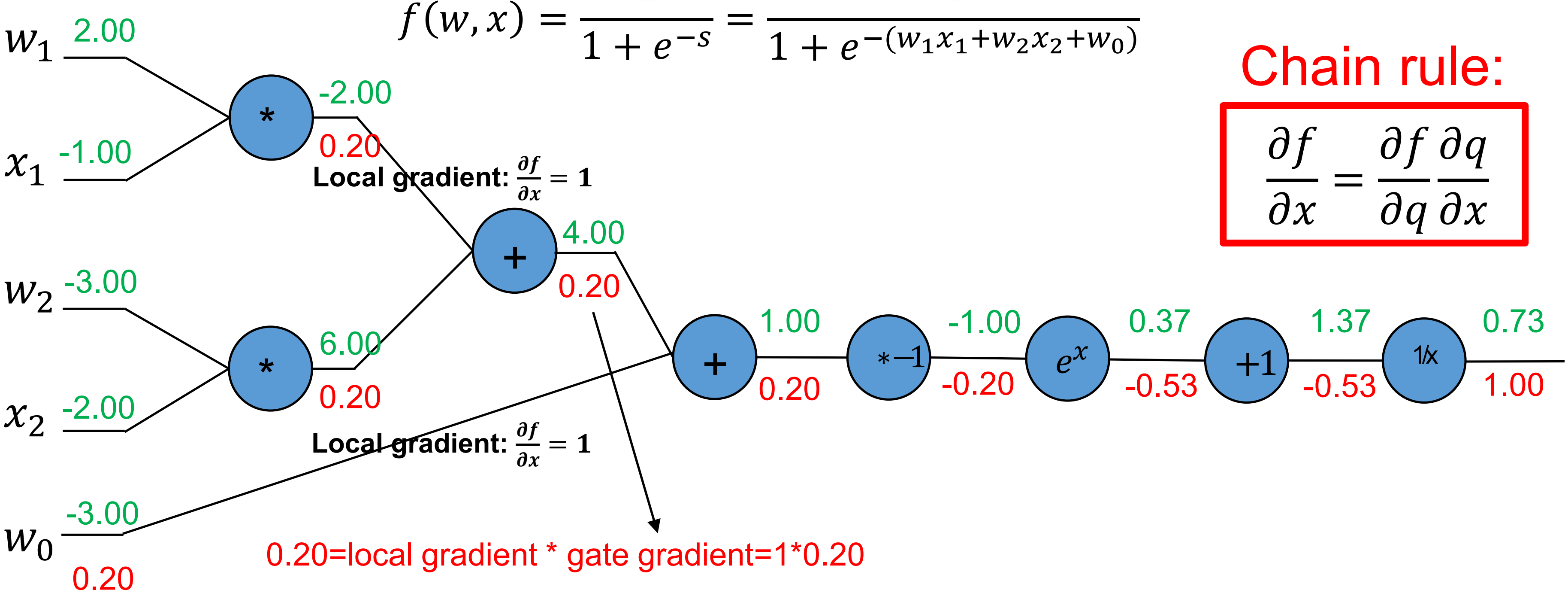
Chain rule:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial x}$$

$f(x) = e^x$	→	$\frac{\partial f}{\partial x} = e^x$	$f(x) = \frac{1}{x}$	→	$\frac{\partial f}{\partial x} = \frac{-1}{x^2}$
$f_a(x) = ax$	→	$\frac{\partial f}{\partial x} = a$	$f_c(x) = c + x$	→	$\frac{\partial f}{\partial x} = 1$

Sigmoid example

$$f(w,x) = \frac{1}{1+e^{-s}} = \frac{1}{1+e^{-(w_1x_1+w_2x_2+w_0)}}$$



Local gradient: $\frac{\partial f}{\partial x} = 1$

Local gradient: $\frac{\partial f}{\partial x} = 1$

0.20=local gradient * gate gradient=1*0.20

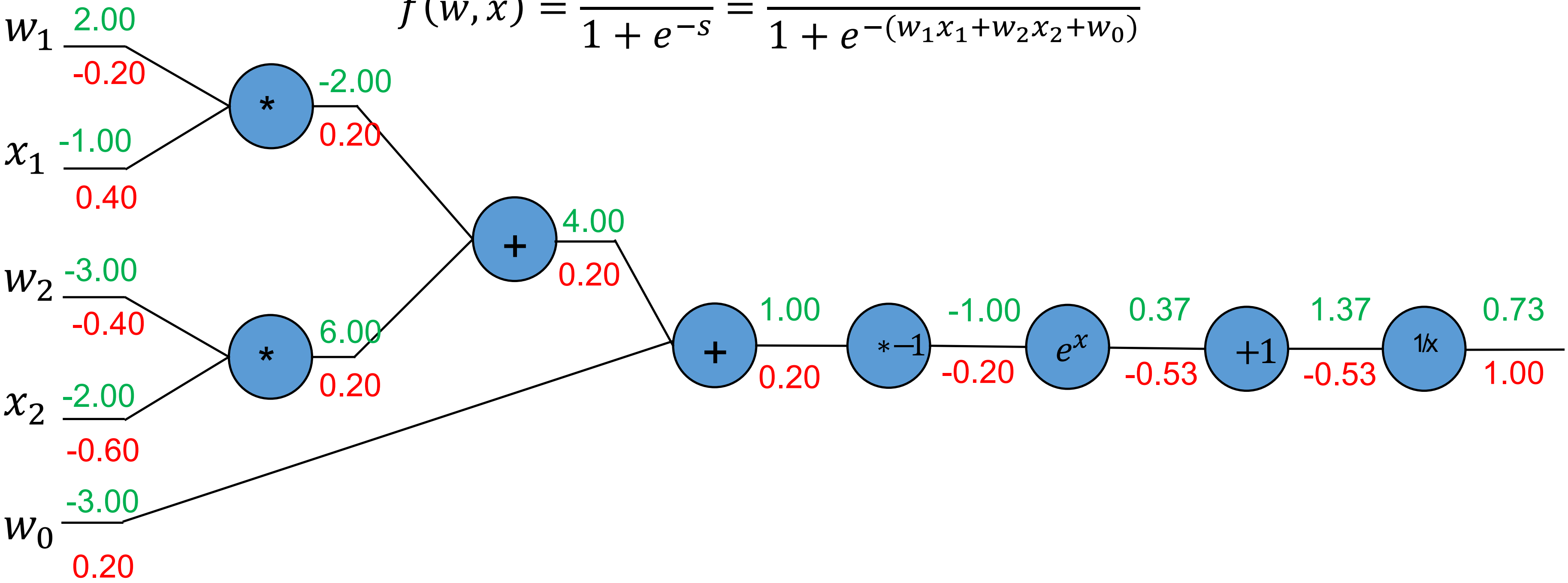
Chain rule:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial x}$$

$f(x) = e^x$	→	$\frac{\partial f}{\partial x} = e^x$	$f(x) = \frac{1}{x}$	→	$\frac{\partial f}{\partial x} = \frac{-1}{x^2}$
$f_a(x) = ax$	→	$\frac{\partial f}{\partial x} = a$	$f_c(x) = c + x$	→	$\frac{\partial f}{\partial x} = 1$

Sigmoid example

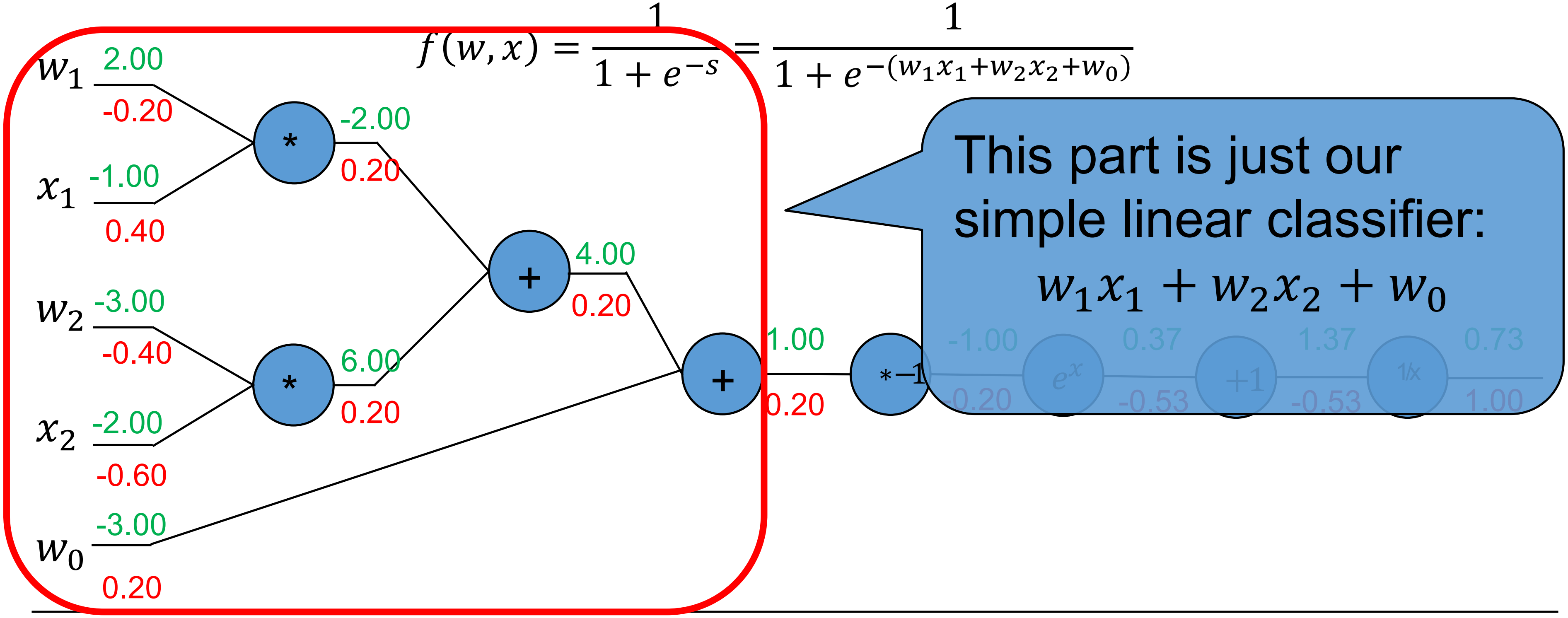
$$f(w,x) = \frac{1}{1+e^{-s}} = \frac{1}{1+e^{-(w_1 x_1 + w_2 x_2 + w_0)}}$$



$f(x) = e^x$	→	$\frac{\partial f}{\partial x} = e^x$	$f(x) = \frac{1}{x}$	→	$\frac{\partial f}{\partial x} = \frac{-1}{x^2}$
$f_a(x) = ax$	→	$\frac{\partial f}{\partial x} = a$	$f_c(x) = c + x$	→	$\frac{\partial f}{\partial x} = 1$

Sigmoid example

$$f(w,x) = \frac{1}{1+e^{-s}} = \frac{1}{1+e^{-(w_1x_1+w_2x_2+w_0)}}$$

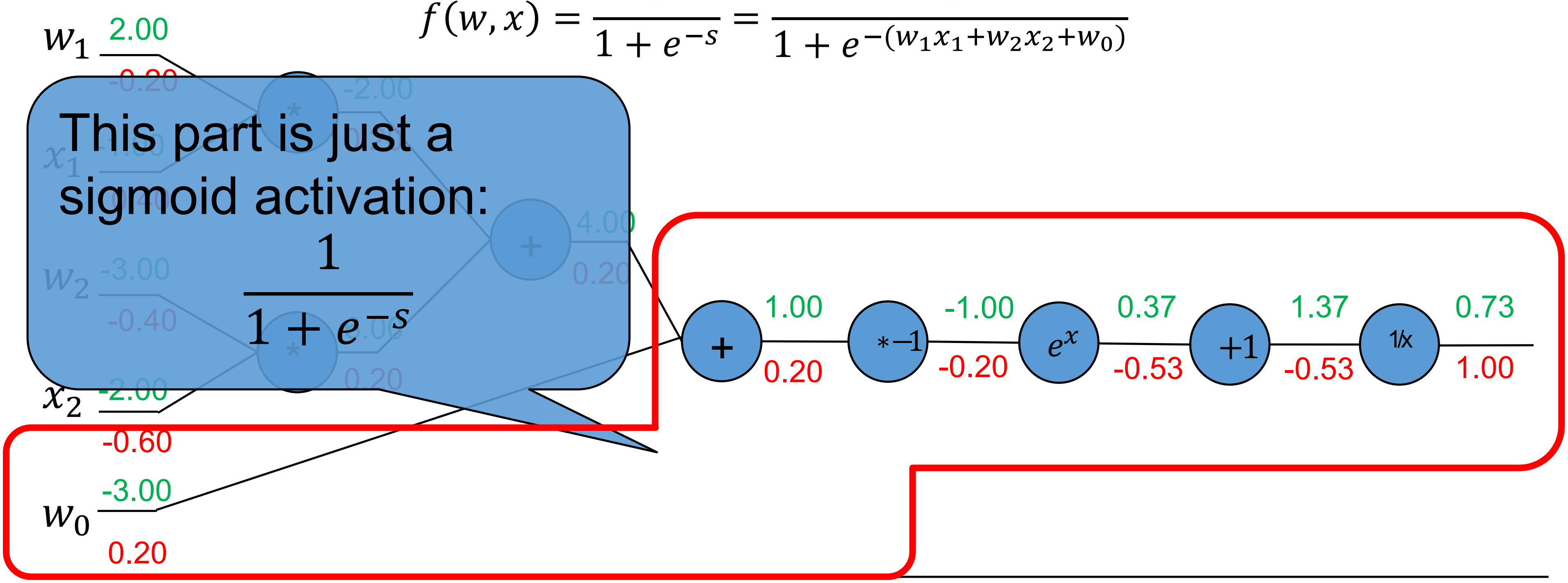


This part is just our simple linear classifier:
$w_1x_1 + w_2x_2 + w_0$

$f(x) = e^x$	→	$\frac{\partial f}{\partial x} = e^x$	$f(x) = \frac{1}{x}$	→	$\frac{\partial f}{\partial x} = \frac{-1}{x^2}$
$f_a(x) = ax$	→	$\frac{\partial f}{\partial x} = a$	$f_c(x) = c + x$	→	$\frac{\partial f}{\partial x} = 1$

Sigmoid example

$$f(w,x) = \frac{1}{1+e^{-s}} = \frac{1}{1+e^{-(w_1x_1+w_2x_2+w_0)}}$$



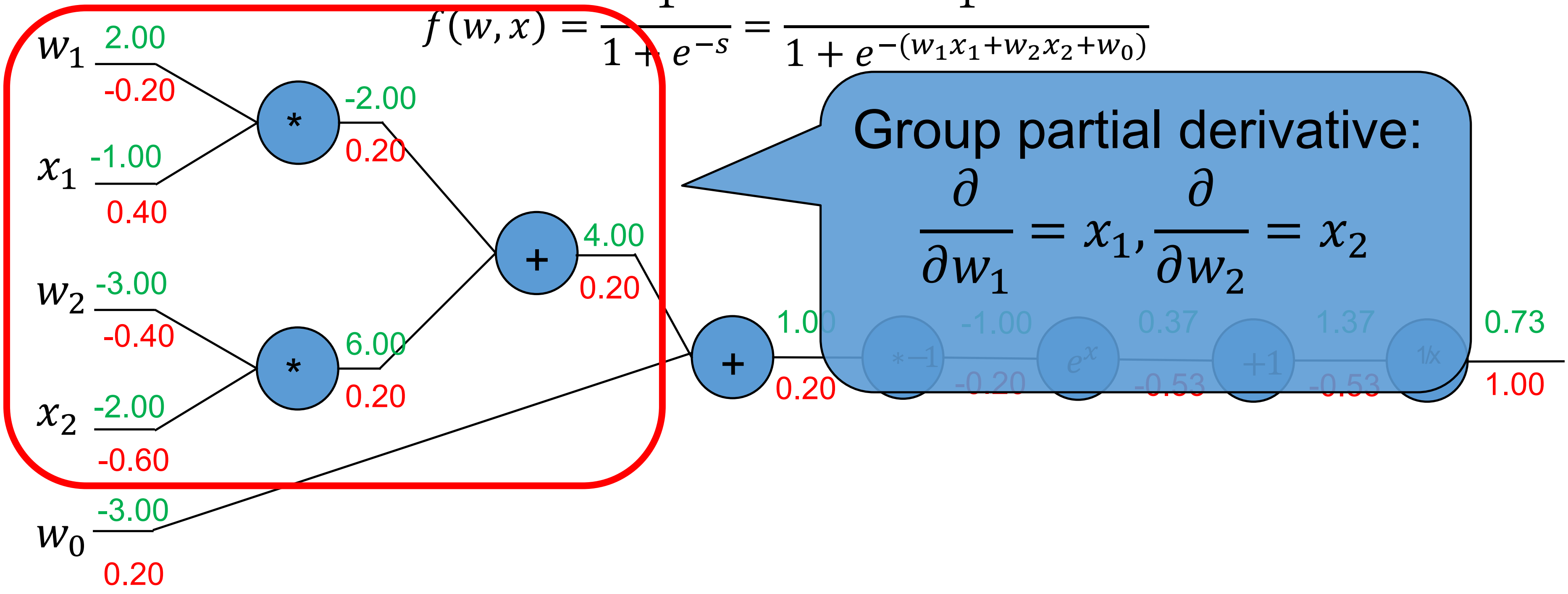
This part is just a sigmoid activation:

$$\frac{1}{1+e^{-s}}$$

$f(x) = e^x$	→	$\frac{\partial f}{\partial x} = e^x$	$f(x) = \frac{1}{x}$	→	$\frac{\partial f}{\partial x} = \frac{-1}{x^2}$
$f_a(x) = ax$	→	$\frac{\partial f}{\partial x} = a$	$f_c(x) = c + x$	→	$\frac{\partial f}{\partial x} = 1$

Sigmoid example

$$f(w,x) = \frac{1}{1+e^{-s}} = \frac{1}{1+e^{-(w_1x_1+w_2x_2+w_0)}}$$



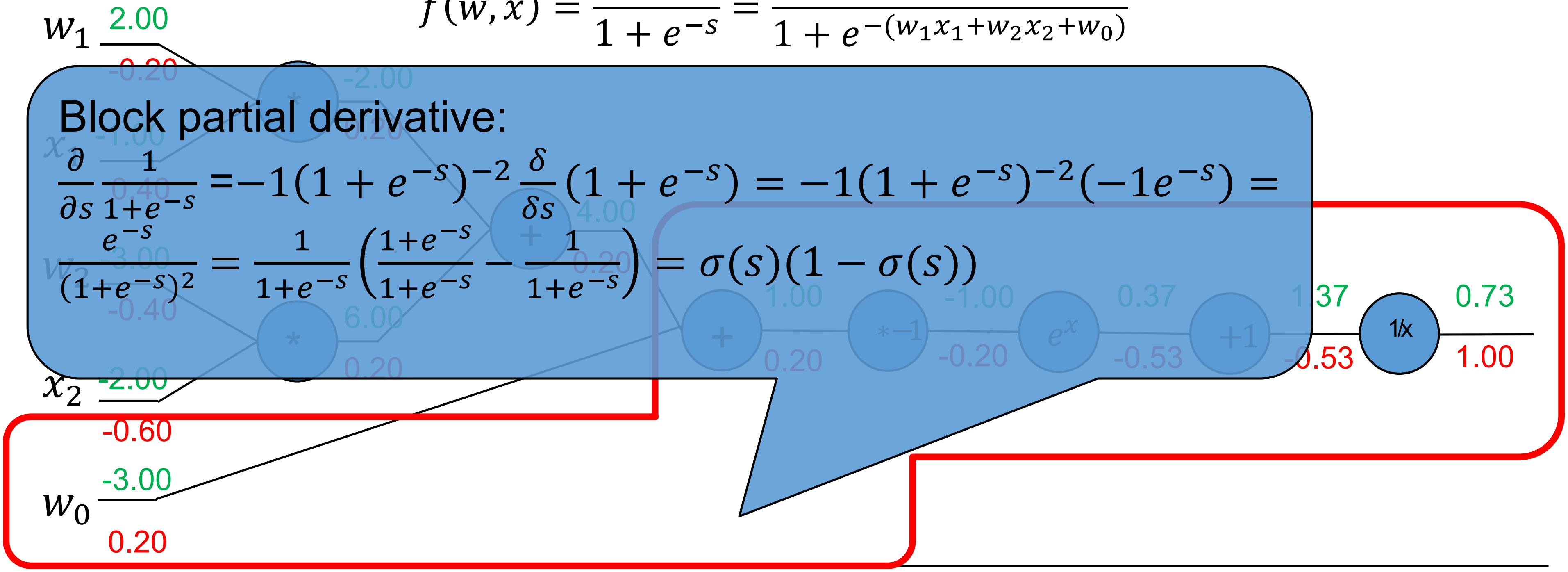
Group partial derivative:

$$\frac{\partial}{\partial w_1} = x_1, \frac{\partial}{\partial w_2} = x_2$$

$f(x) = e^x$	→	$\frac{\partial f}{\partial x} = e^x$	$f(x) = \frac{1}{x}$	→	$\frac{\partial f}{\partial x} = \frac{-1}{x^2}$
$f_a(x) = ax$	→	$\frac{\partial f}{\partial x} = a$	$f_c(x) = c + x$	→	$\frac{\partial f}{\partial x} = 1$

Sigmoid example

$$f(w,x) = \frac{1}{1+e^{-s}} = \frac{1}{1+e^{-(w_1x_1+w_2x_2+w_0)}}$$



Block partial derivative:

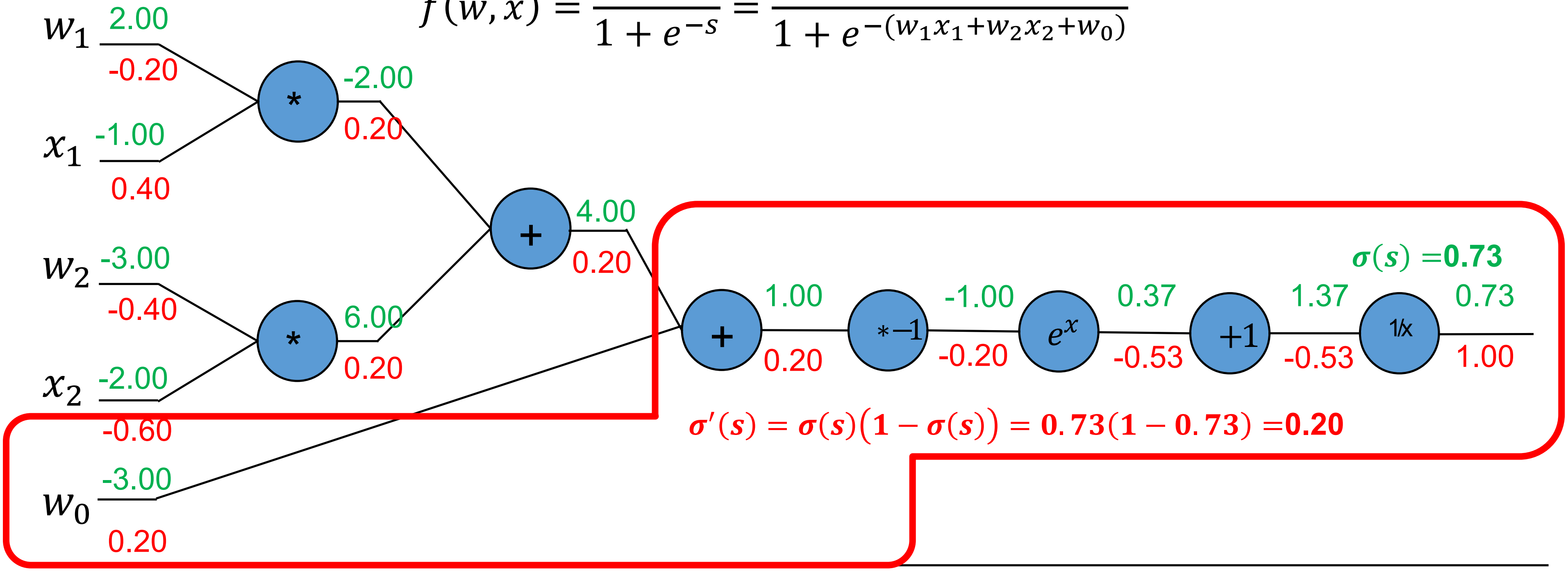
$$\frac{\partial}{\partial s}\frac{1}{1+e^{-s}} = -1(1+e^{-s})^{-2}\frac{\delta}{\delta s}(1+e^{-s}) = -1(1+e^{-s})^{-2}(-1e^{-s}) =$$

$$\frac{e^{-s}}{(1+e^{-s})^{2}} = \frac{1}{1+e^{-s}}\left(\frac{1+e^{-s}}{1+e^{-s}} - \frac{1}{1+e^{-s}}\right) = \sigma(s)(1-\sigma(s))$$

$f(x) = e^x \rightarrow \frac{\partial f}{\partial x} = e^x$	$f(x) = \frac{1}{x} \rightarrow \frac{\partial f}{\partial x} = \frac{-1}{x^2}$
$f_a(x) = ax \rightarrow \frac{\partial f}{\partial x} = a$	$f_c(x) = c + x \rightarrow \frac{\partial f}{\partial x} = 1$

Sigmoid example

$$f(w,x) = \frac{1}{1+e^{-s}} = \frac{1}{1+e^{-(w_1x_1+w_2x_2+w_0)}}$$



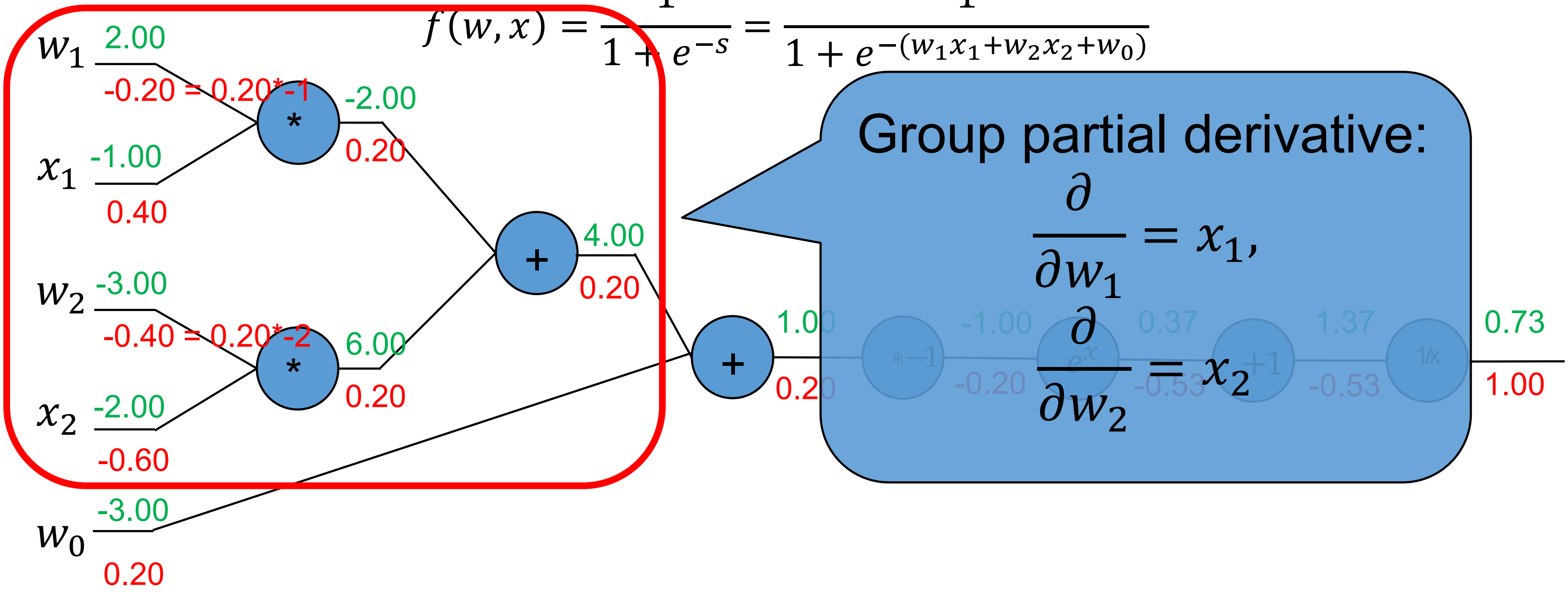
$\sigma(s) = \mathbf{0.73}$

$\sigma'(s) = \sigma(s)(1-\sigma(s)) = 0.73(1-0.73) = \mathbf{0.20}$

$f(x) = e^x \quad \rightarrow \quad \frac{\partial f}{\partial x} = e^x$	$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{\partial f}{\partial x} = \frac{-1}{x^2}$
$f_a(x) = ax \quad \rightarrow \quad \frac{\partial f}{\partial x} = a$	$f_c(x) = c + x \quad \rightarrow \quad \frac{\partial f}{\partial x} = 1$

Sigmoid example

$$f(w,x) = \frac{1}{1+e^{-s}} = \frac{1}{1+e^{-(w_1x_1+w_2x_2+w_0)}}$$



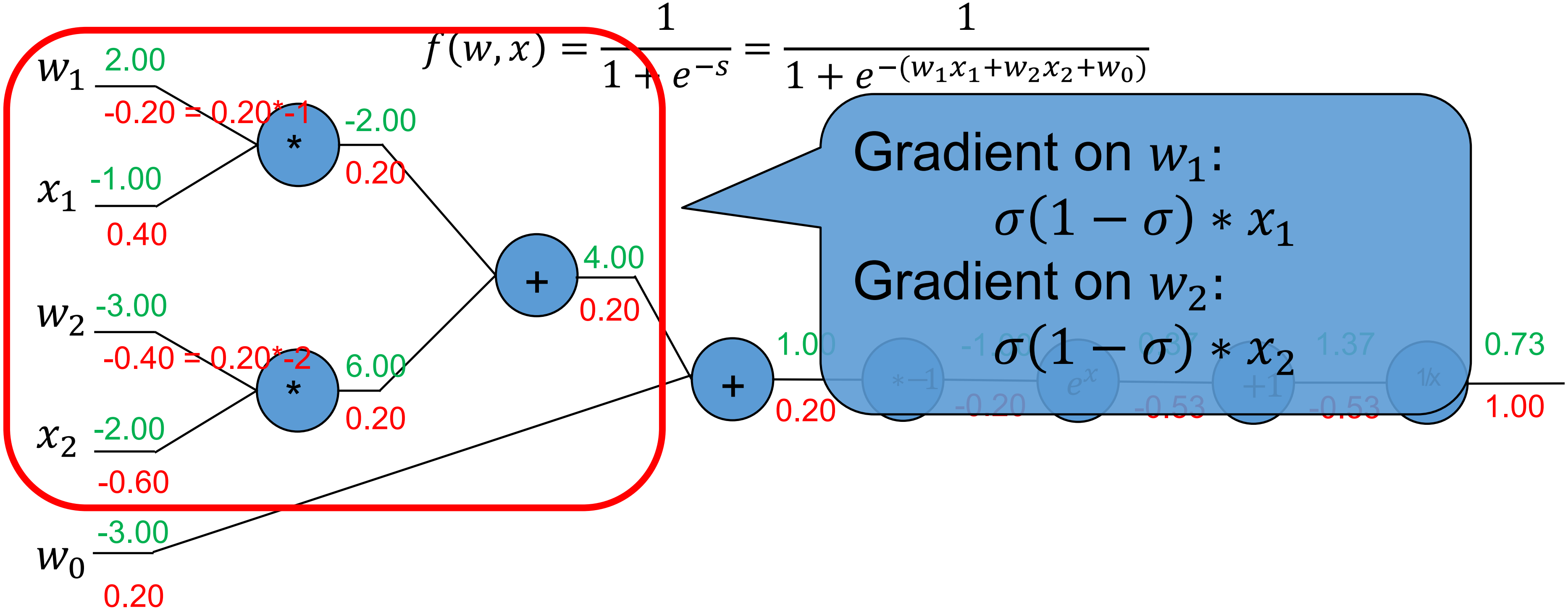
Group partial derivative:

$$\frac{\partial}{\partial w_1} = x_1,$$
$$\frac{\partial}{\partial w_2} = x_2$$

$f(x) = e^x$	→	$\frac{\partial f}{\partial x} = e^x$	$f(x) = \frac{1}{x}$	→	$\frac{\partial f}{\partial x} = \frac{-1}{x^2}$
$f_a(x) = ax$	→	$\frac{\partial f}{\partial x} = a$	$f_c(x) = c + x$	→	$\frac{\partial f}{\partial x} = 1$

Sigmoid example

$$f(w,x) = \frac{1}{1+e^{-s}} = \frac{1}{1+e^{-(w_1x_1+w_2x_2+w_0)}}$$



Gradient on w_1:

$$\sigma(1-\sigma) * x_1$$

Gradient on w_2:

$$\sigma(1-\sigma) * x_2$$

$f(x) = e^x$	→	$\frac{\partial f}{\partial x} = e^x$	$f(x) = \frac{1}{x}$	→	$\frac{\partial f}{\partial x} = \frac{-1}{x^2}$
$f_a(x) = ax$	→	$\frac{\partial f}{\partial x} = a$	$f_c(x) = c + x$	→	$\frac{\partial f}{\partial x} = 1$

- Gradient on MSE:
- $MSE = (y_i - t_i)^2$
- $\frac{\partial}{\partial y} MSE = 2 * (y_i - t_i)$
- MSE gradient * Sigmoid gradient:
- Overall gradient=MSE gradient * Sigmoid gradient:
- $\underset{\text{MSE}}{\boxed{2(\sigma - t)}} * \underset{\text{Sigmoid}}{\boxed{\sigma(1 - \sigma)}} * \underset{\text{Dot product}}{\boxed{x_1}}$

Now that we know the gradient…

- Pros:
  - Simple
- Cons:
  - Slow (updates only one direction at a time)
  - Requires lots of evaluation (2x params)
  - Stepping in one direction can "undo" gains in another
  - Finite difference unbalanced (so use centered difference)

Improvement:
Take a step η in the direction of steepest descent as determined by the analytical gradient

Use numeric gradient to double-check analytical gradient

$$slope \approx \lim_{h \to 0} \frac{f(a+h) - f(a-h)}{2h}$$

Why?

Finite difference error: O(Δh)

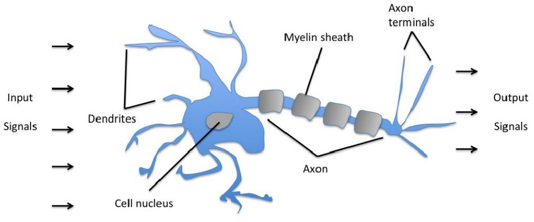
Centered difference error: O(Δh^2)

http://www.mathematik.uni-dortmund.de/~kuzmin/cfdintro/lecture4.pdf

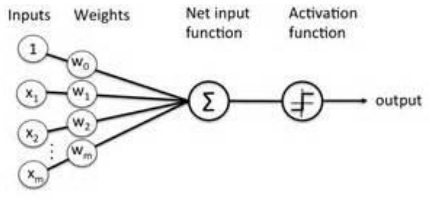
NORTHWESTERN UNIVERSITY

Sigmoid example

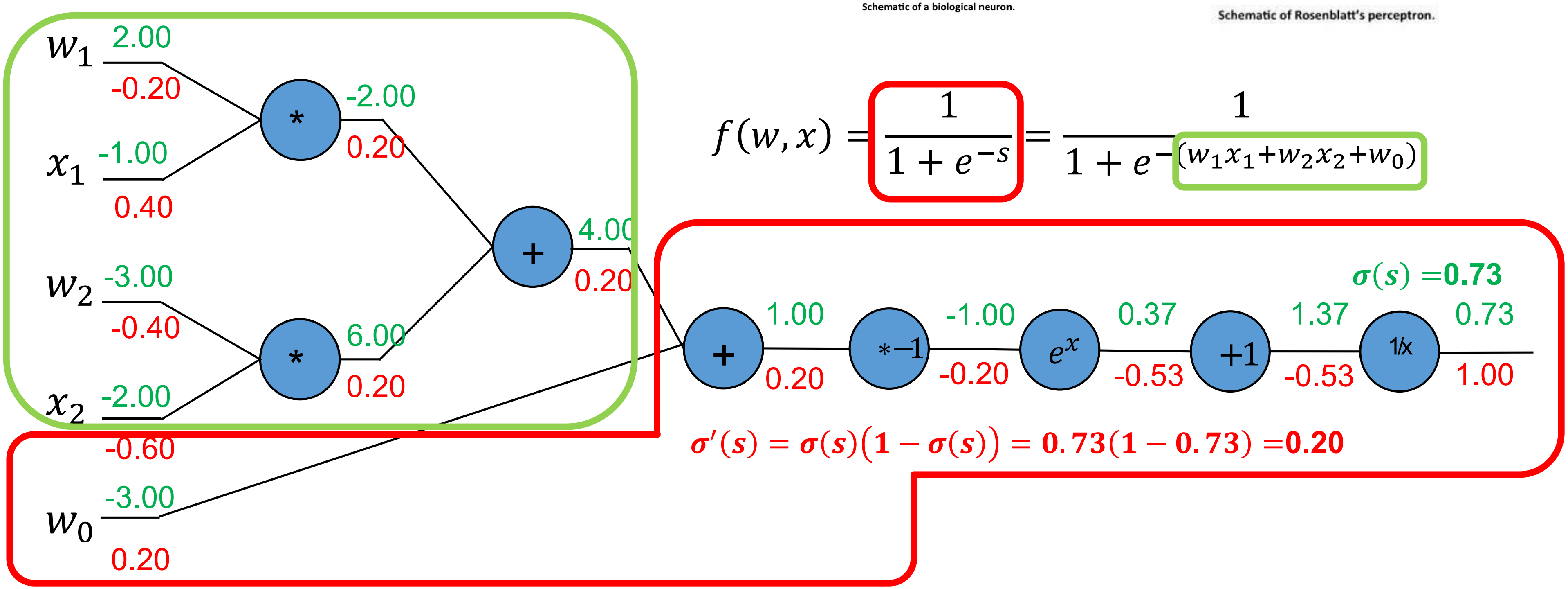
- Our long example was just a single neuron
- This neuron can be trained using backpropagation



Schematic of a biological neuron.



Schematic of Rosenblatt's perceptron.



$$f(w,x) = \boxed{\frac{1}{1+e^{-s}}} = \frac{1}{1+e^{-\boxed{(w_1x_1+w_2x_2+w_0)}}}$$

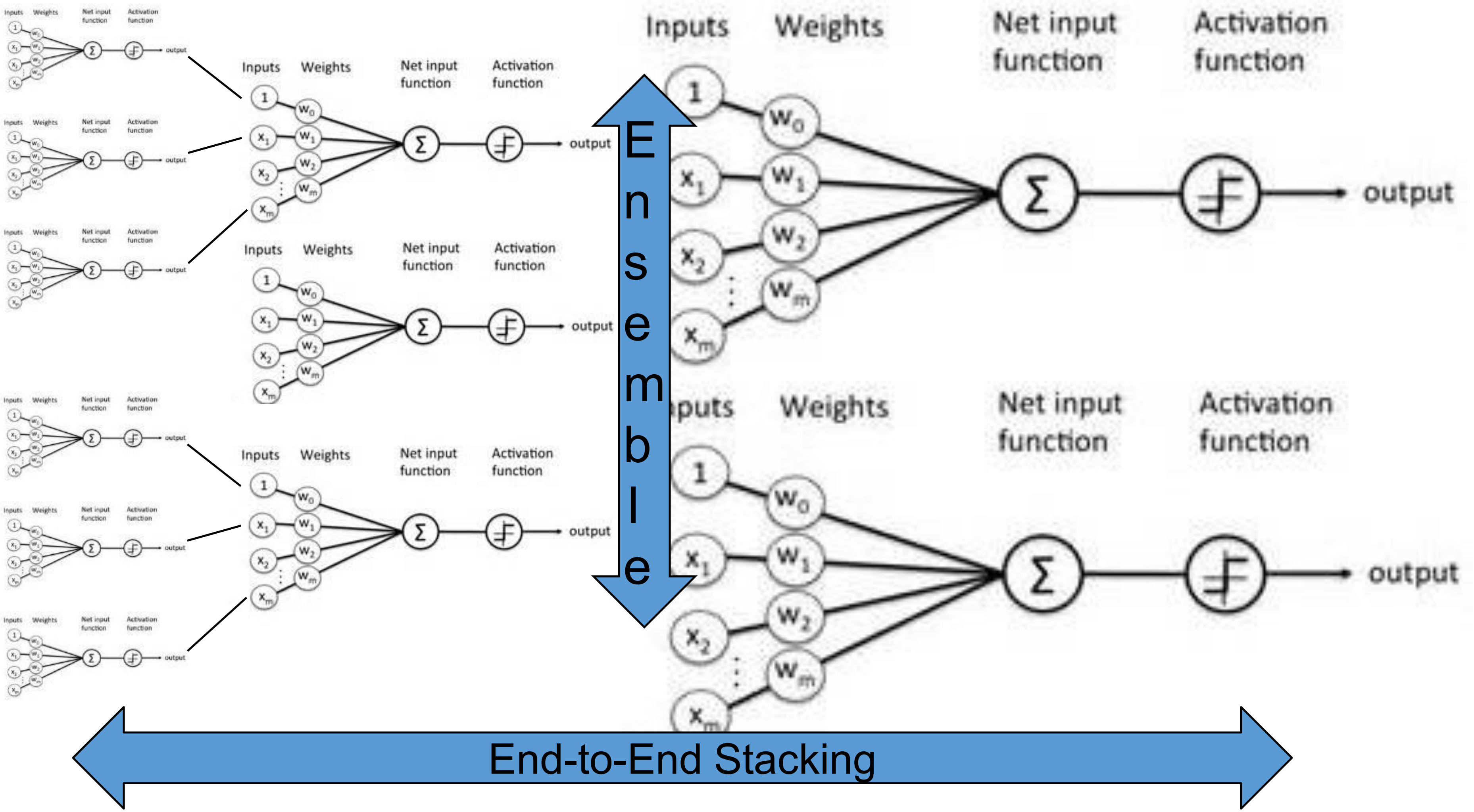
$\boldsymbol{\sigma(s) = 0.73}$

$\boldsymbol{\sigma'(s) = \sigma(s)\big(1-\sigma(s)\big) = 0.73(1-0.73) = 0.20}$

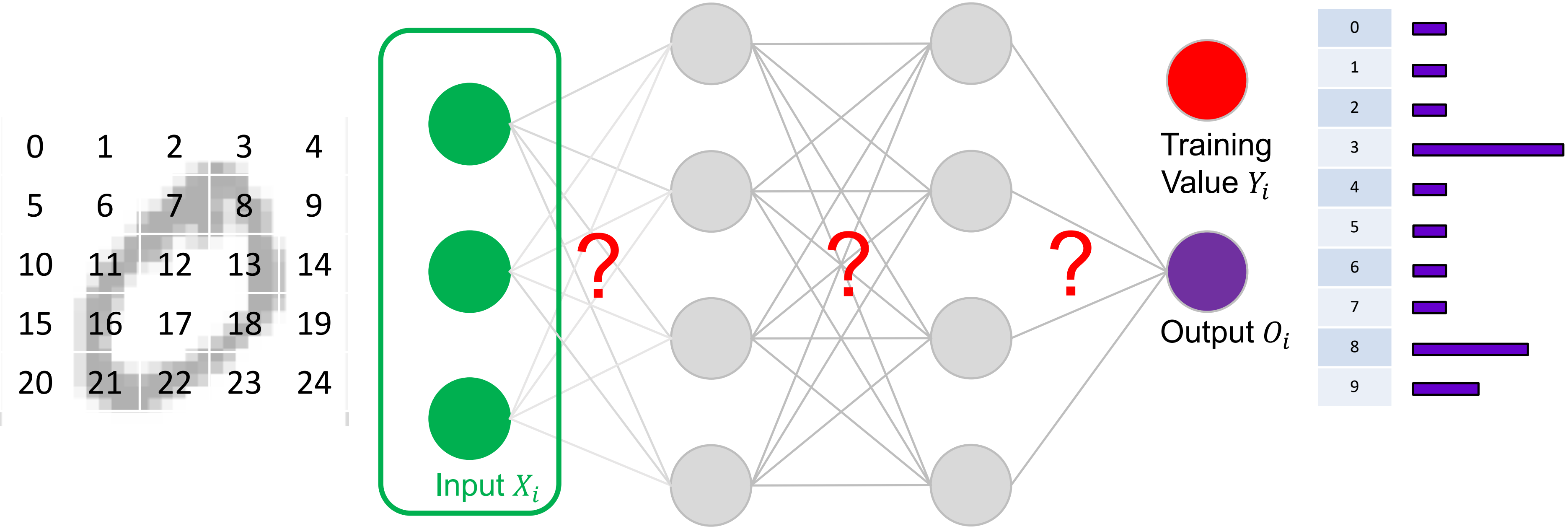
MSIA 490-30: Deep Learning. Spring 2017.
Instructor: Dr. Ellick Chan. TA: Mark Harmon.

NORTHWESTERN
UNIVERSITY

"Layered" Logistic Regression

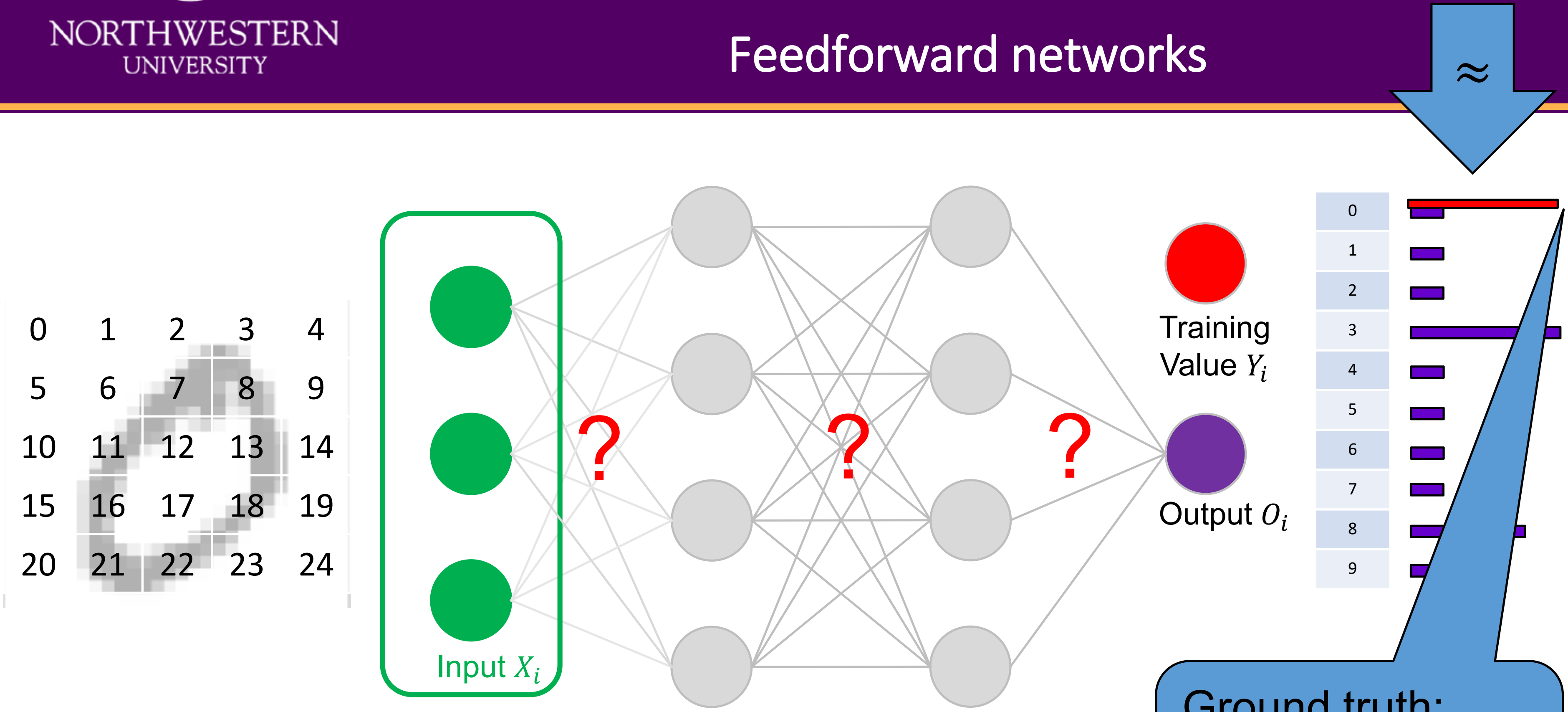


Feedforward networks



Goal: Learn weights such that outputs agree with training data → Minimize $\sum_i (Y_i - O_i)^2$

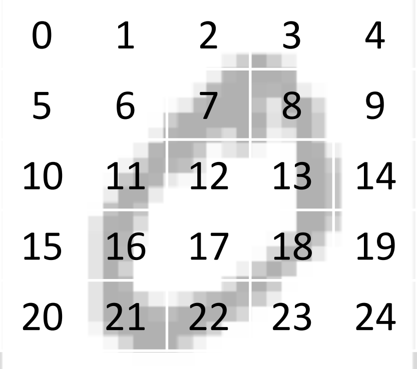
Feedforward networks



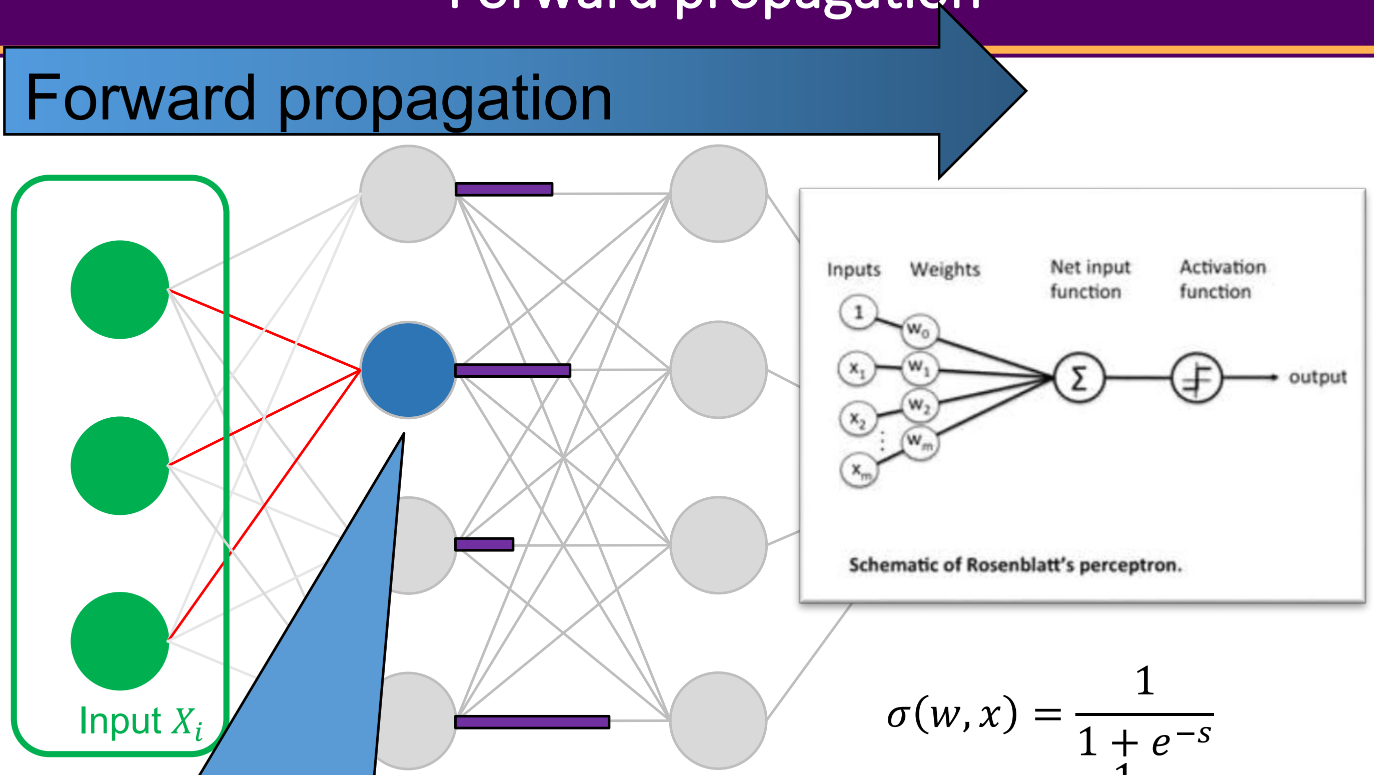
Ground truth: Make prediction match this

Goal: Learn weights such that outputs agree with training data → Minimize $\sum_i (Y_i - O_i)^2$

Forward propagation



0	1	2	3	4
5	6	7	8	9
10	11	12	13	14
15	16	17	18	19
20	21	22	23	24



Forward propagation

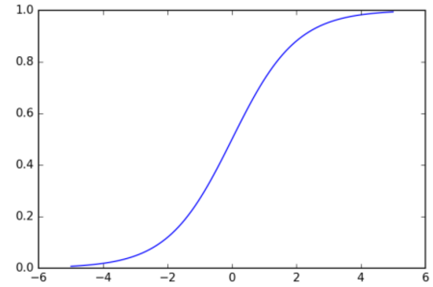
Input X_i

Schematic of Rosenblatt's perceptron.

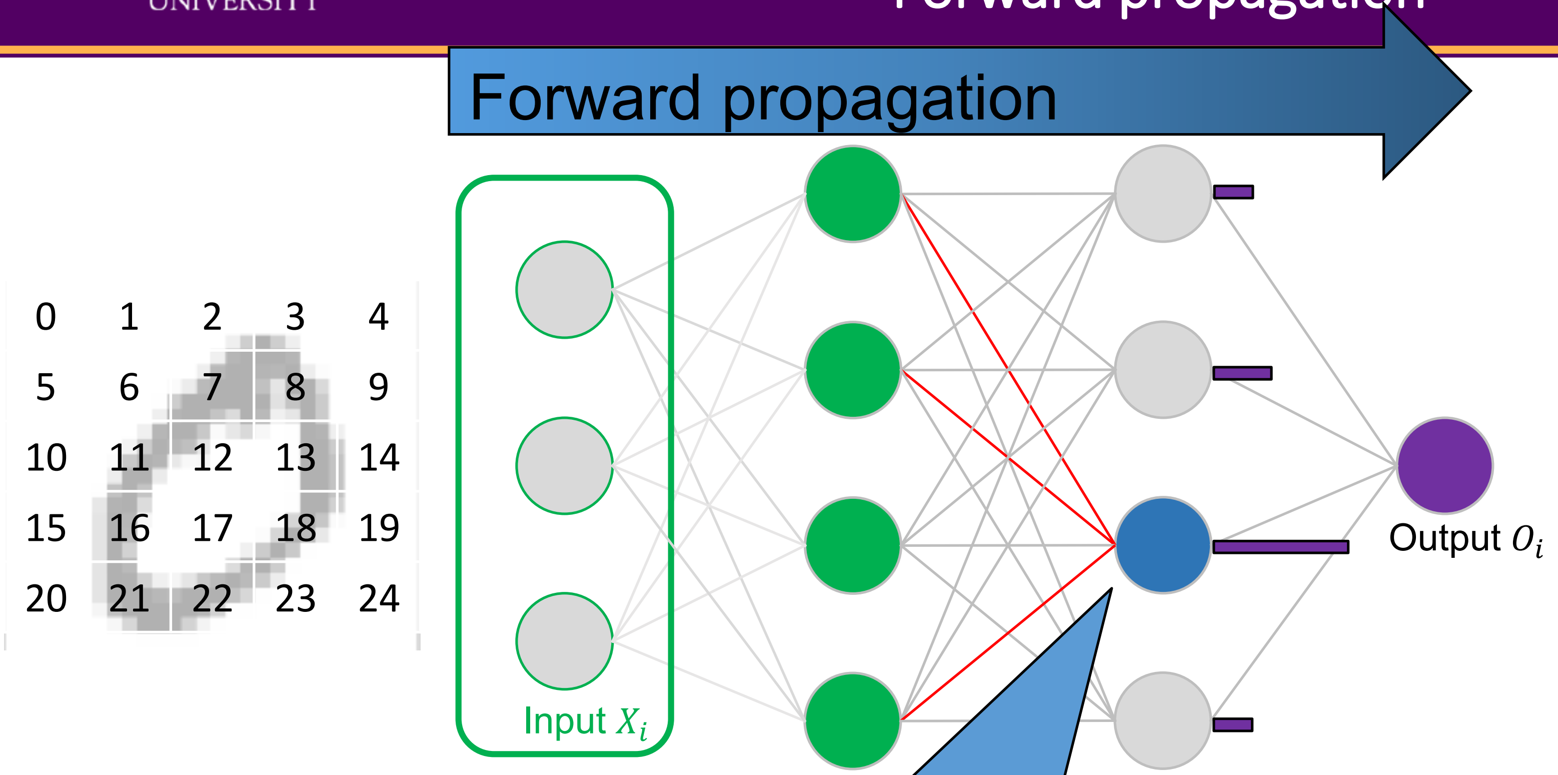
Each activation is simply a logistic regression.

Bar length = level of activation

$$\sigma(w,x) = \frac{1}{1+e^{-s}}$$
$$= \frac{1}{1+e^{-(w_0+w_1x_1+w_2x_2)}}$$



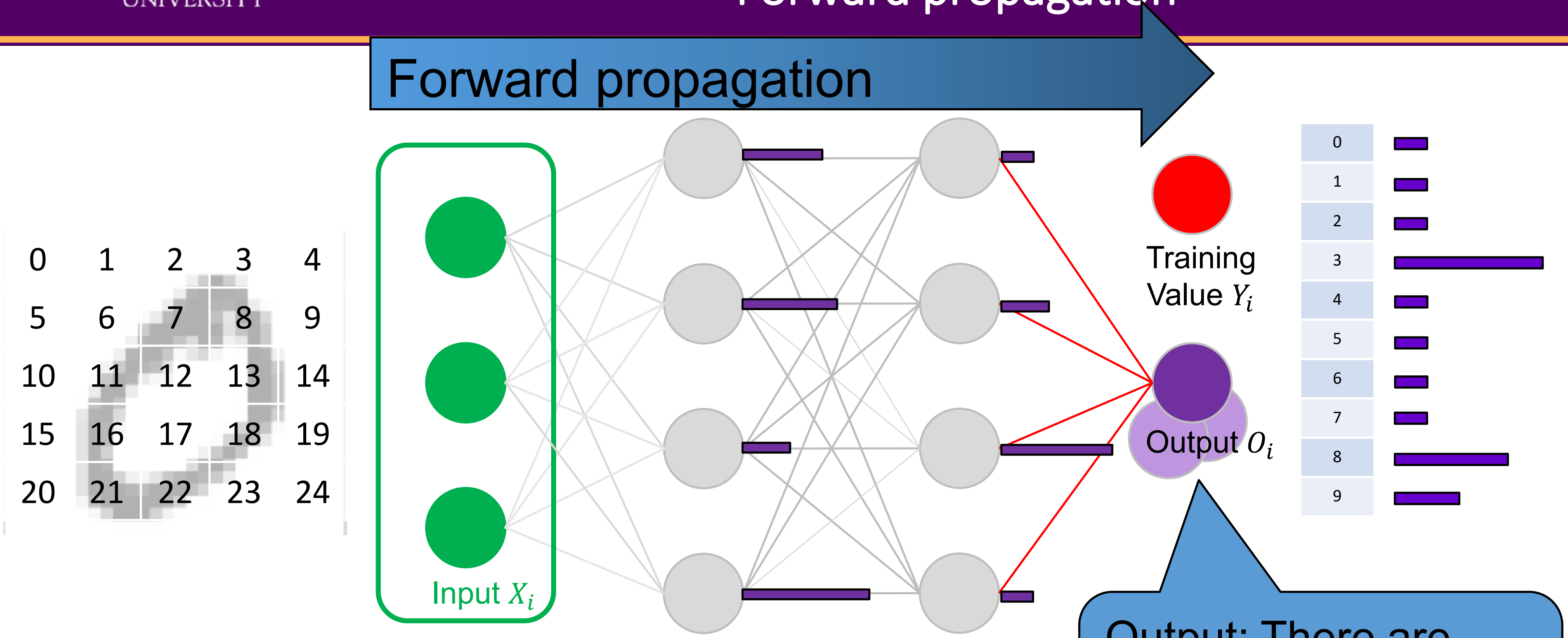
Forward propagation



Each activation is simply a stacked logistic regression.

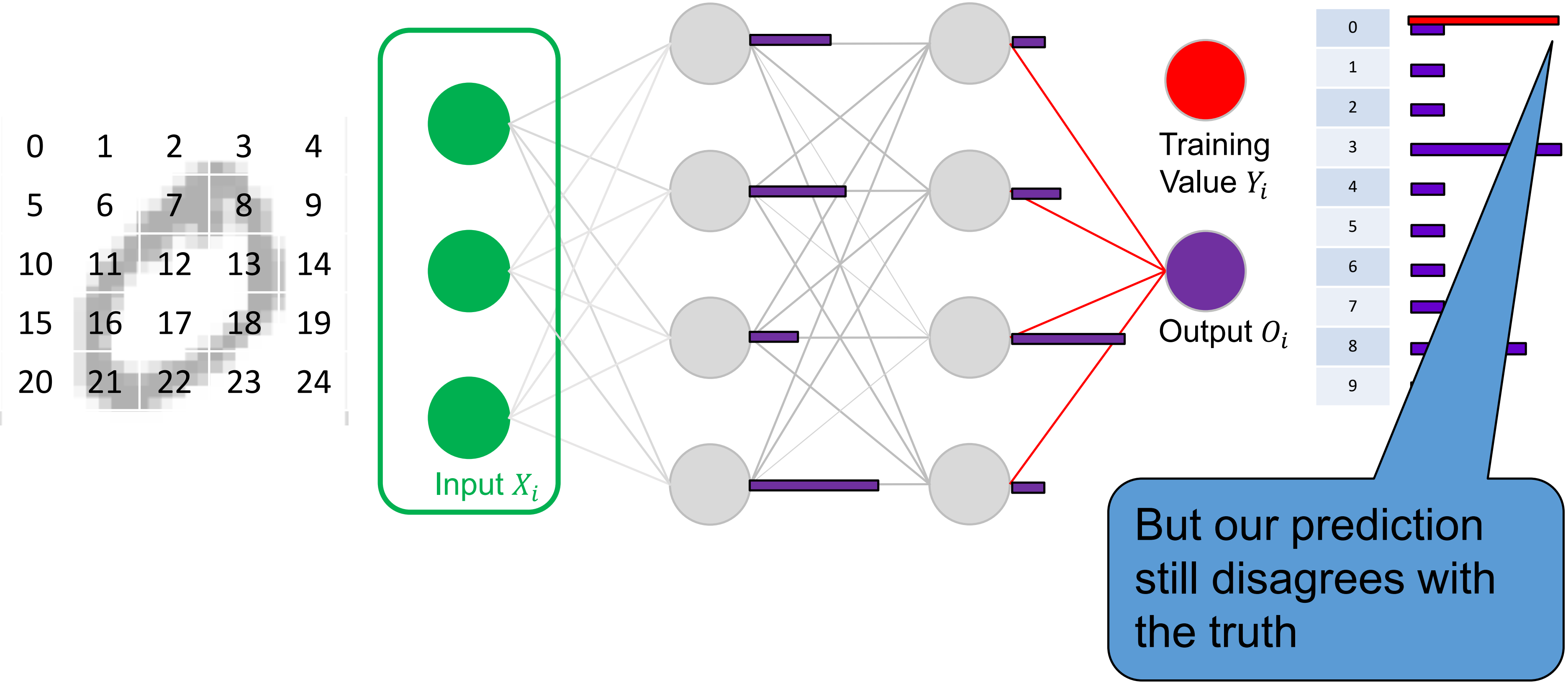
Bar length = level of activation

Forward propagation



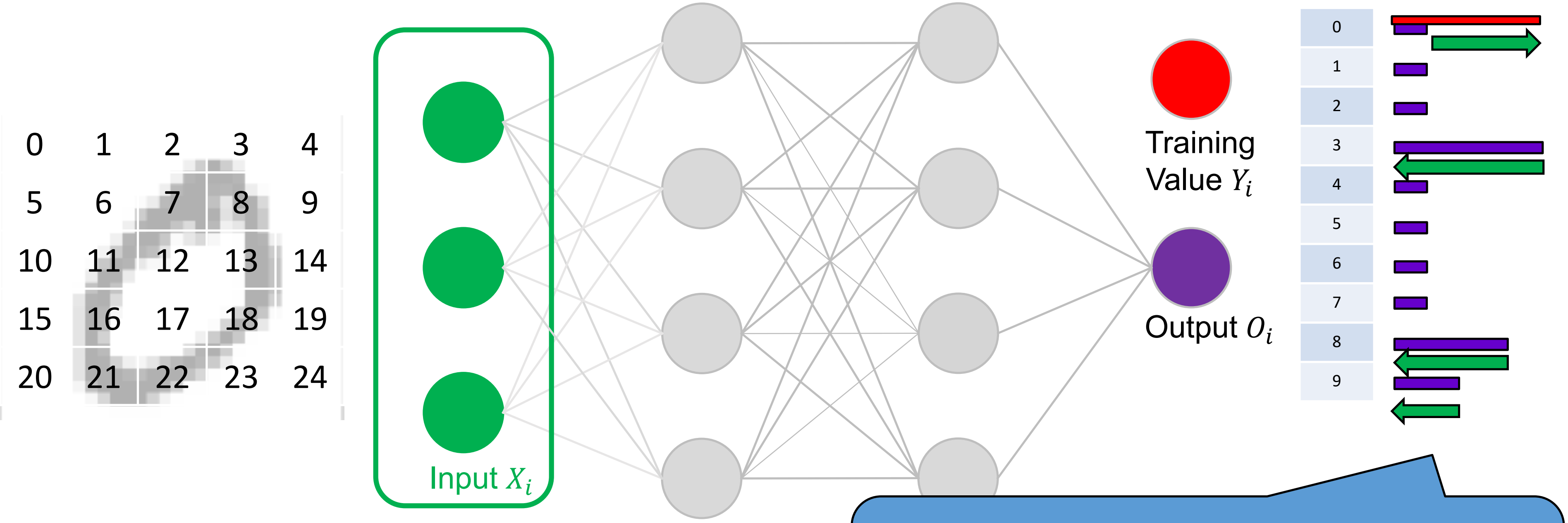
Output: There are actually 10 outputs, but we only show one

Forward propagation



But our prediction still disagrees with the truth

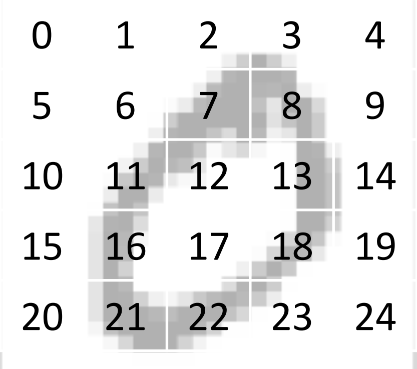
Backpropagation

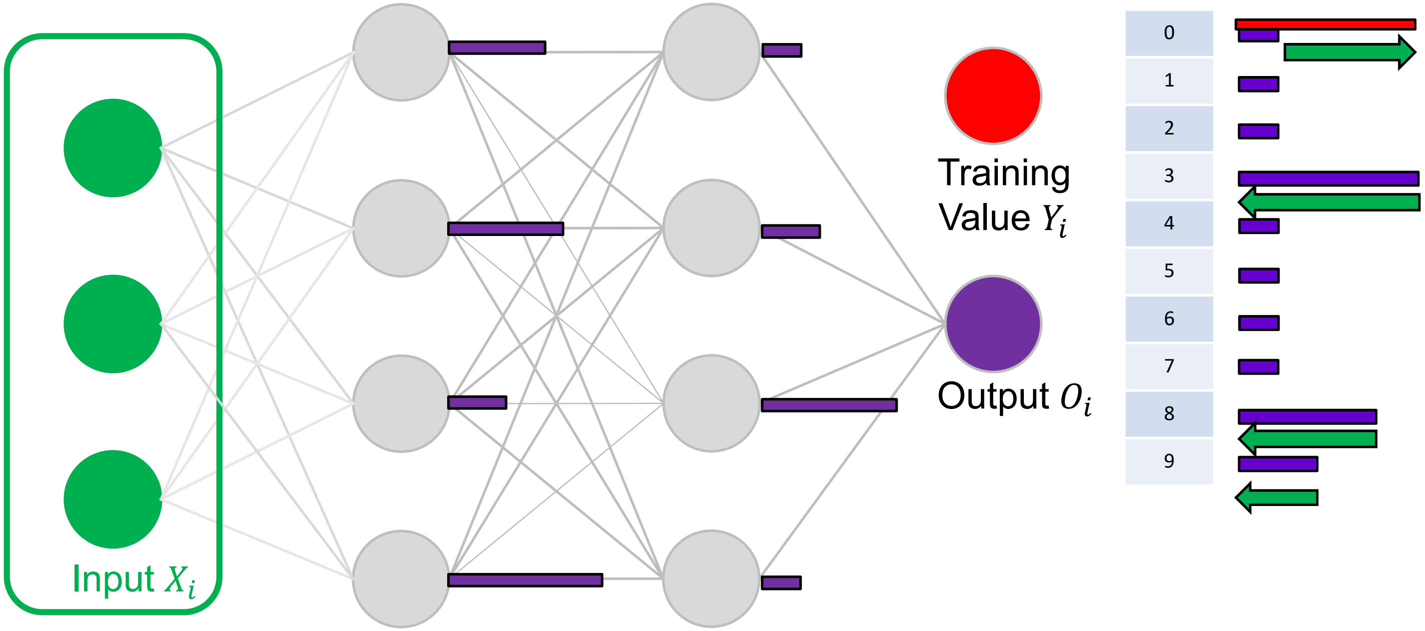


We must adjust the predictions using error derivatives → Backpropagation

Goal: Learn weights such that outputs agree with training data → Minimize $\sum_i (Y_i - O_i)^2$

Backpropagation





Recall the following activations in the forward pass

Backpropagation

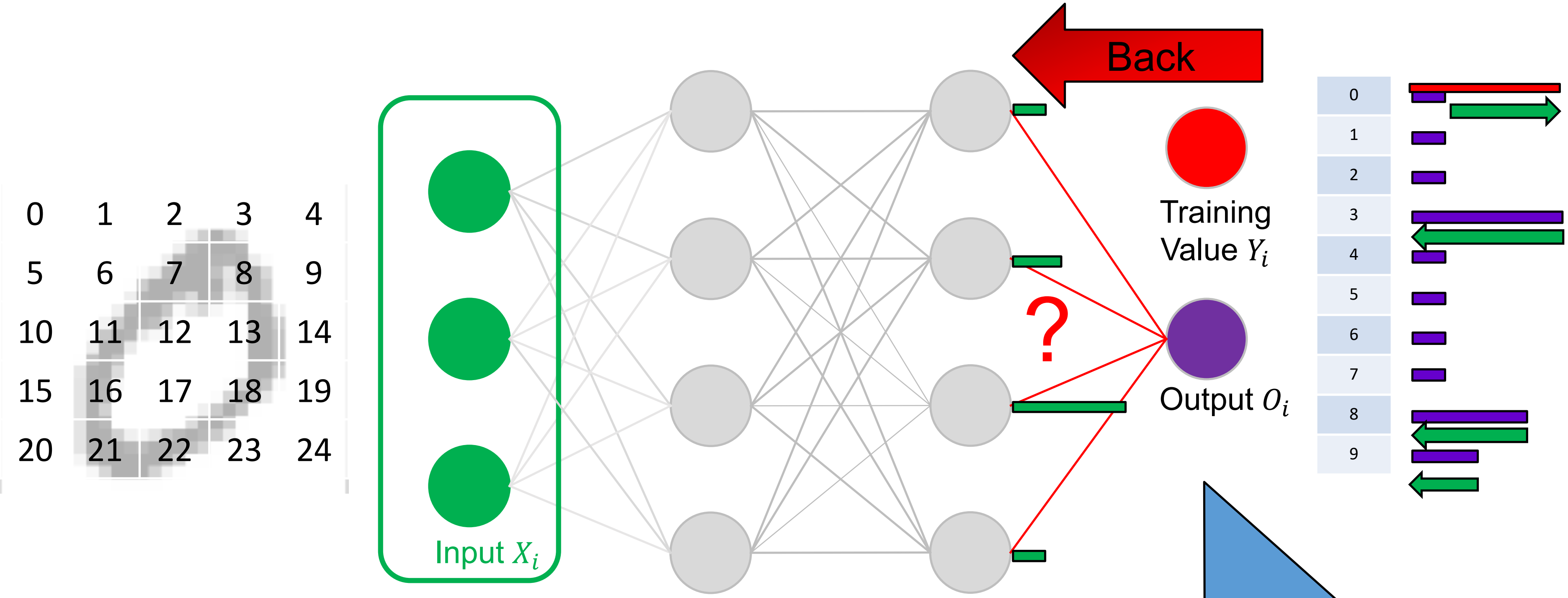


Figure out how to adjust input(green) weights(red) to match target activations(purple)

Backpropagation

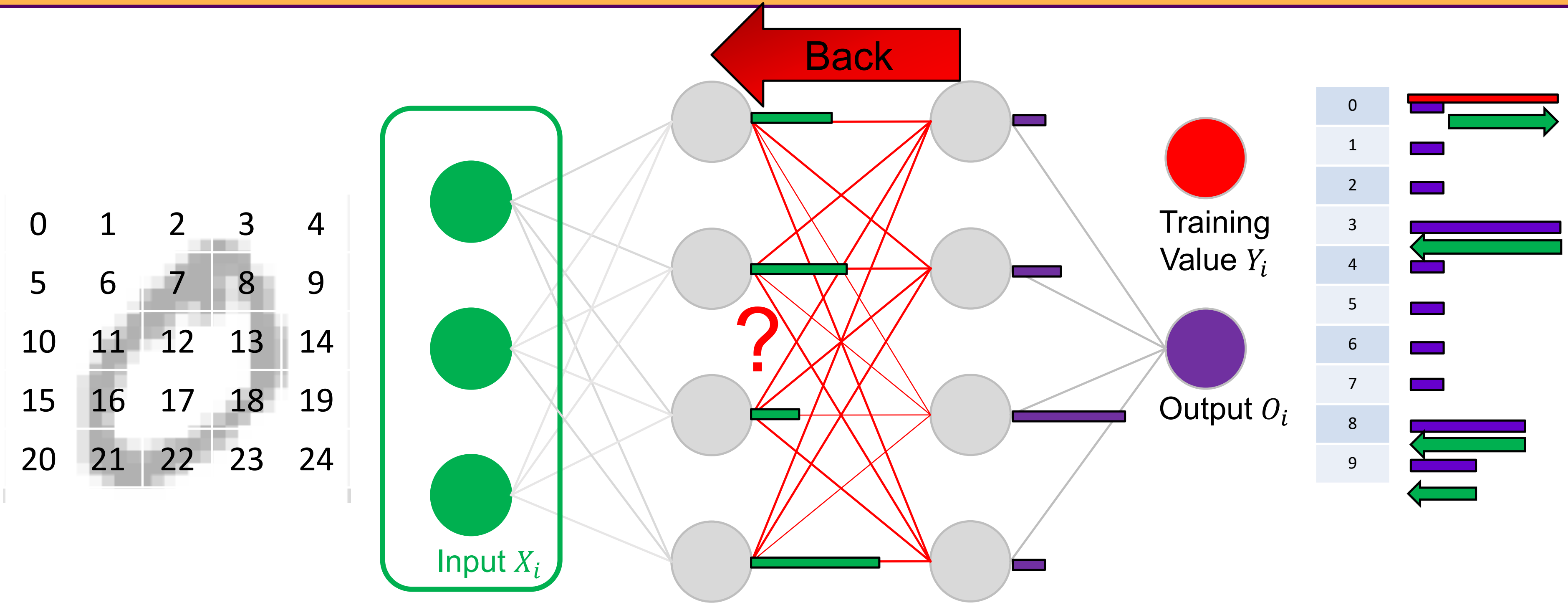


Figure out how to adjust input(green) weights(red) to match target activations(purple)

Backpropagation

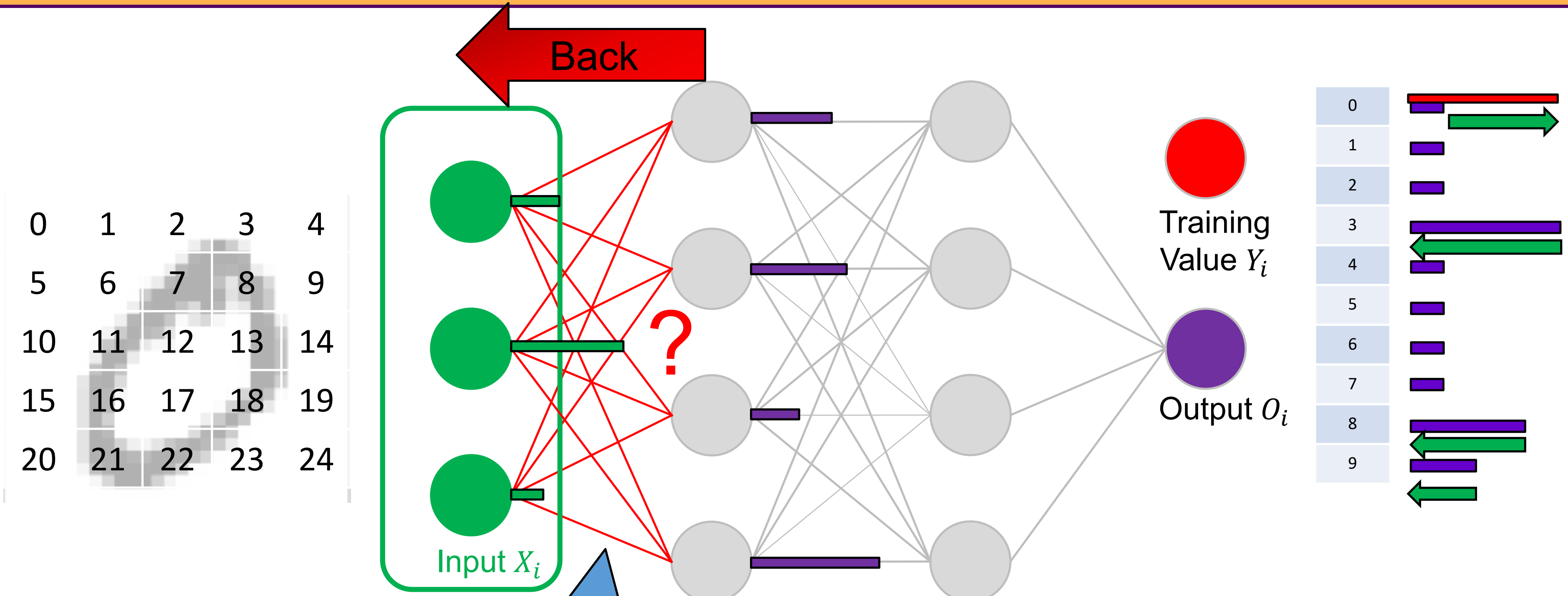
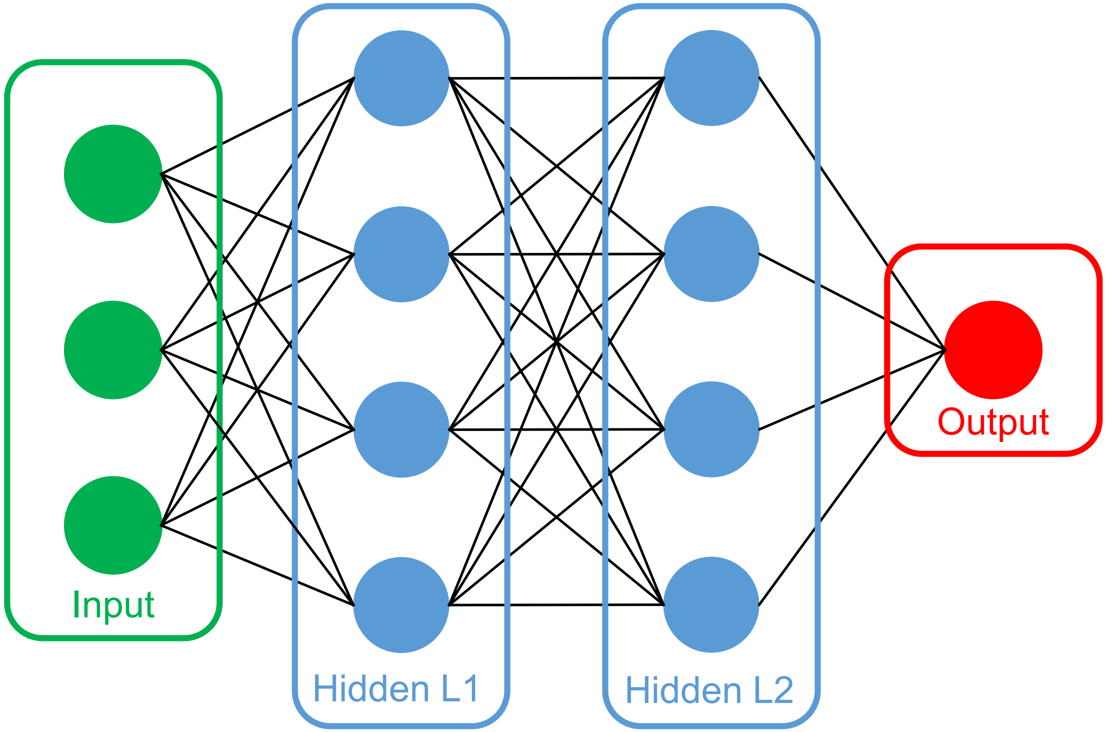
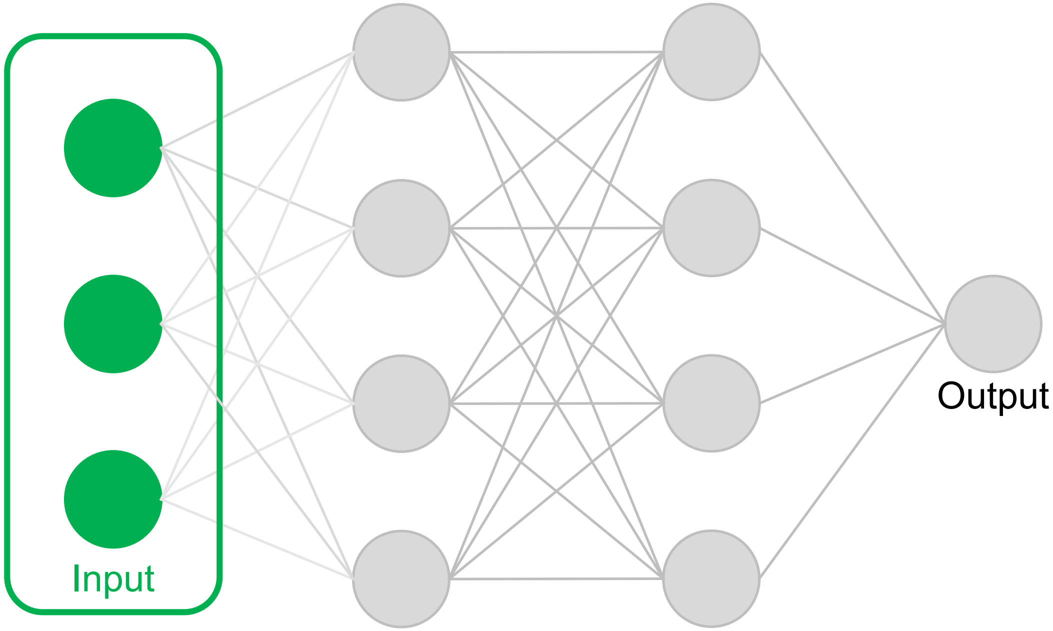


Figure out how to adjust input(green) weights(red) to match target activations(purple)

Feedforward networks

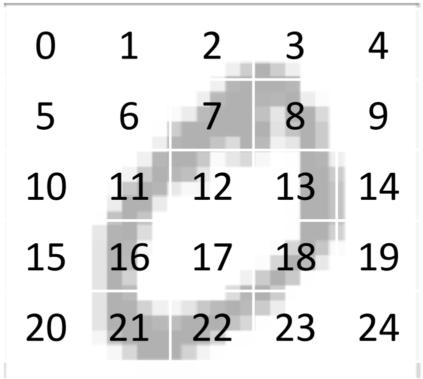


Feedforward networks



Each input neuron carries a signal
# neurons in input layer
= # inputs
= #pixels = 25

Recall: vectorizing inputs

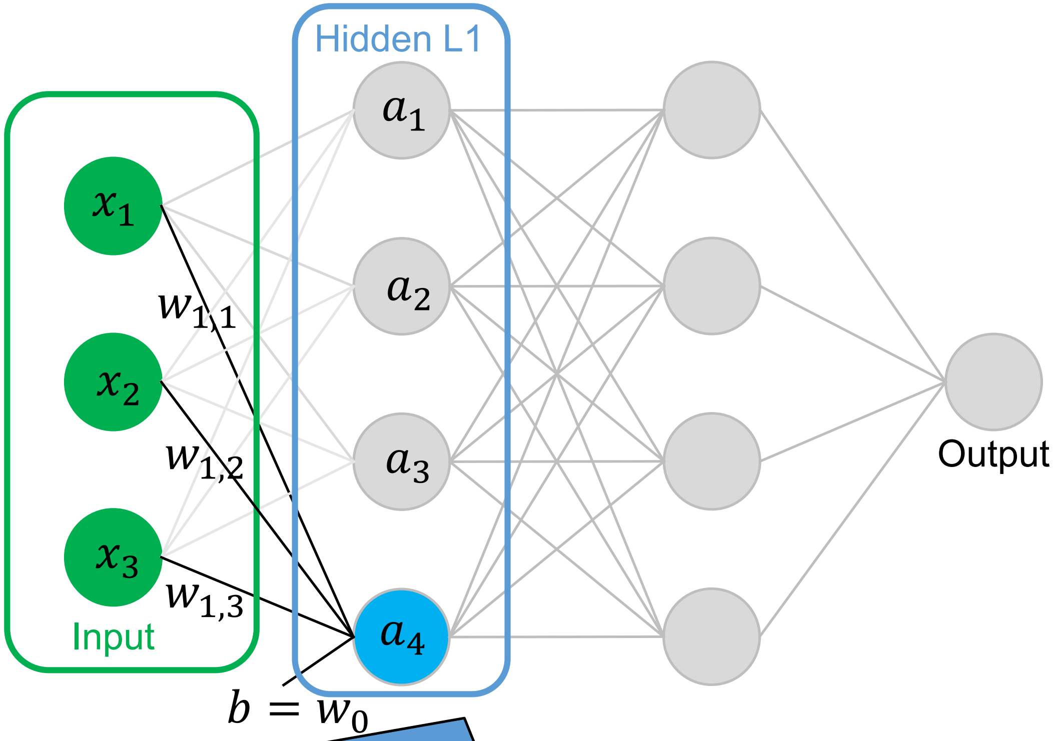


0	1	2	3	4
5	6	7	8	9
10	11	12	13	14
15	16	17	18	19
20	21	22	23	24

X1	X2	X3	…	…	Xn

Samples are column vectors

Feedforward networks



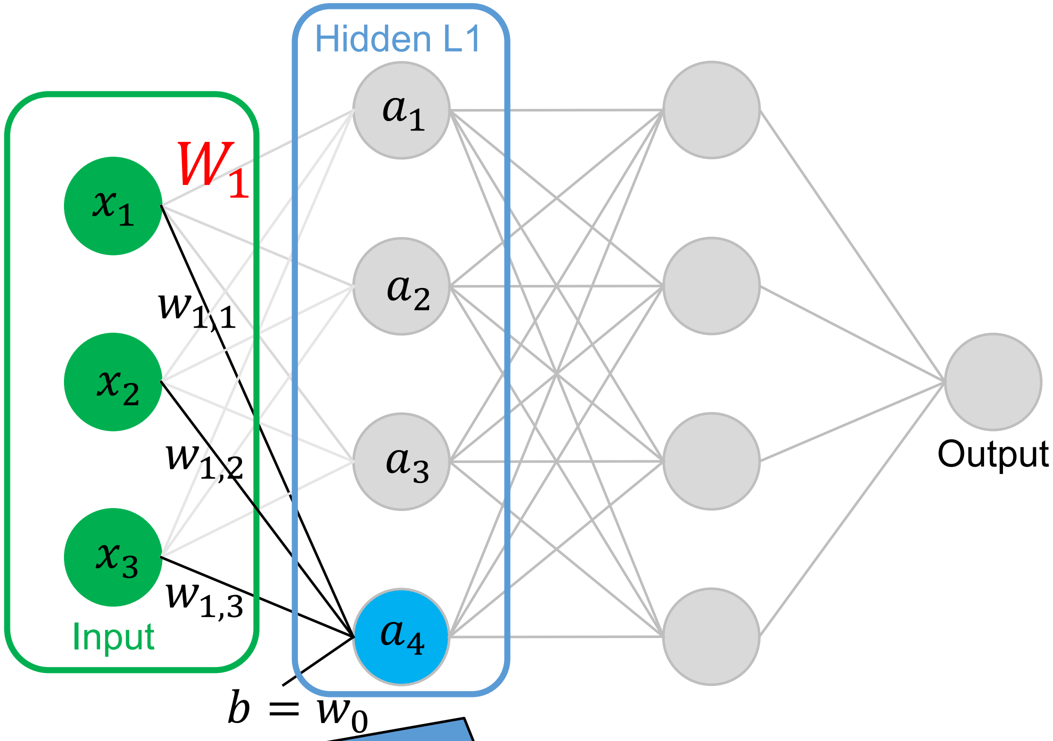
$$\sigma\left(\begin{bmatrix} w_{1,1} & w_{1,2} & w_{1,3} \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + w_0\right) = a_1$$

- Signals are propagated to the second layer
- Each neuron in the second layer is a logistic regression

$$\begin{bmatrix} w_1 \\ \dots \\ \dots \\ \dots \\ w_n \end{bmatrix}$$

Weights are row vectors

Feedforward networks



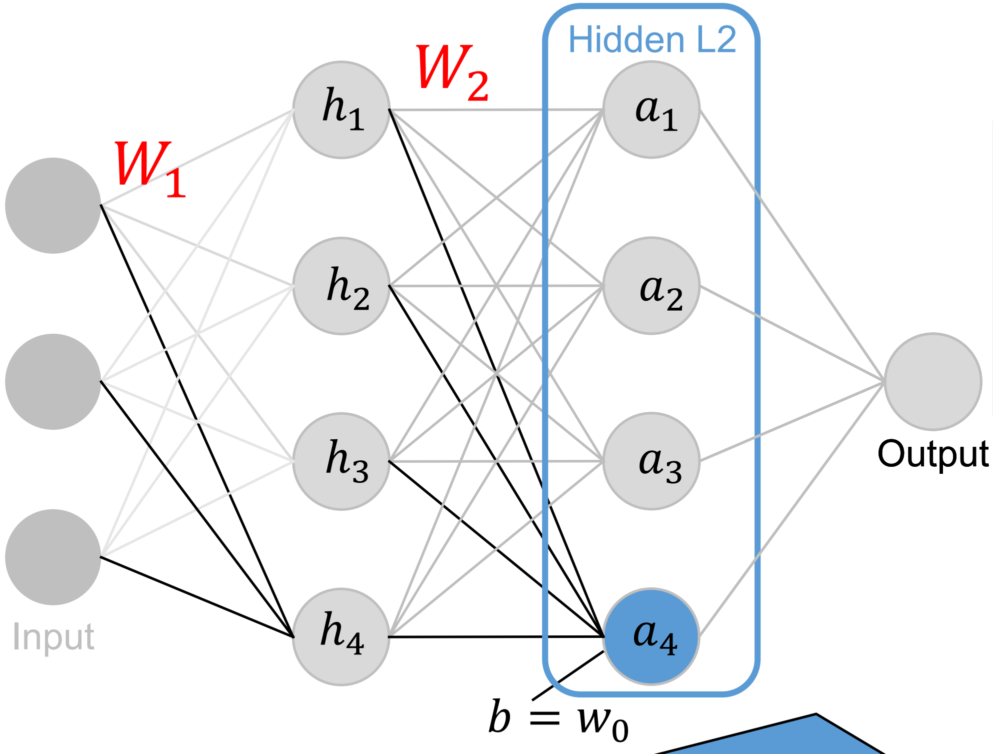
$$\sigma\left(\underbrace{\begin{bmatrix} w_{1,1} & w_{1,2} & w_{1,3} \\ w_{1,4} & w_{1,5} & w_{1,6} \\ w_{1,7} & w_{1,8} & w_{1,9} \\ w_{1,10} & w_{1,11} & w_{1,12} \end{bmatrix}}_{W_1}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + w_0\right) = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix}$$

```
def sigm(s):
    return 1.0/(1 + np.e**-s)
A1 = sigm(np.dot(W1, X))
```

- Signals are propagated to the second layer → Vectorize
- Each neuron in the second layer is a logistic regression

NORTHWESTERN UNIVERSITY

Feedforward networks

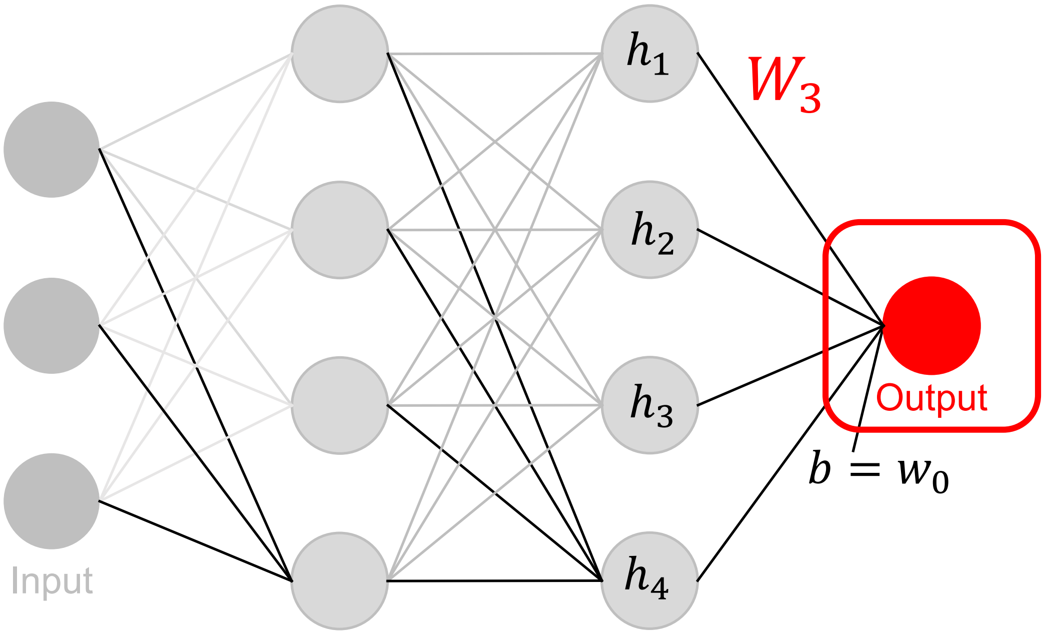


$$\sigma\left(W_2 \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \end{bmatrix} + w_0\right) = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix}$$

- Signals are propagated to the third layer
- Each neuron in the third layer is a logistic regression

```
def sigm(s):
    return 1.0/(1 + np.e**-s)
A1 = sigm(np.dot(W1, X))
A2 = sigm(np.dot(W2, A1))
```

Feedforward networks



$$\sigma\left(W_3 \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \end{bmatrix} + w_0 \right) = Out$$

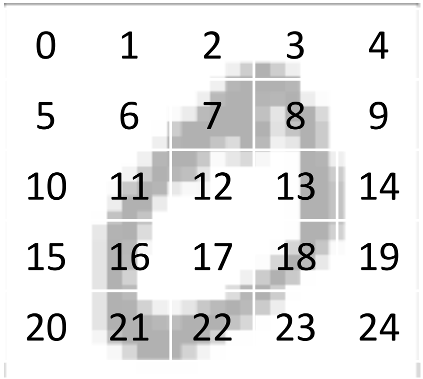
```
def sigm(s):
    return 1.0/(1 + np.e**-s)
A1 = sigm(np.dot(W1, X))
A2 = sigm(np.dot(W2, A1))
Out = sigm(np.dot(W3, A2))
```

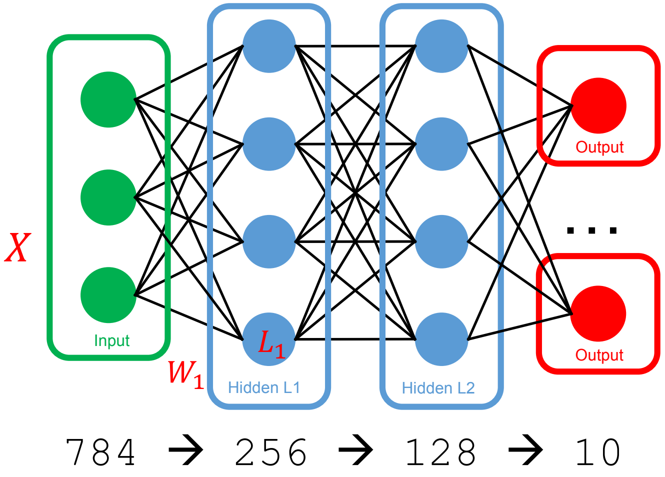
- Signals are propagated to the final layer
- The output is a logistic regression

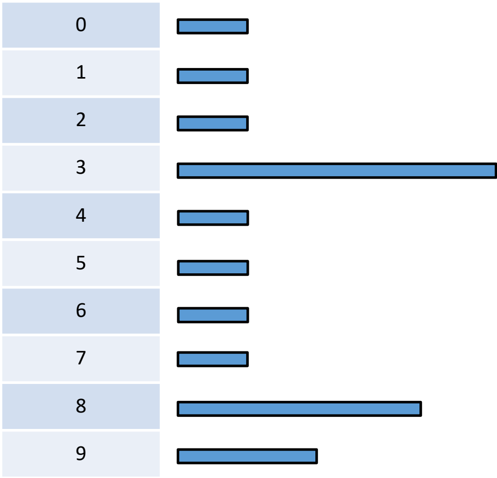
Making things fast

VECTORIZING FEEDFORWARD NETS

Vectorized forward pass







Output:
Prediction
vector Y of class

Setup code:

```
# 784 → 256 → 128 → 10
X = training_data    # 784 x 60000
T = target_classes # 10 x 60000

W1 = 2*rand(784, 256).T - 1
W2 = 2*rand(256, 128).T - 1
W3 = 2*rand(128, 10).T  - 1
```

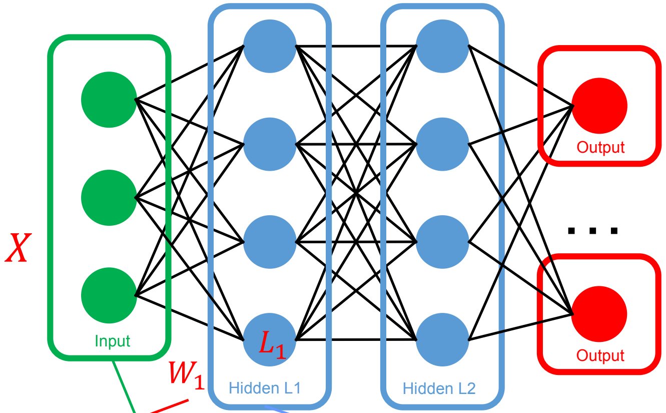
A.T means A transpose: $A^T_{ij} = A_{ji}$

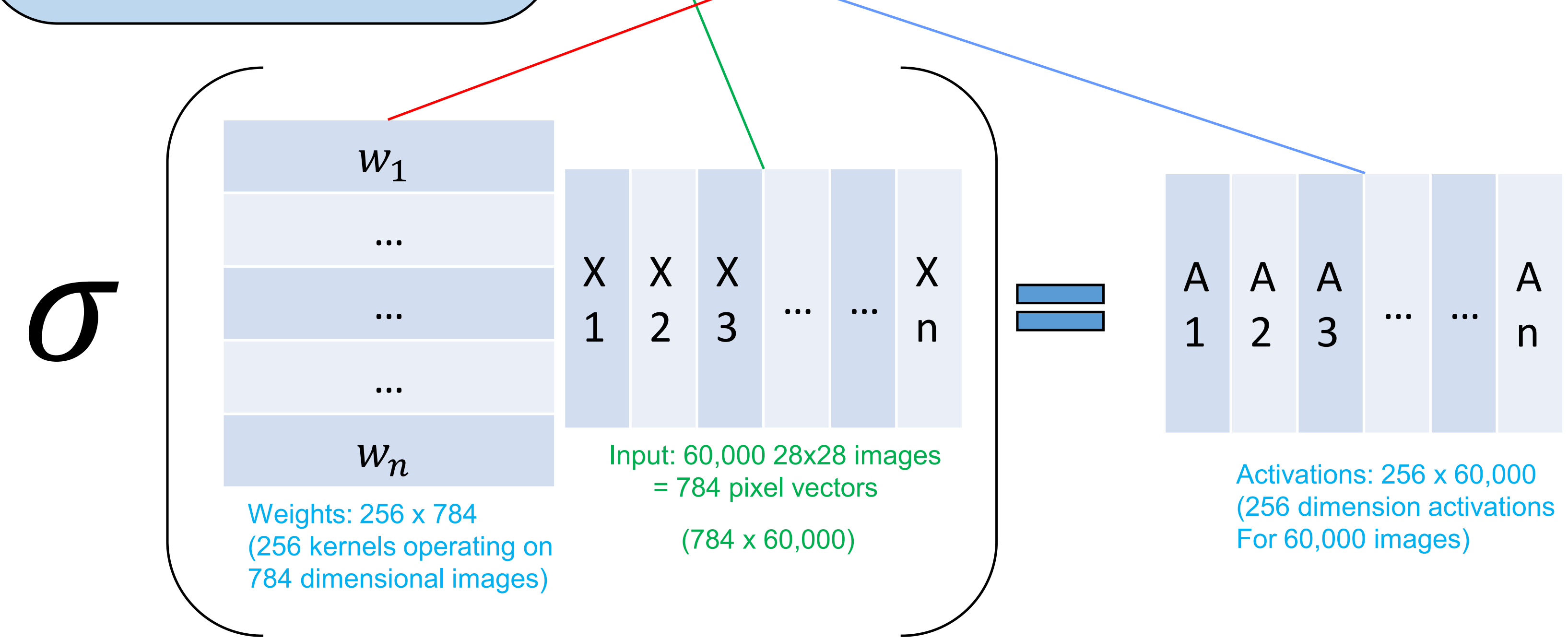
Don't confuse this with target T

Vectorized forward pass

Code:

```
L1 = sigmoid(W1.dot(X))
```





Weights: 256 x 784
(256 kernels operating on
784 dimensional images)

Input: 60,000 28x28 images
= 784 pixel vectors

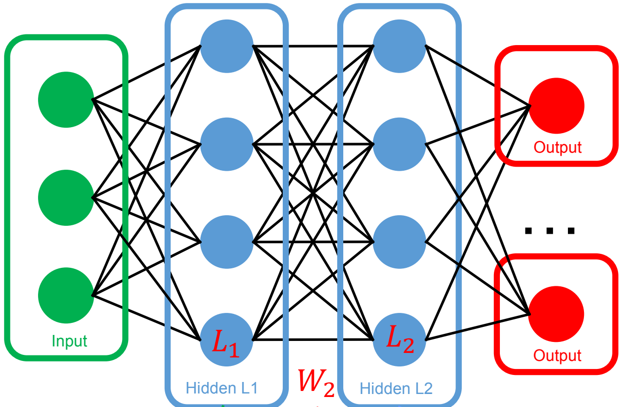
(784 x 60,000)

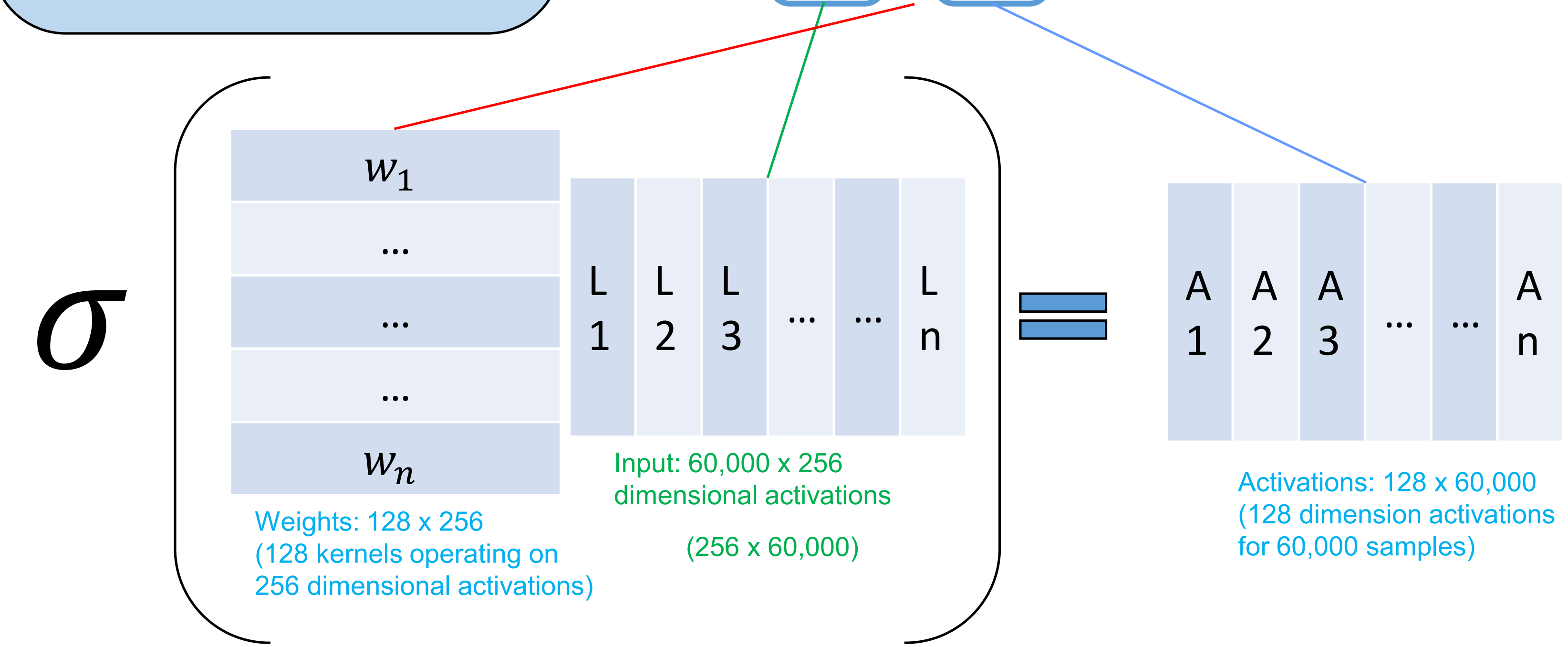
Activations: 256 x 60,000
(256 dimension activations
For 60,000 images)

Vectorized forward pass

Code:

```
L1 = sigmoid(W1.dot(X))
L2 = sigmoid(W2.dot(L1))
```





Weights: 128 x 256
(128 kernels operating on
256 dimensional activations)

Input: 60,000 x 256
dimensional activations

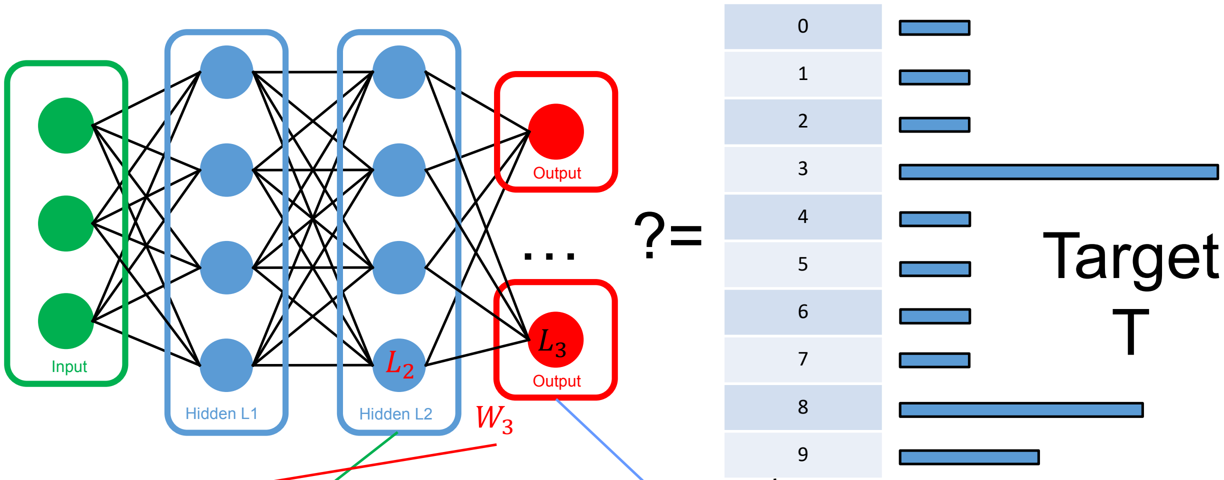
(256 x 60,000)

Activations: 128 x 60,000
(128 dimension activations
for 60,000 samples)

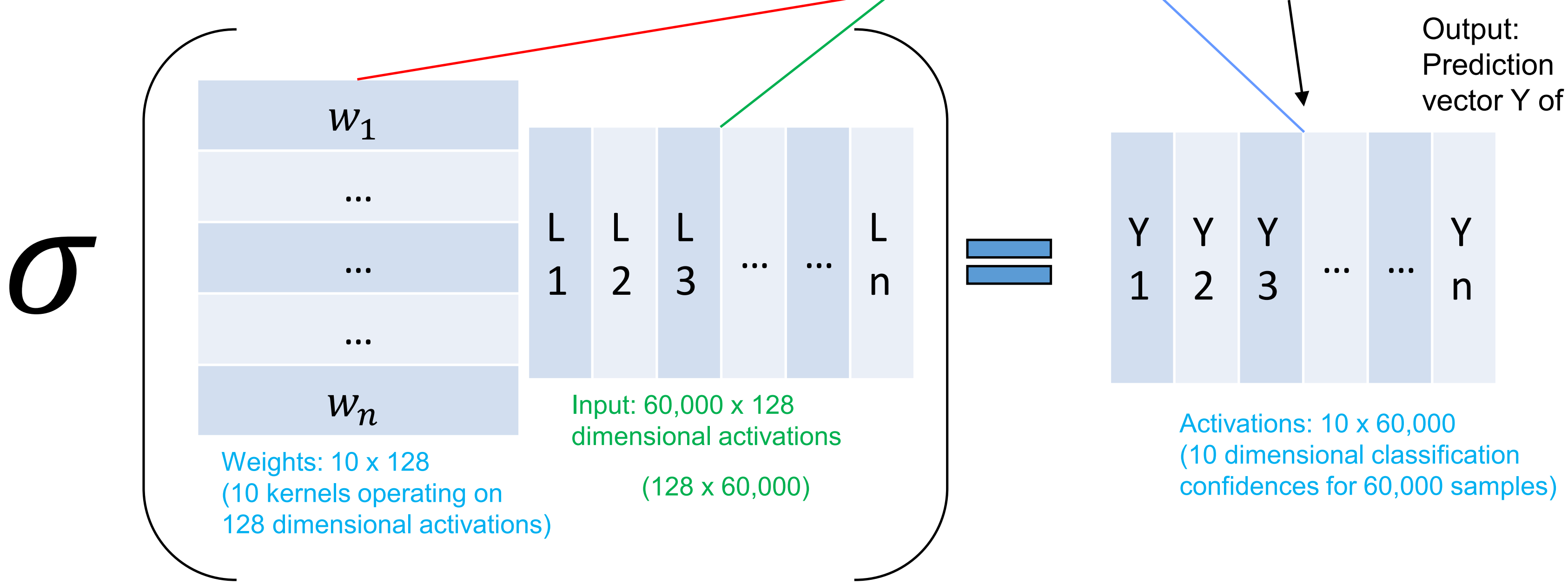
Vectorized forward pass

Code:

```
L1 = sigmoid(W1.dot(X))
L2 = sigmoid(W2.dot(L1))
L3 = sigmoid(W3.dot(L2))
```



Output:
Prediction
vector Y of class



Weights: 10 x 128
(10 kernels operating on
128 dimensional activations)

Input: 60,000 x 128
dimensional activations

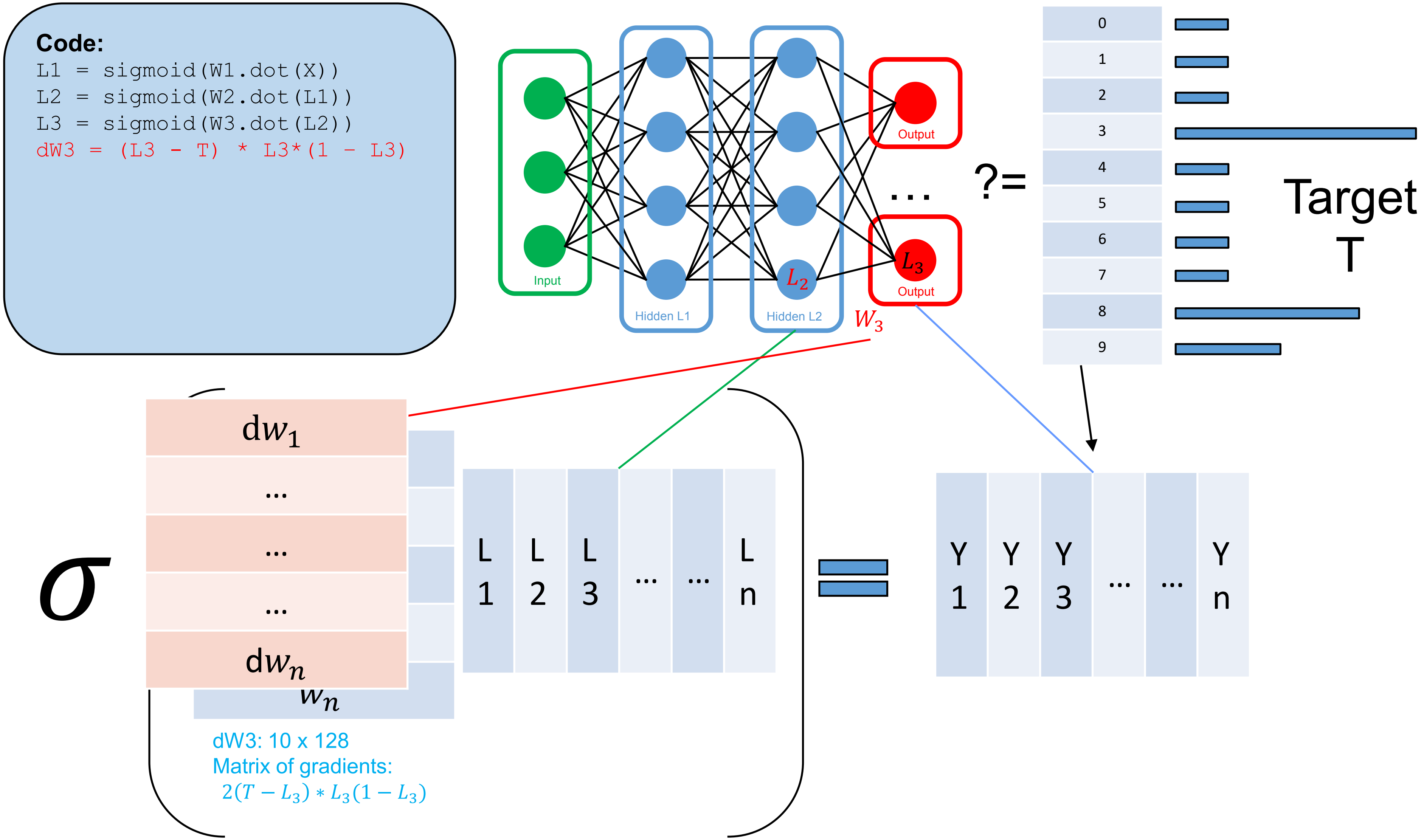
(128 x 60,000)

Activations: 10 x 60,000
(10 dimensional classification
confidences for 60,000 samples)

Vectorized backward pass

Code:

```
L1 = sigmoid(W1.dot(X))
L2 = sigmoid(W2.dot(L1))
L3 = sigmoid(W3.dot(L2))
dW3 = (L3 - T) * L3*(1 - L3)
```



dW3: 10 x 128

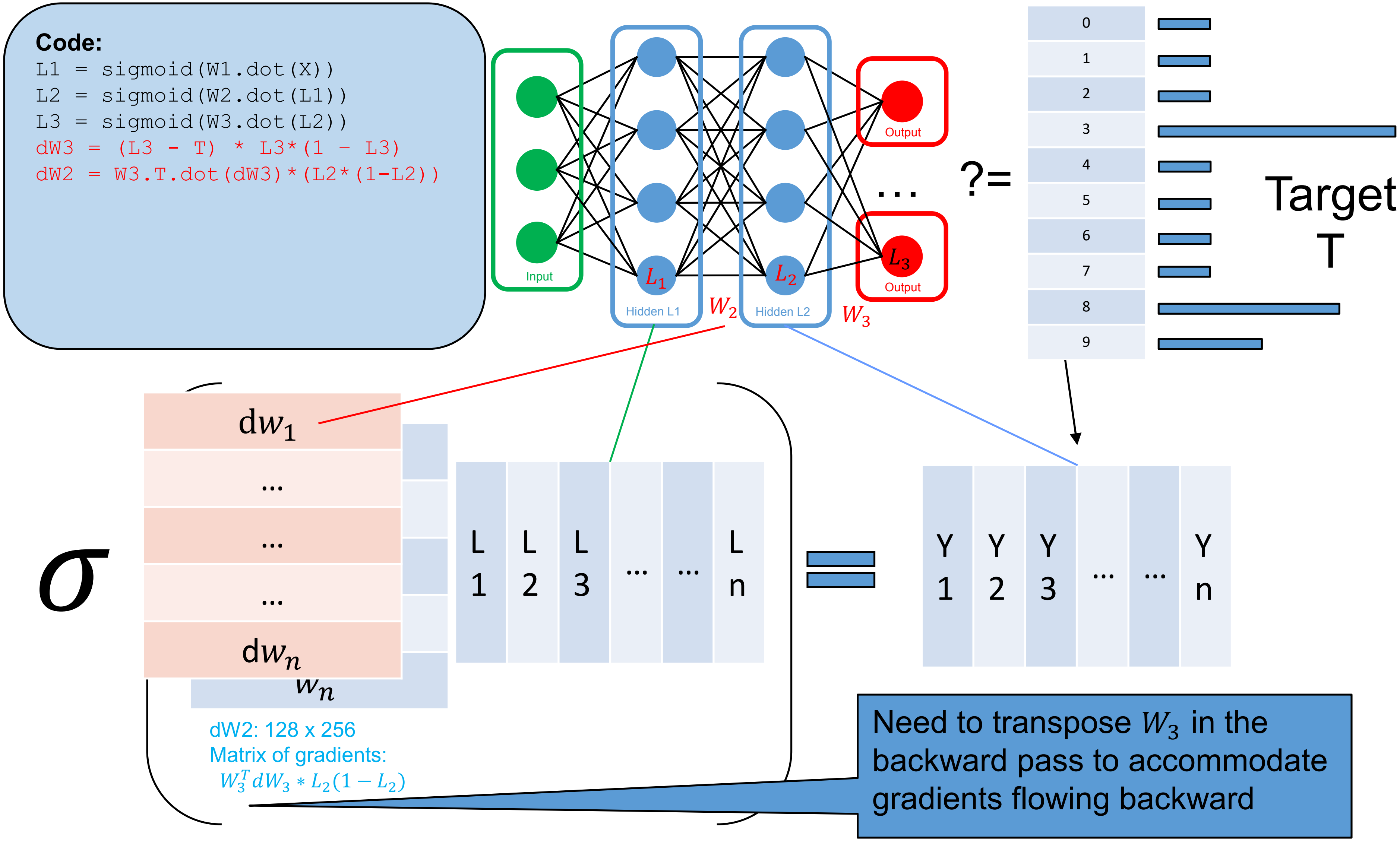
Matrix of gradients:

$2(T - L_3) * L_3(1 - L_3)$

Vectorized backward pass

Code:

```
L1  = sigmoid(W1.dot(X))
L2  = sigmoid(W2.dot(L1))
L3  = sigmoid(W3.dot(L2))
dW3  = (L3 - T) * L3*(1 - L3)
dW2  = W3.T.dot(dW3)*(L2*(1-L2))
```



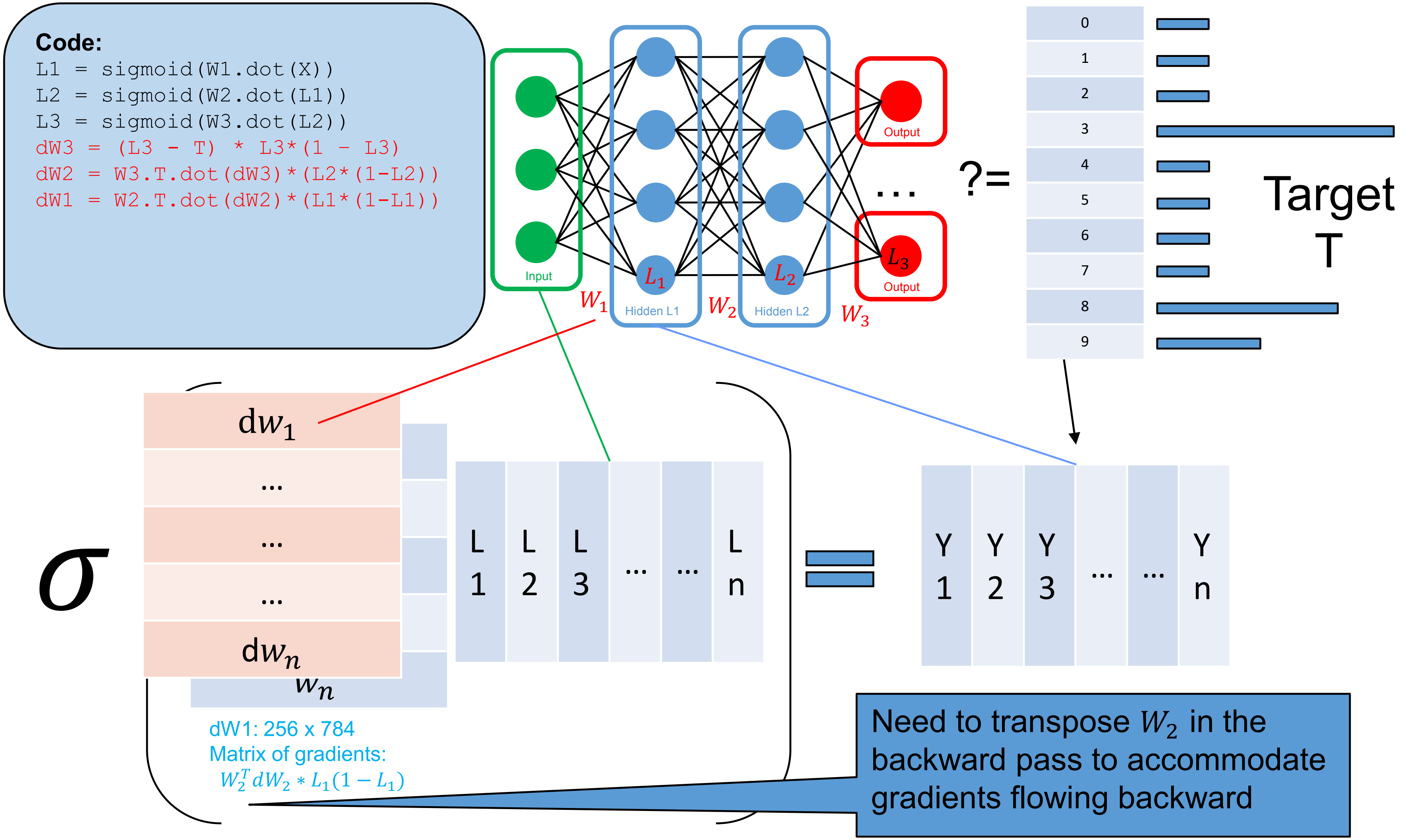
dW2: 128 x 256
Matrix of gradients:
$W_3^T dW_3 * L_2(1 - L_2)$

Need to transpose W_3 in the backward pass to accommodate gradients flowing backward

Vectorized backward pass

Code:

```
L1 = sigmoid(W1.dot(X))
L2 = sigmoid(W2.dot(L1))
L3 = sigmoid(W3.dot(L2))
dW3 = (L3 - T) * L3*(1 - L3)
dW2 = W3.T.dot(dW3)*(L2*(1-L2))
dW1 = W2.T.dot(dW2)*(L1*(1-L1))
```



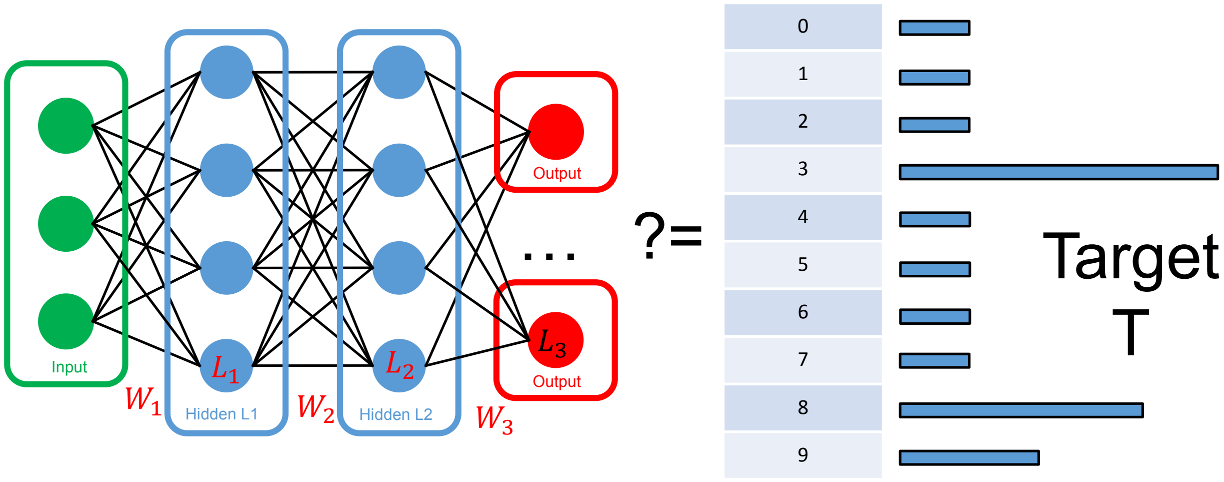
dW1: 256 x 784
Matrix of gradients:
$W_2^T dW_2 * L_1(1 - L_1)$

Need to transpose W_2 in the backward pass to accommodate gradients flowing backward

Vectorized weight updates

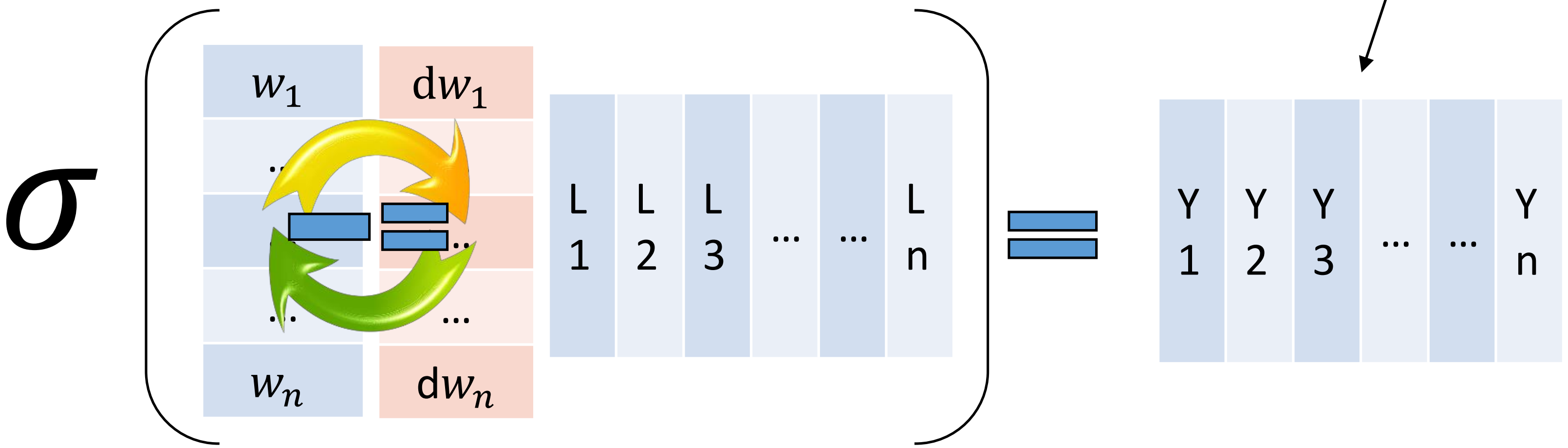
Code:

```
L1  = sigmoid(W1.dot(X))
L2  = sigmoid(W2.dot(L1))
L3  = sigmoid(W3.dot(L2))
dW3  = (L3 - T) * L3*(1 - L3)
dW2  = W3.T.dot(dW3)*(L2*(1-L2))
dW1  = W2.T.dot(dW2)*(L1*(1-L1))
W3  -= lr*np.dot(dW3, L2.T)
W2  -= lr*np.dot(dW2, L1.T)
W1  -= lr*np.dot(dW1, X.T)
```



$$\boxed{2(\sigma - t)} * \boxed{\sigma(1-\sigma)} * \boxed{x_1}$$

MSE Sigmoid Dot product



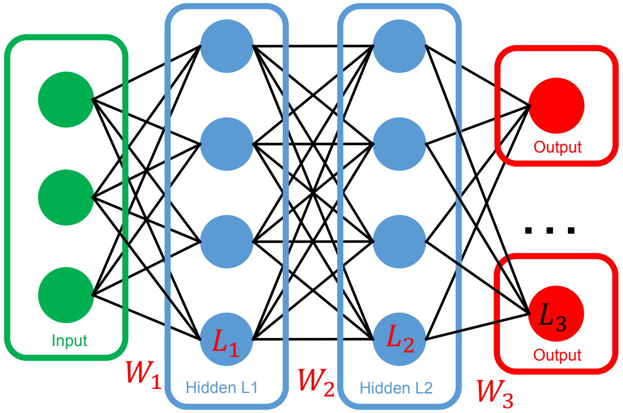
Vectorized weight updates

Code:

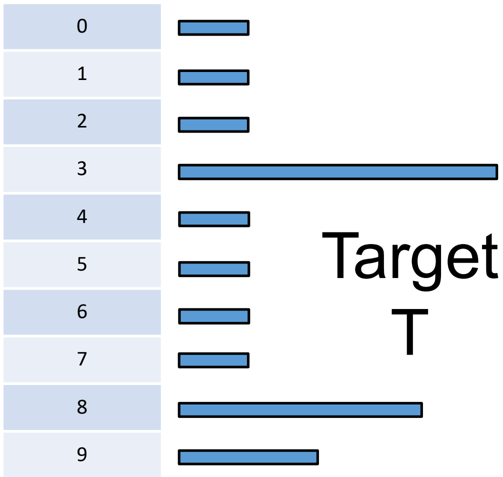
```
L1 = sigmoid(W1.dot(X))
L2 = sigmoid(W2.dot(L1))
L3 = sigmoid(W3.dot(L2))
dW3 = (L3 - T) * L3*(1 - L3)
dW2 = W3.T.dot(dW3)*(L2*(1-L2))
dW1 = W2.T.dot(dW2)*(L1*(1-L1))
W3 -= lr*np.dot(dW3, L2.T)
W2 -= lr*np.dot(dW2, L1.T)
W1 -= lr*np.dot(dW1, X.T)
```

$$\boxed{2(\sigma - t)} * \boxed{\sigma(1-\sigma)} * \boxed{x_1}$$

MSE Sigmoid Dot pro

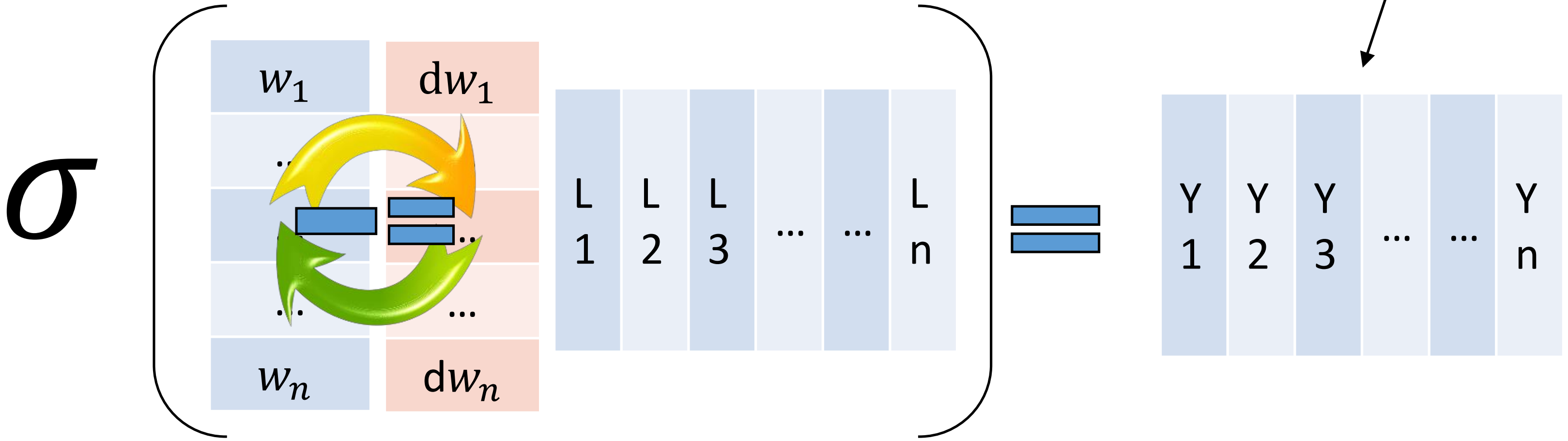


?=



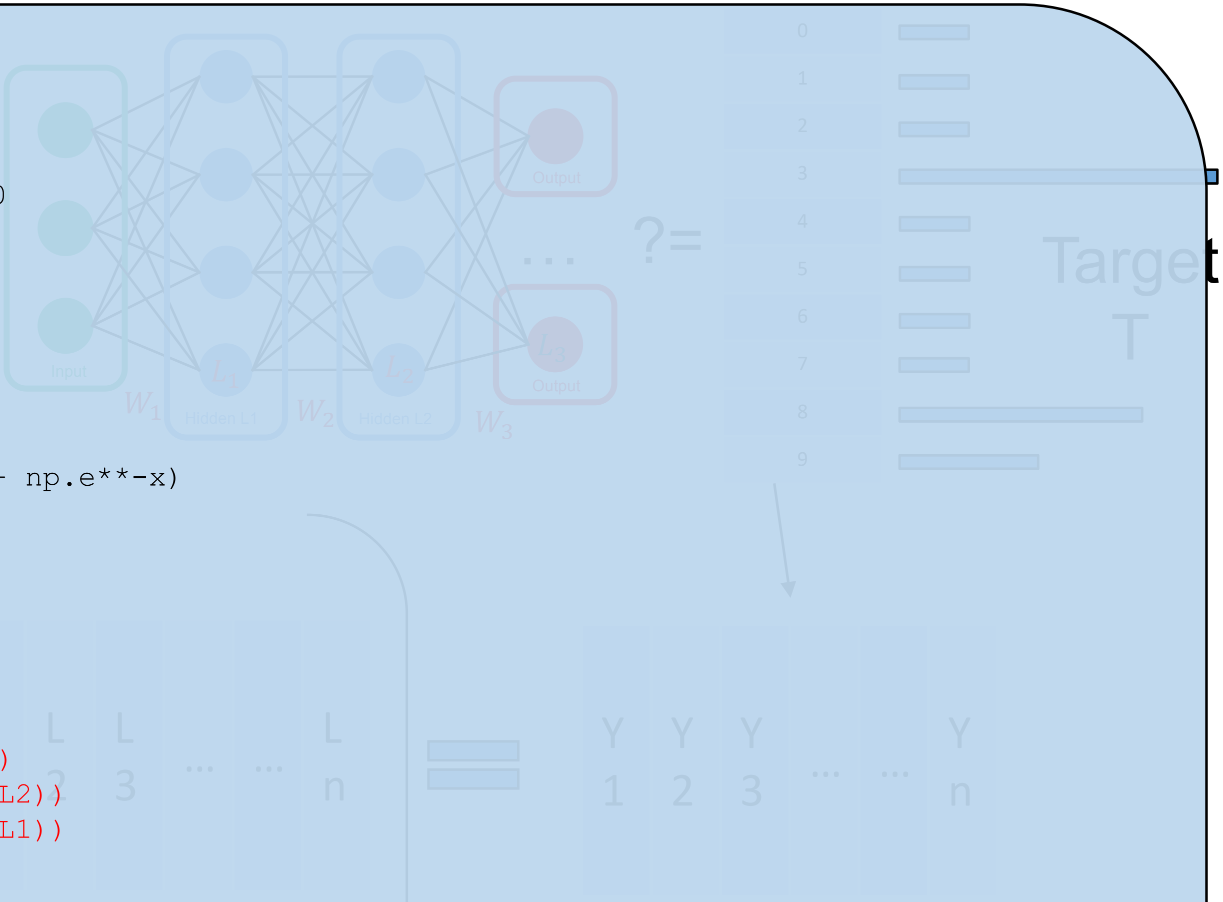
Target
T

NB: Factor of 2 in MSE is absorbed in learning rate



NORTHWESTERN UNIVERSITY

Vectorized weight updates

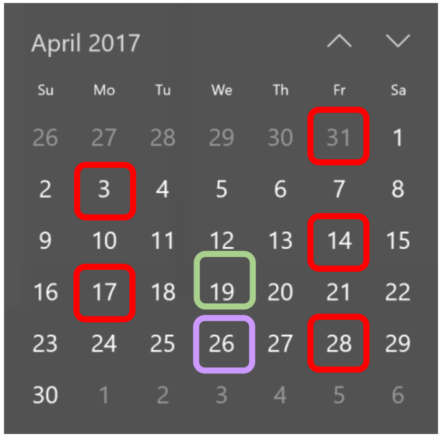


Code:

```
# Setup
# 784 → 256 → 128 → 10
X = training_data   # 784 x 60000
T = target_classes # 10 x 60000

W1 = 2*rand(784, 256).T - 1
W2 = 2*rand(256, 128).T - 1
W3 = 2*rand(128, 10).T  - 1

lr = 1e-5
def sigmoid(x): return 1.0/(1.0 + np.e**-x)
for i in range(5000):
     # Forward pass
     L1 = sigmoid(W1.dot(X))
     L2 = sigmoid(W2.dot(L1))
     L3 = sigmoid(W3.dot(L2))

     # Backward pass
     dW3 = (L3 - T) * L3*(1 - L3)
     dW2 = W3.T.dot(dW3)*(L2*(1-L2))
     dW1 = W2.T.dot(dW2)*(L1*(1-L1))

     # Update
     W3 -= lr*np.dot(dW3, L2.T)
     W2 -= lr*np.dot(dW2, L1.T)
     W1 -= lr*np.dot(dW1, X.T)

     print("[%04d] MSE Loss: %0.3f” % (i, np.sum((L3 - T)**2)/len(T.T)))
```

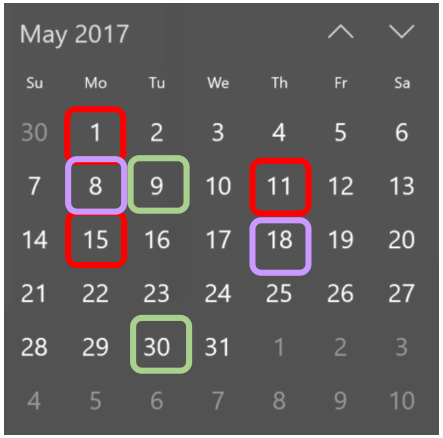
Assignment schedule

- Assignment 1: Due April 26th
  - Basic NN concepts
  - Optimization
- Assignment 2 (Group): Due May 18th
  - Computer Vision
  - Preparation for project
- Mini Quiz 1: May 8th: Basic NN, optimization and generalization multiple choice
- Mini Quiz 2: June 9th (2nd half): Generative, recurrent and applications multiple choice



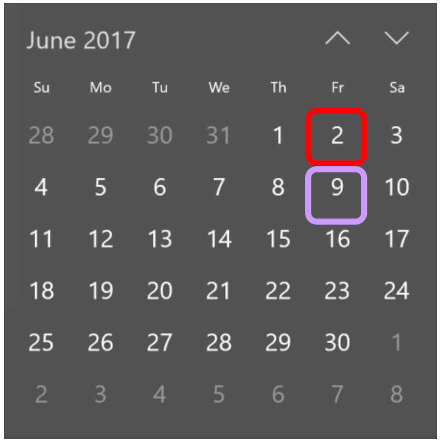
April 2017

Su	Mo	Tu	We	Th	Fr	Sa
26	27	28	29	30	31	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	1	2	3	4	5	6



May 2017

Su	Mo	Tu	We	Th	Fr	Sa
30	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31	1	2	3
4	5	6	7	8	9	10



June 2017

Su	Mo	Tu	We	Th	Fr	Sa
28	29	30	31	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	1
2	3	4	5	6	7	8

Projects

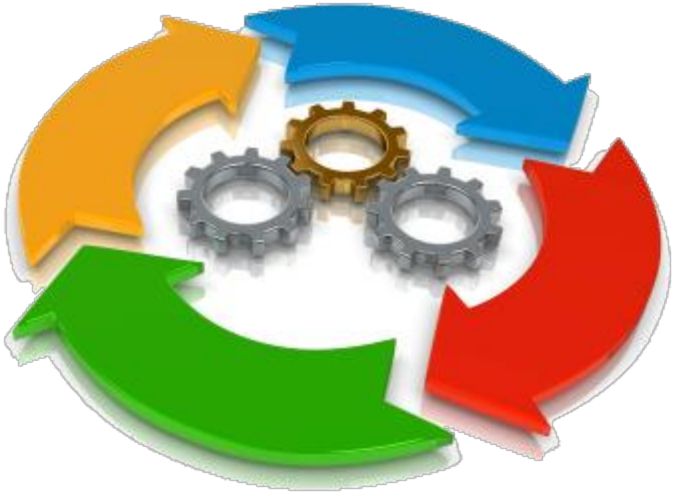
- Project group size
- Poster invitation
- Project proposal

Homework

- HW1

Next time

- Next time:
  - Advanced optimization techniques



- Reading: http://neuralnetworksanddeeplearning.com/chap3.html

New stuff

- Genetic algorithms to design NN: https://en.wikipedia.org/wiki/Neuroevolution_of_augmenting_topologies
- Overview of various NN: https://culurciello.github.io/tech/2016/06/04/nets.html