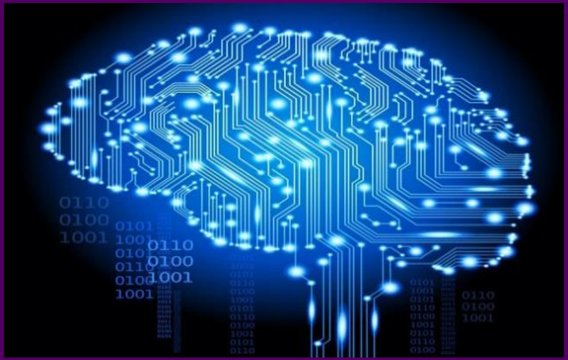
Deep Learning
MSiA 490-30



Theory and Applications

NORTHWESTERN
UNIVERSITY

NORTHWESTERN UNIVERSITY

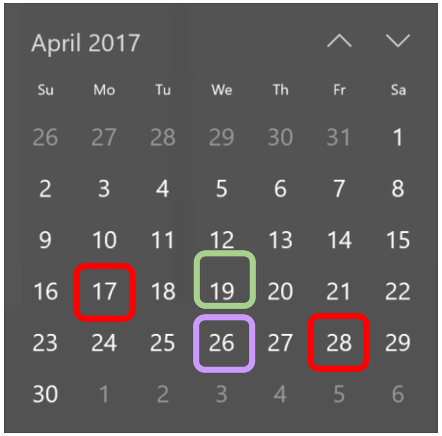
Feedback: News and Ideas

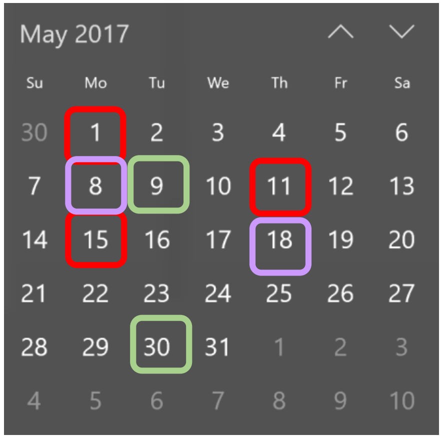
~5-10 min to share ideas/news

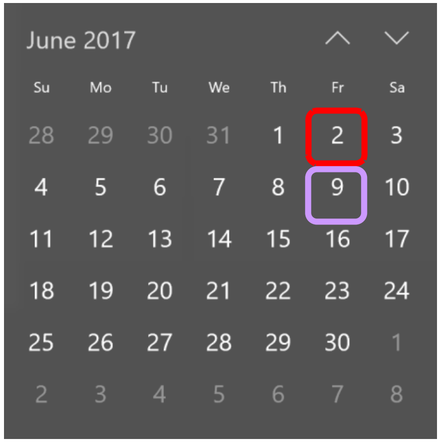
NORTHWESTERN UNIVERSITY

Assignment schedule

- Assignment 1: Due April 26th
  - Basic NN concepts
  - Optimization
  - Discuss with 1 partner, submit individually
- 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





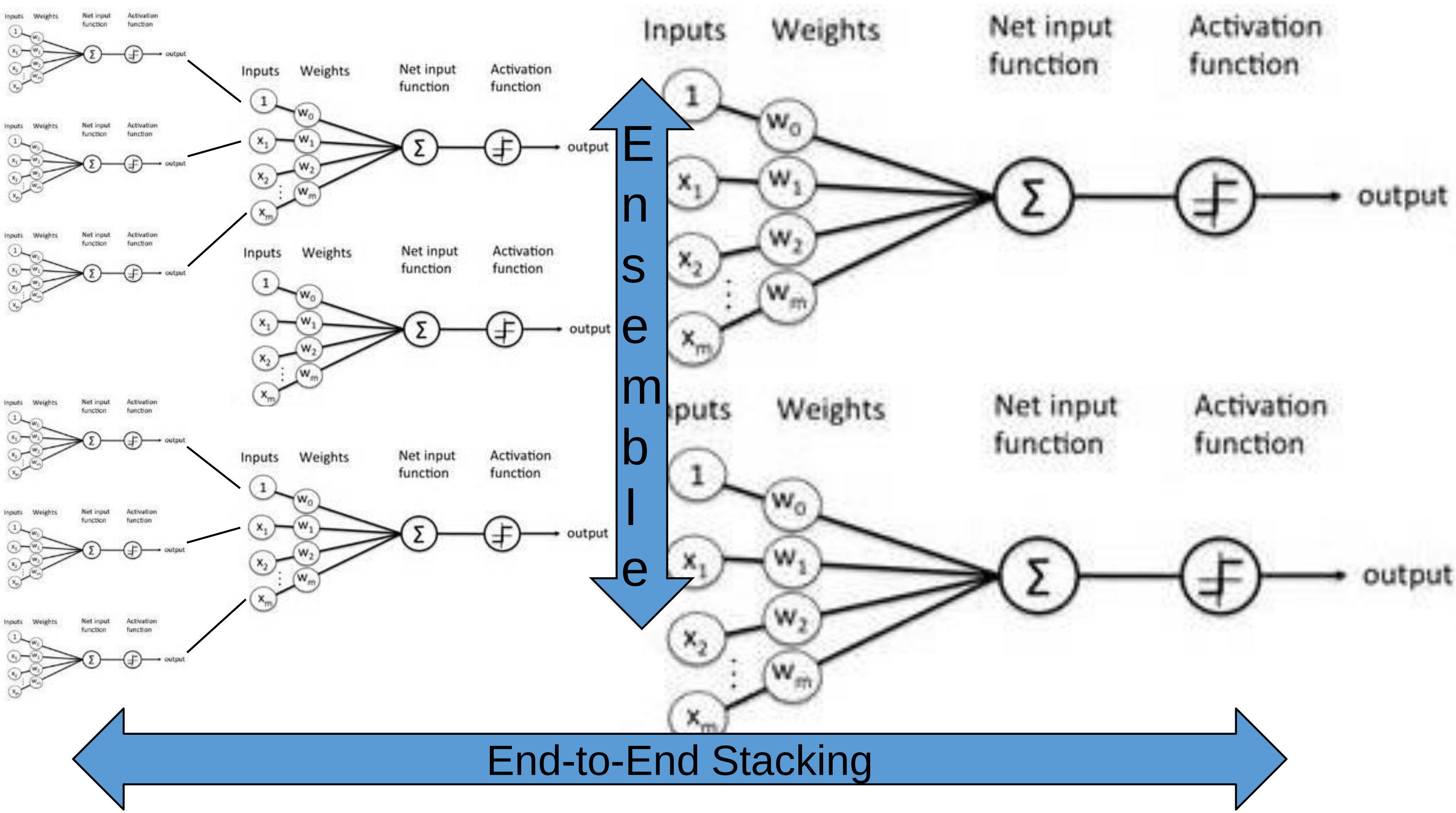


NORTHWESTERN UNIVERSITY

Last time

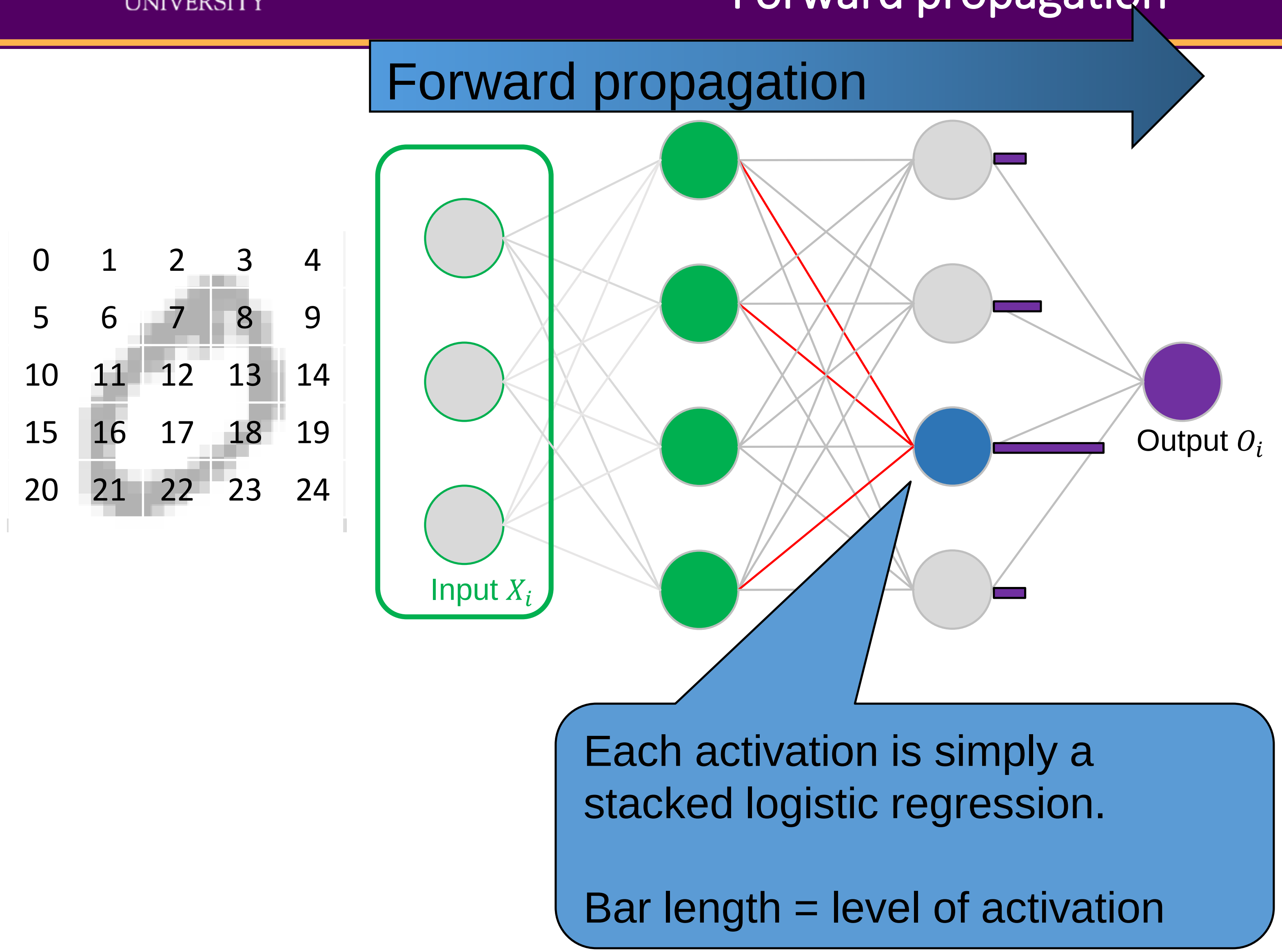
- Introduced multilayer nets
  - Apply optimization rules in layers
- This time
  - Why Optimization may slow down/fail
  - Some simple workarounds
  - Advanced optimization methods

“Layered” Logistic Regression



NORTHWESTERN UNIVERSITY

Forward propagation



NORTHWESTERN UNIVERSITY

Backpropagation

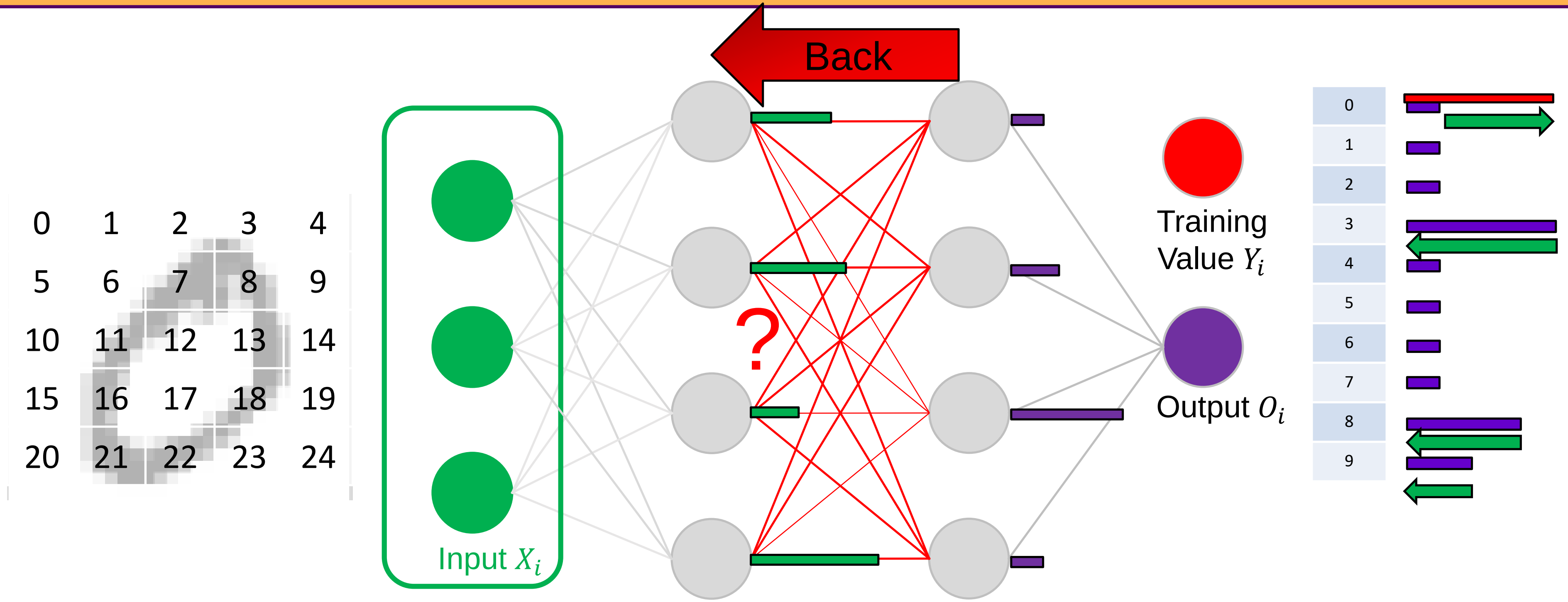
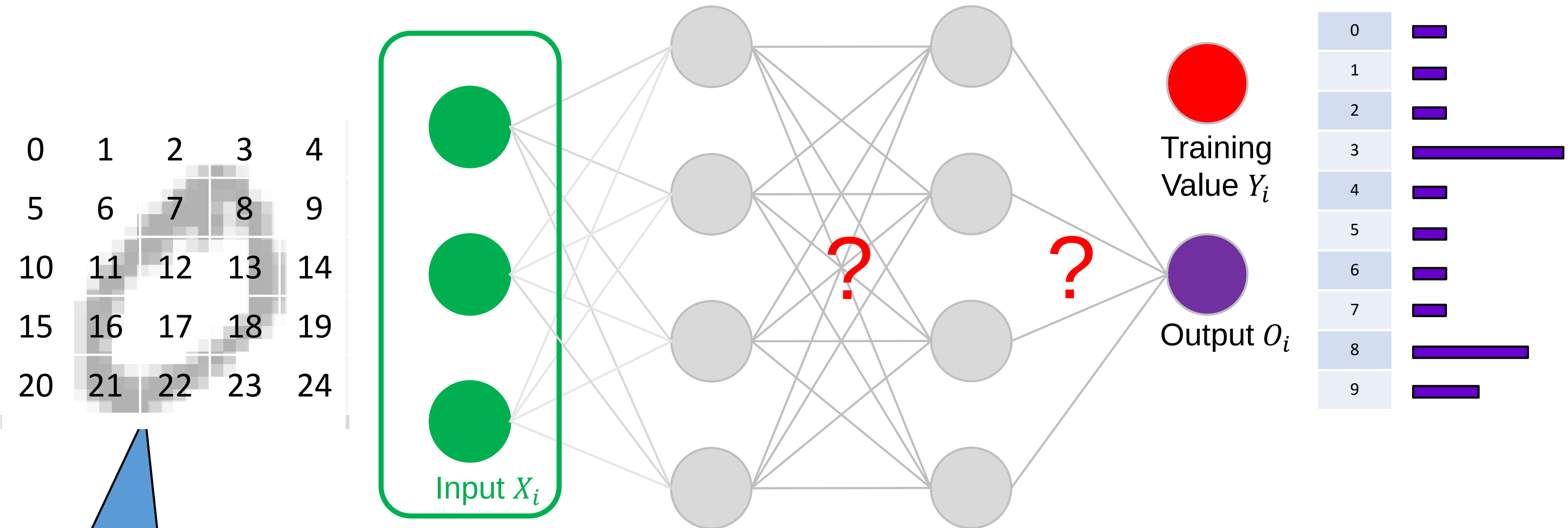


Figure out how to adjust input(green) weights(red) to match target activations(purple)

NORTHWESTERN UNIVERSITY

Feedforward networks

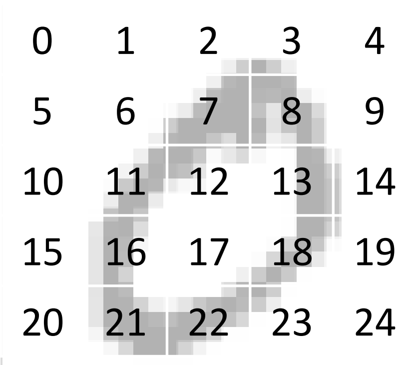


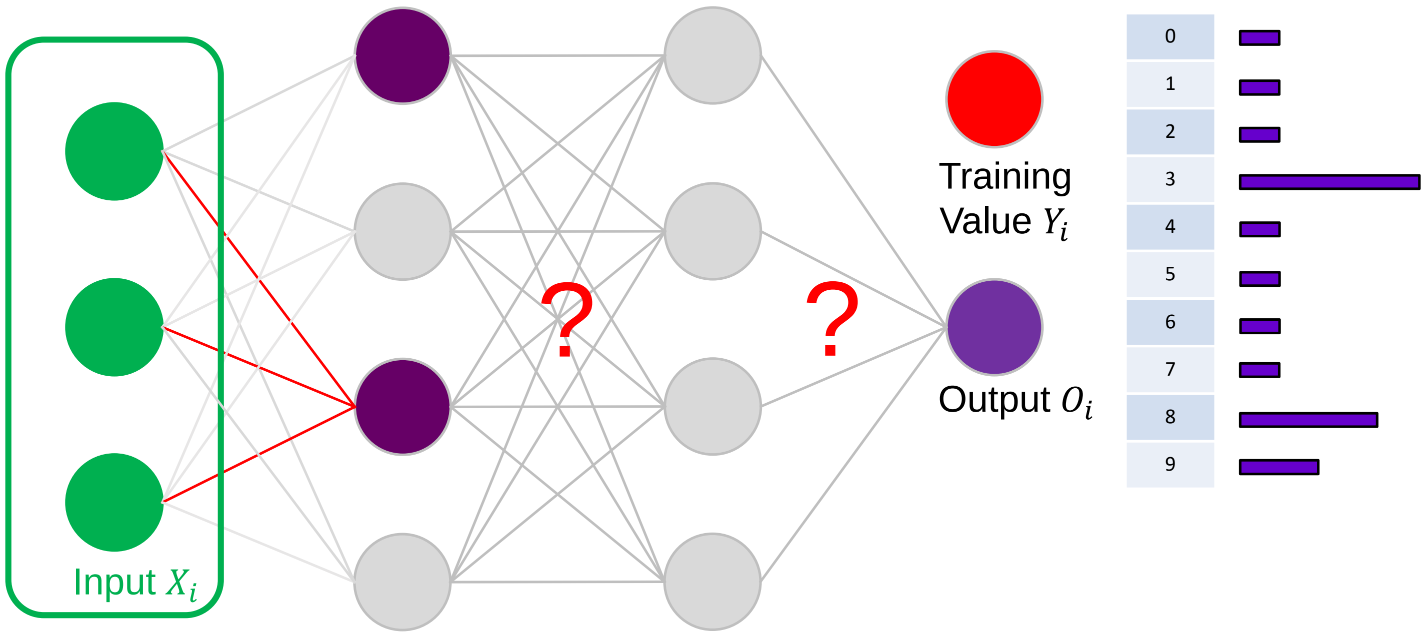
Input is
28x28=784
60,000 Ex

Flatten image to 784 dim x 60k

NORTHWESTERN UNIVERSITY

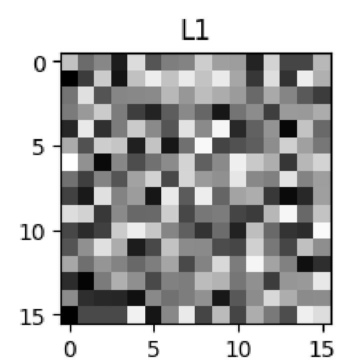
Visualizing NN





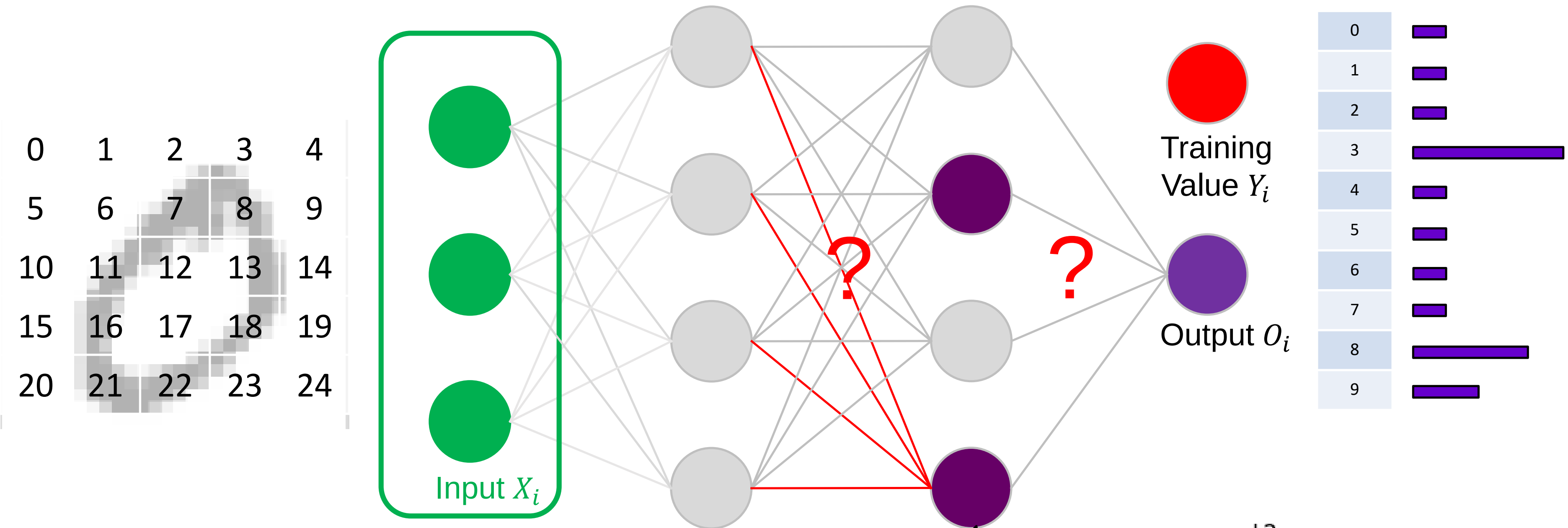
L1 = sigmoid(W1.dot(X))

Layer 1 is 256 units: 16x16 (60k) visualization



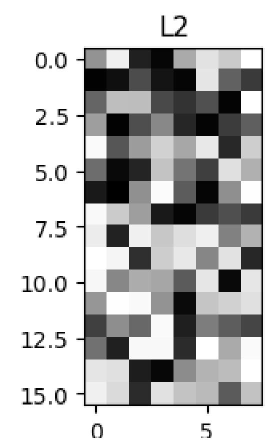
NORTHWESTERN UNIVERSITY

Visualizing NN

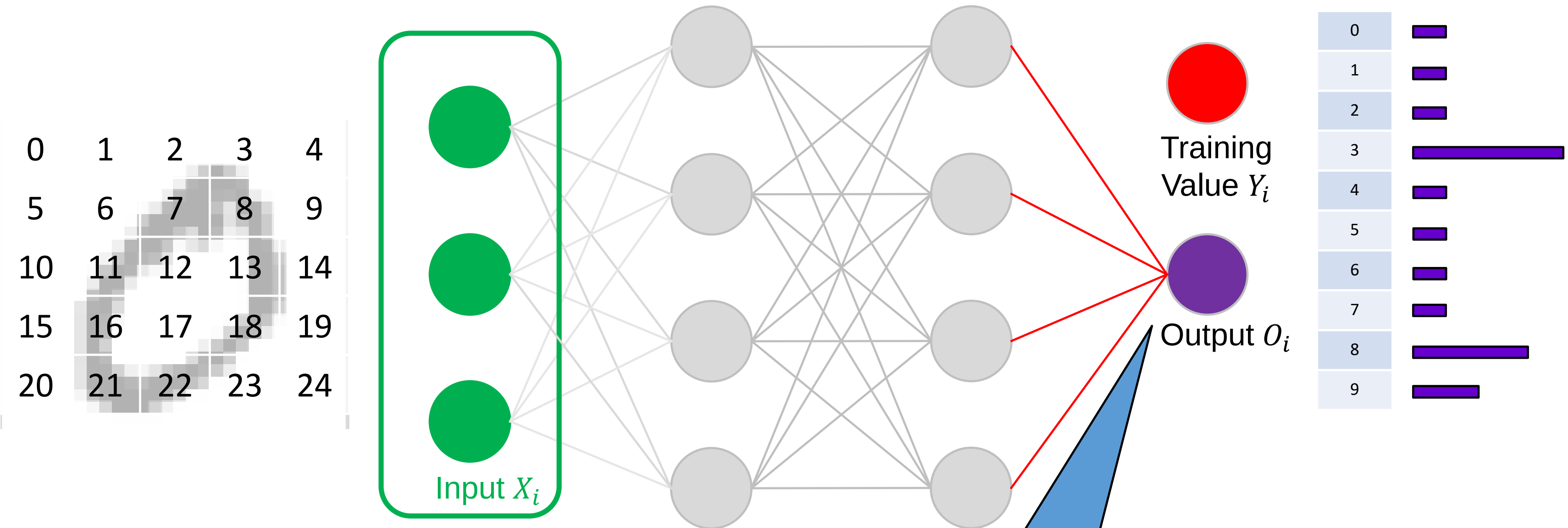


L1 = sigmoid(W1.dot(X))
L2 = sigmoid(W2.dot(L1))

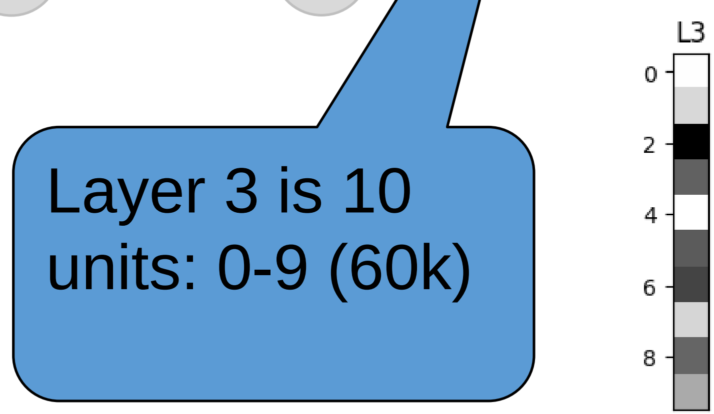
Layer 2 is 128 units: 16x8 (60k) visualization



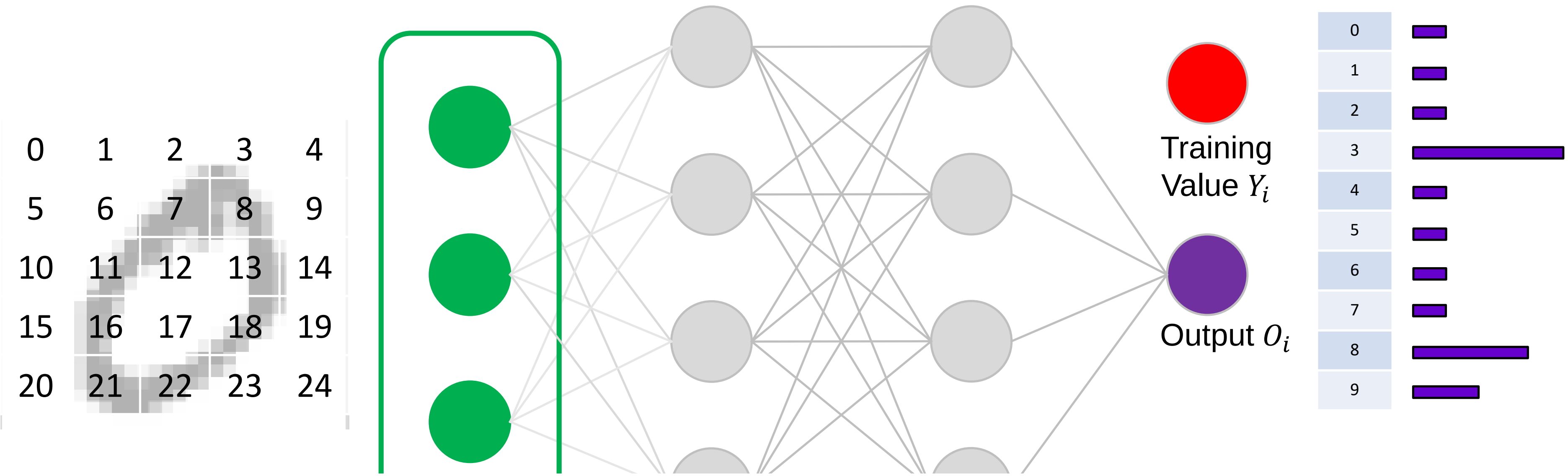
Visualizing NN



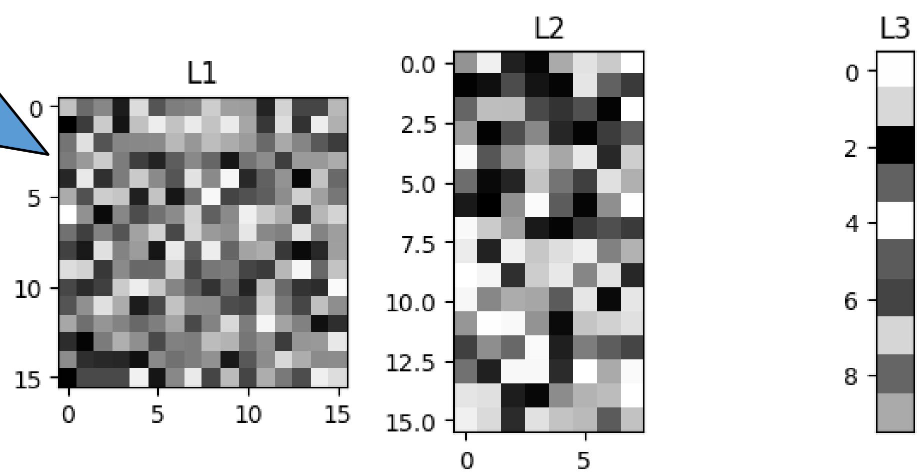
L1 = sigmoid(W1.dot(X))
L2 = sigmoid(W2.dot(L1))
L3 = sigmoid(W3.dot(L2))



Visualizing NN

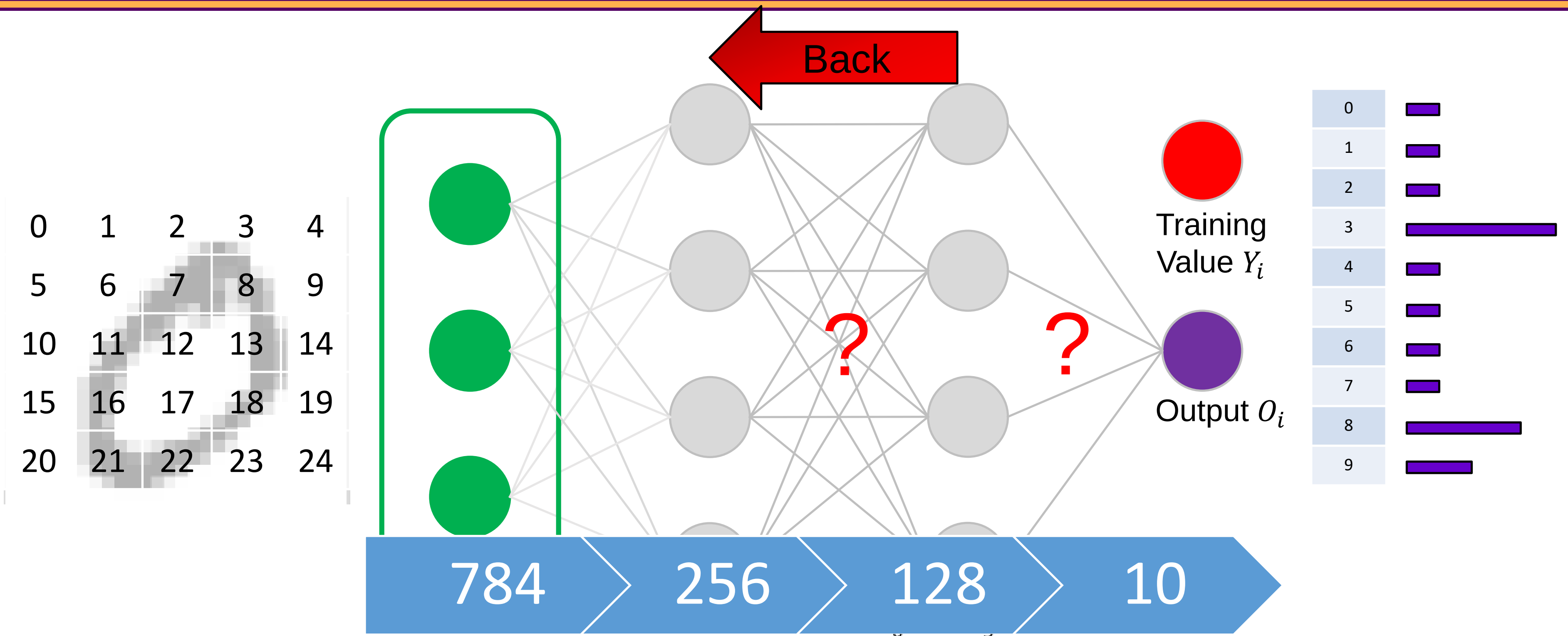


Mean activations over 60k images

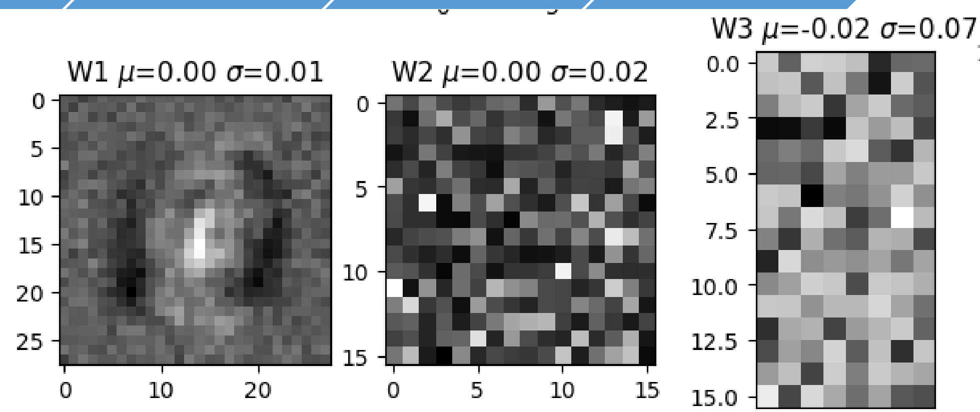


NORTHWESTERN UNIVERSITY

Visualizing NN

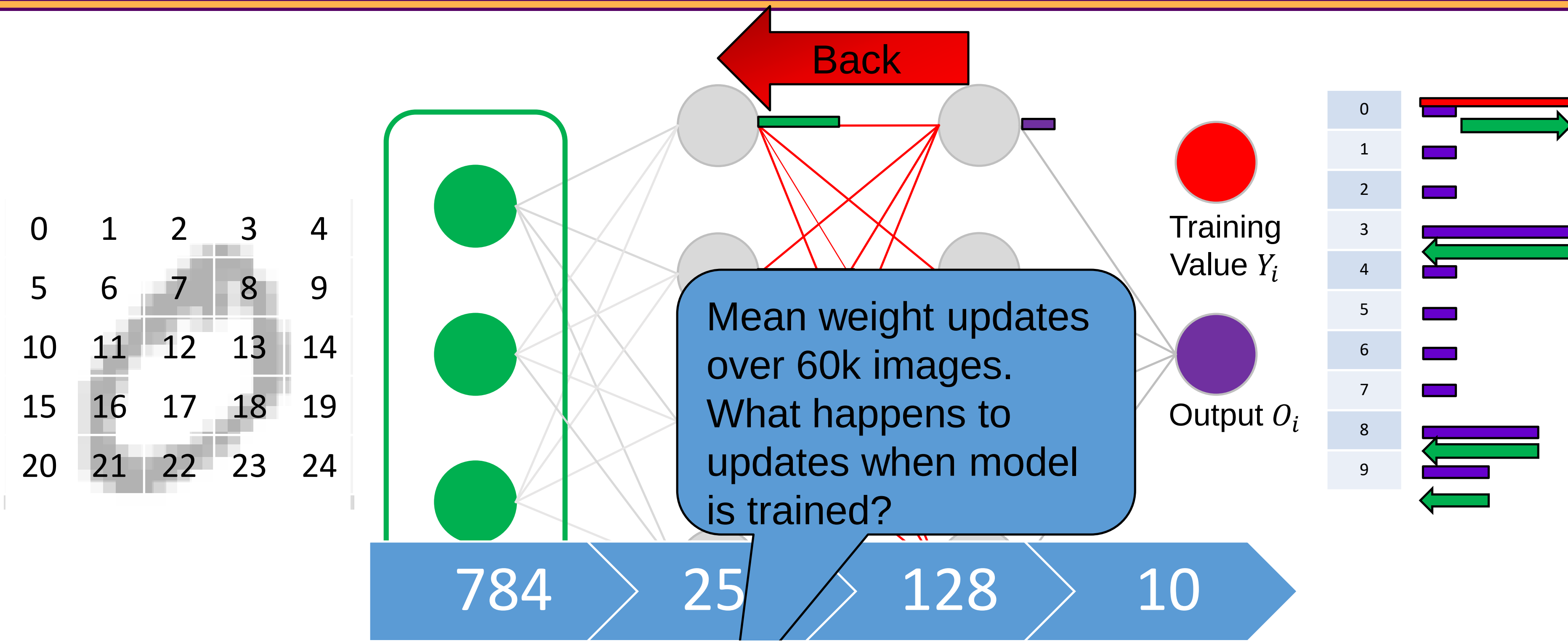


Mean weights over 60k images. Remember: $w_i x_i$ means input and weights share same shape

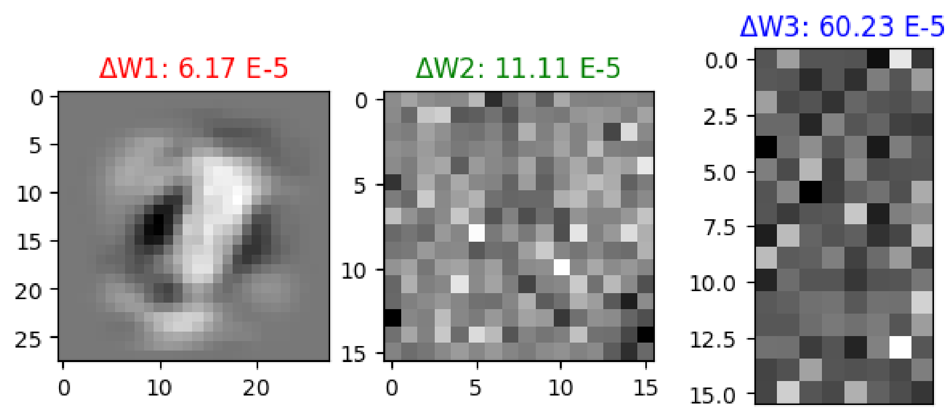


NORTHWESTERN UNIVERSITY

Visualizing NN



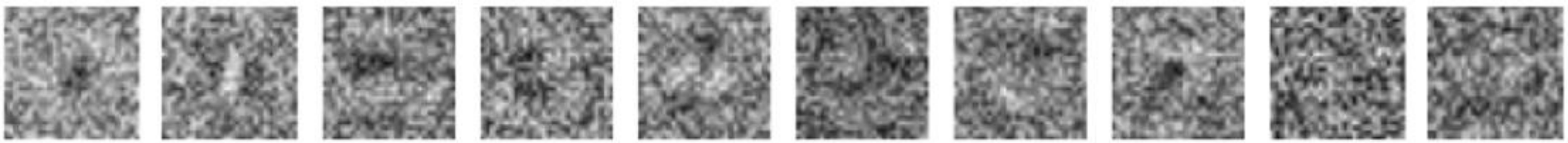

```
# Backward pass
dW3 = (L3 - T) * L3*(1 - L3)
dW2 = W3.T.dot(dW3)*(L2*(1-L2))
dW1 = W2.T.dot(dW2)*(L1*(1-L1))
# Update
W3 -= lr*np.dot(dW3, L2.T)
W2 -= lr*np.dot(dW2, L1.T)
W1 -= lr*np.dot(dW1, X.T)
```



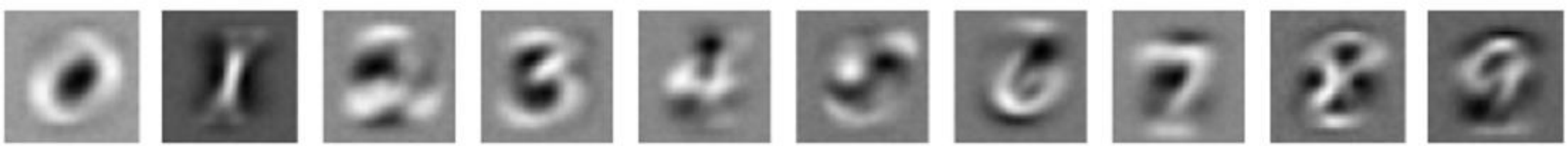
Effects of Regularization

- L1 regularization encourages stronger feature learning

No regularization



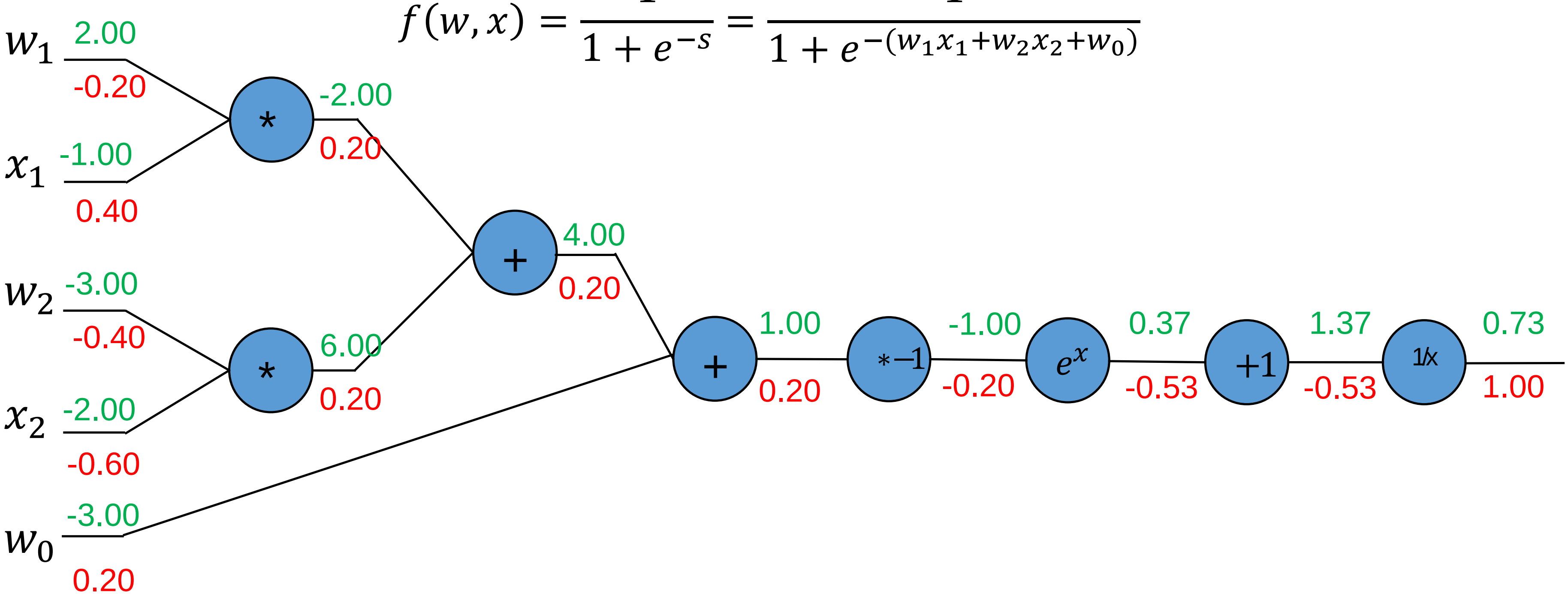
L1 regularization



NORTHWESTERN UNIVERSITY

Backprop in “depth”

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

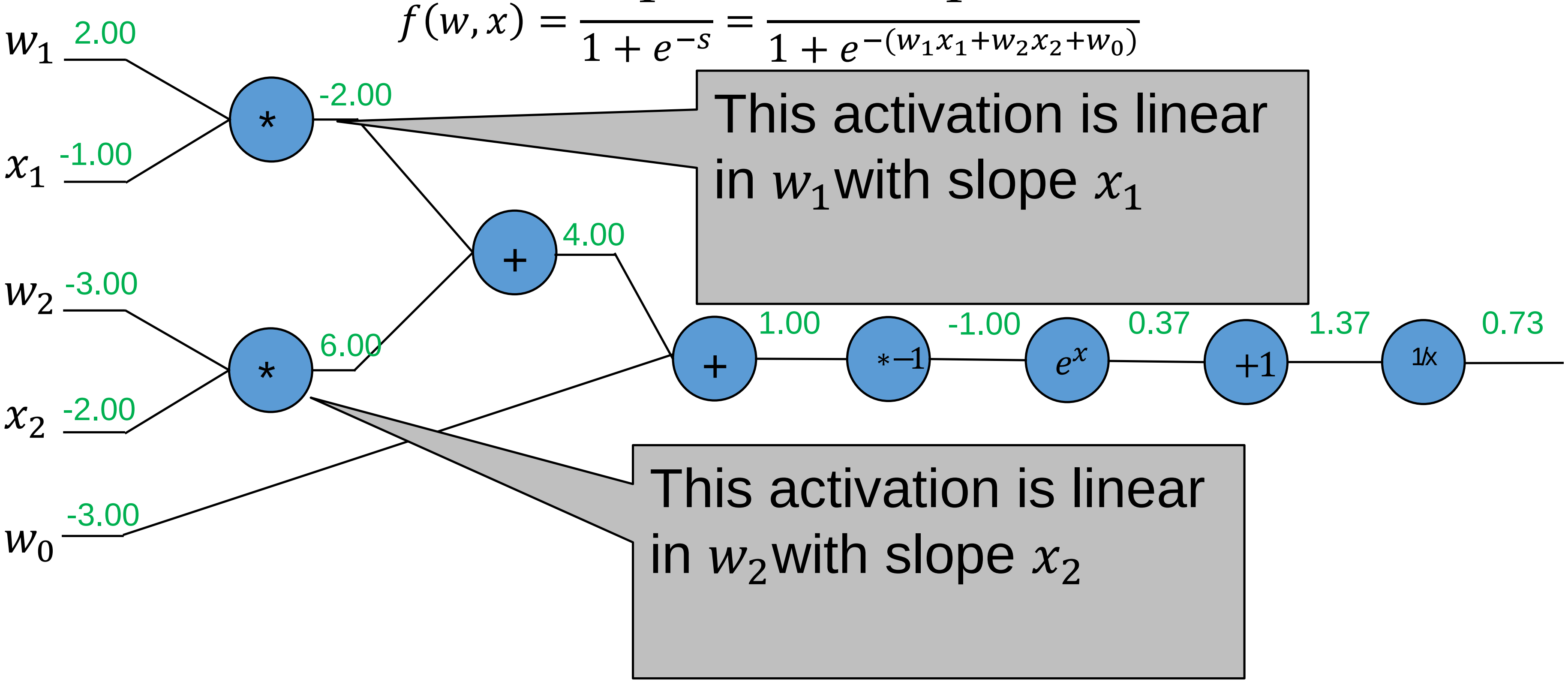


$f(x) = e^x$	$\rightarrow$ $\frac{\partial f}{\partial x} = e^x$	$f(x) = \frac{1}{x}$	$\rightarrow$ $\frac{\partial f}{\partial x} = \frac{-1}{x^2}$
$f_a(x) = ax$	$\rightarrow$ $\frac{\partial f}{\partial x} = a$	$f_c(x) = c + x$	$\rightarrow$ $\frac{\partial f}{\partial x} = 1$

NORTHWESTERN UNIVERSITY

Backprop in "depth": Forward pass

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

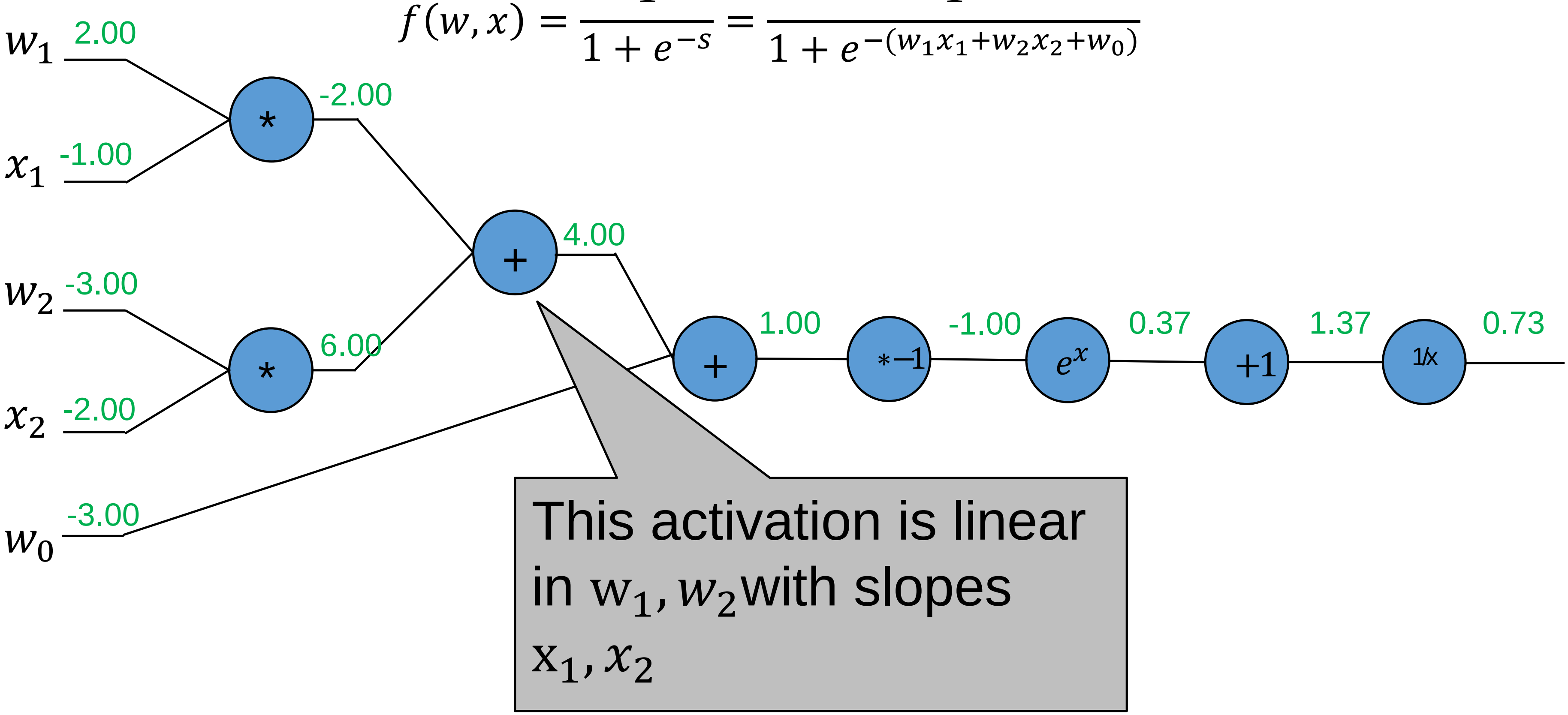


$f(x) = e^x$	$\rightarrow$	$\frac{\partial f}{\partial x} = e^x$	$f(x) = \frac{1}{x}$	$\rightarrow$	$\frac{\partial f}{\partial x} = \frac{-1}{x^2}$
$f_a(x) = ax$	$\rightarrow$	$\frac{\partial f}{\partial x} = a$	$f_c(x) = c + x$	$\rightarrow$	$\frac{\partial f}{\partial x} = 1$

NORTHWESTERN UNIVERSITY

Backprop in "depth": Forward pass

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

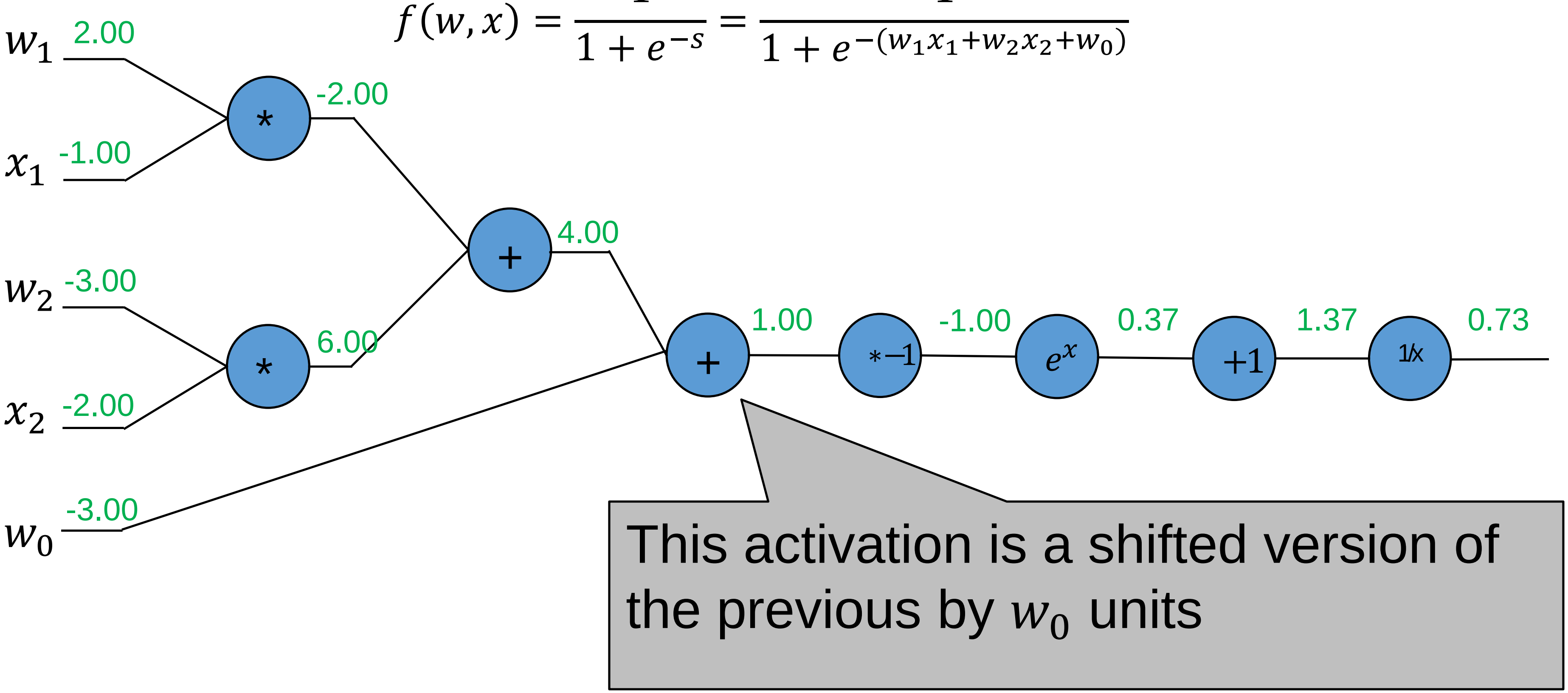


$f(x)=e^x$	$\rightarrow$	$\frac{\partial f}{\partial x}=e^x$	$f(x)=\frac{1}{x}$	$\rightarrow$	$\frac{\partial f}{\partial x}=\frac{-1}{x^2}$
$f_a(x)=ax$	$\rightarrow$	$\frac{\partial f}{\partial x}=a$	$f_c(x)=c+x$	$\rightarrow$	$\frac{\partial f}{\partial x}=1$

NORTHWESTERN UNIVERSITY

Backprop in "depth": Forward pass

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

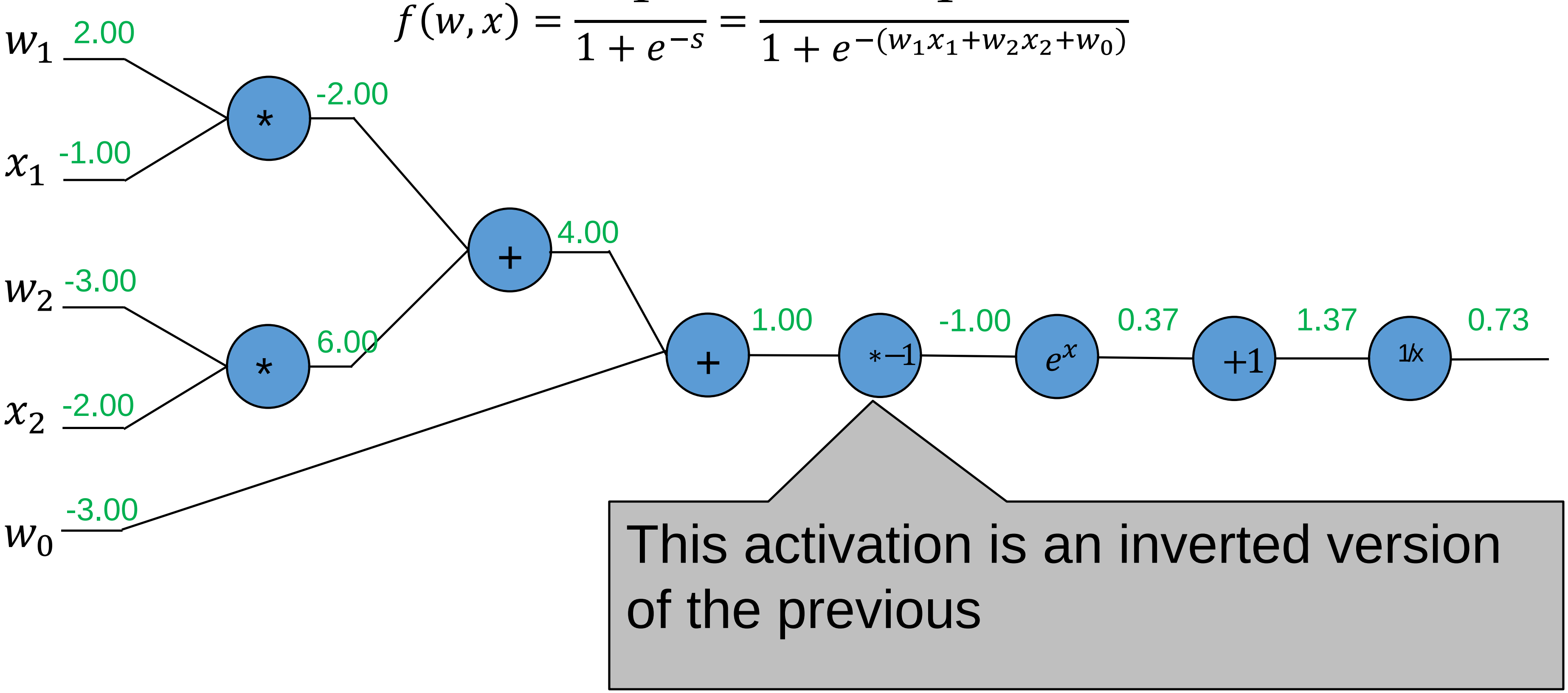


$f(x) = e^x$	$\rightarrow$	$\frac{\partial f}{\partial x} = e^x$	$f(x) = \frac{1}{x}$	$\rightarrow$	$\frac{\partial f}{\partial x} = \frac{-1}{x^2}$
$f_a(x) = ax$	$\rightarrow$	$\frac{\partial f}{\partial x} = a$	$f_c(x) = c + x$	$\rightarrow$	$\frac{\partial f}{\partial x} = 1$

NORTHWESTERN UNIVERSITY

Backprop in "depth": Forward pass

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

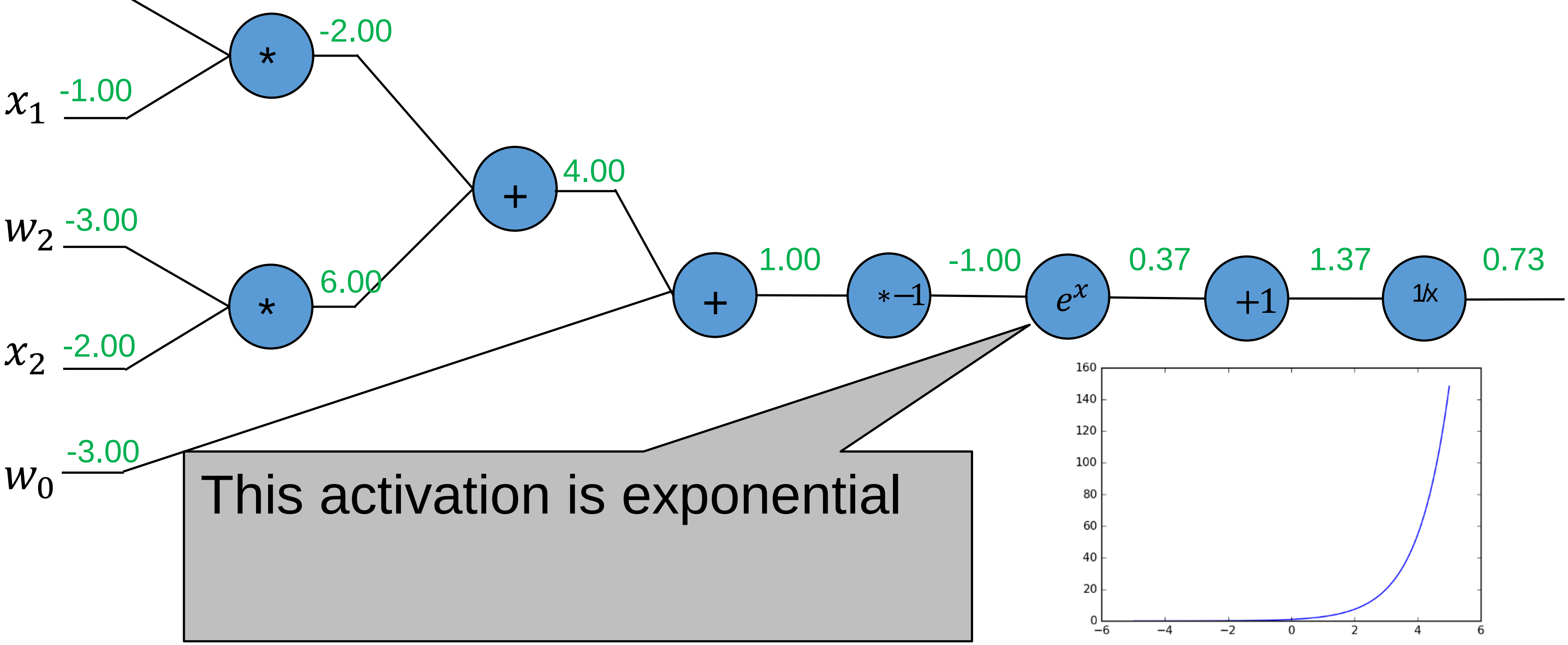


$f(x) = e^x \quad \rightarrow \quad \frac{\partial f}{\partial x} = e^x$	$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{\partial f}{\partial x} = \frac{-1}{x^2}$
$f_a(x) = ax \quad \rightarrow \quad \frac{\partial f}{\partial x} = a$	$f_c(x) = c + x \quad \rightarrow \quad \frac{\partial f}{\partial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

NORTHWESTERN UNIVERSITY

Backprop in “depth”: Forward pass

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

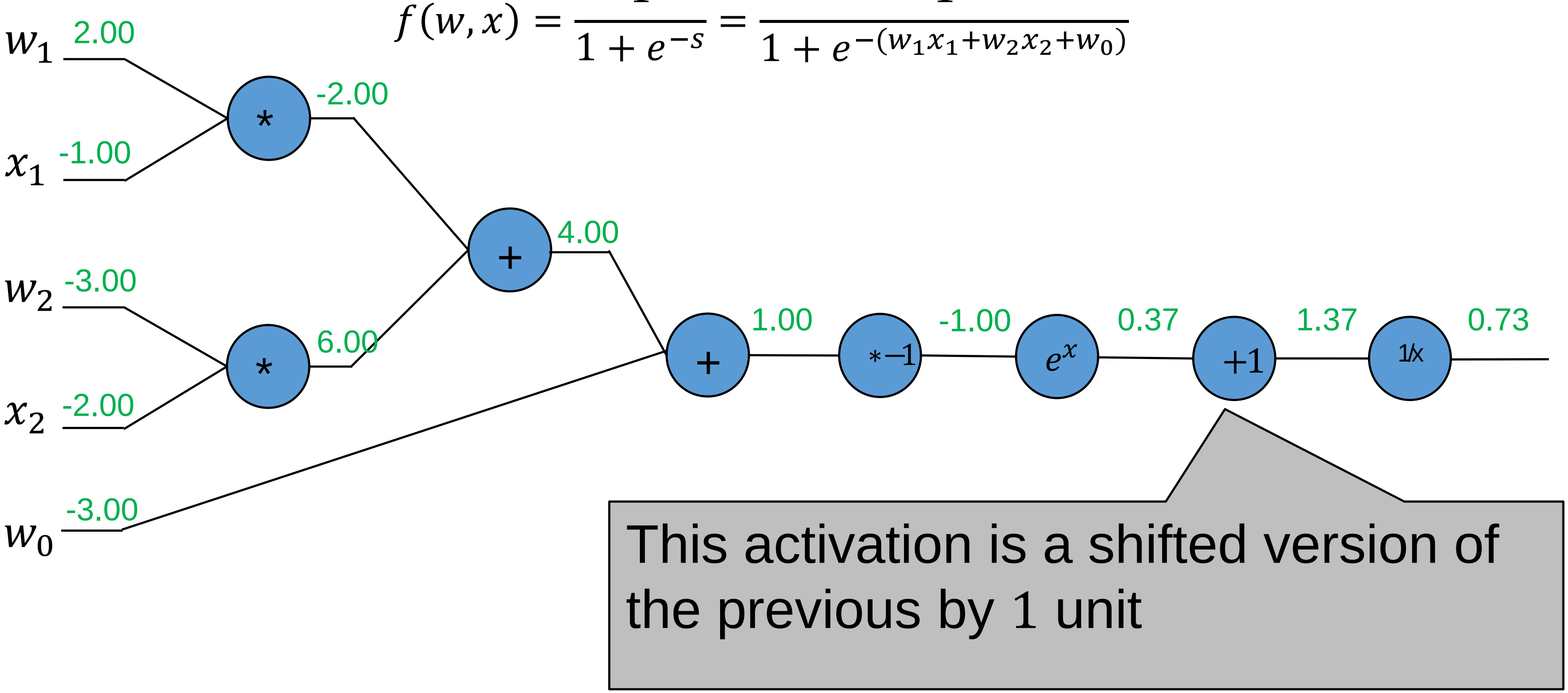


$f(x) = e^x \quad \rightarrow \quad \frac{\partial f}{\partial x} = e^x$	$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{\partial f}{\partial x} = \frac{-1}{x^2}$
$f_a(x) = ax \quad \rightarrow \quad \frac{\partial f}{\partial x} = a$	$f_c(x) = c + x \quad \rightarrow \quad \frac{\partial f}{\partial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

NORTHWESTERN UNIVERSITY

Backprop in “depth”: Forward pass

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

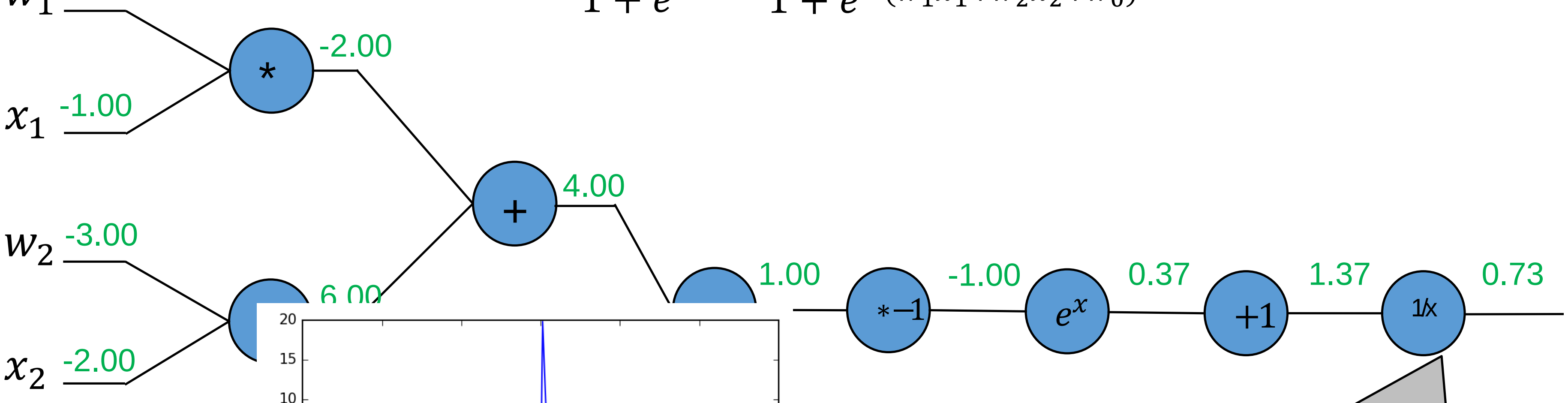


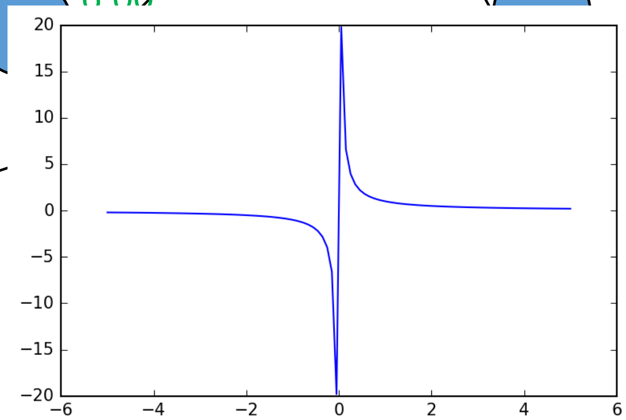
$f(x) = e^x \quad \rightarrow$	$\frac{\partial f}{\partial x} = e^x$	$f(x) = \frac{1}{x} \quad \rightarrow$	$\frac{\partial f}{\partial x} = \frac{-1}{x^2}$
$f_a(x) = ax \quad \rightarrow$	$\frac{\partial f}{\partial x} = a$	$f_c(x) = c + x \quad \rightarrow$	$\frac{\partial f}{\partial x} = 1$

NORTHWESTERN UNIVERSITY

Backprop in "depth": Forward pass

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





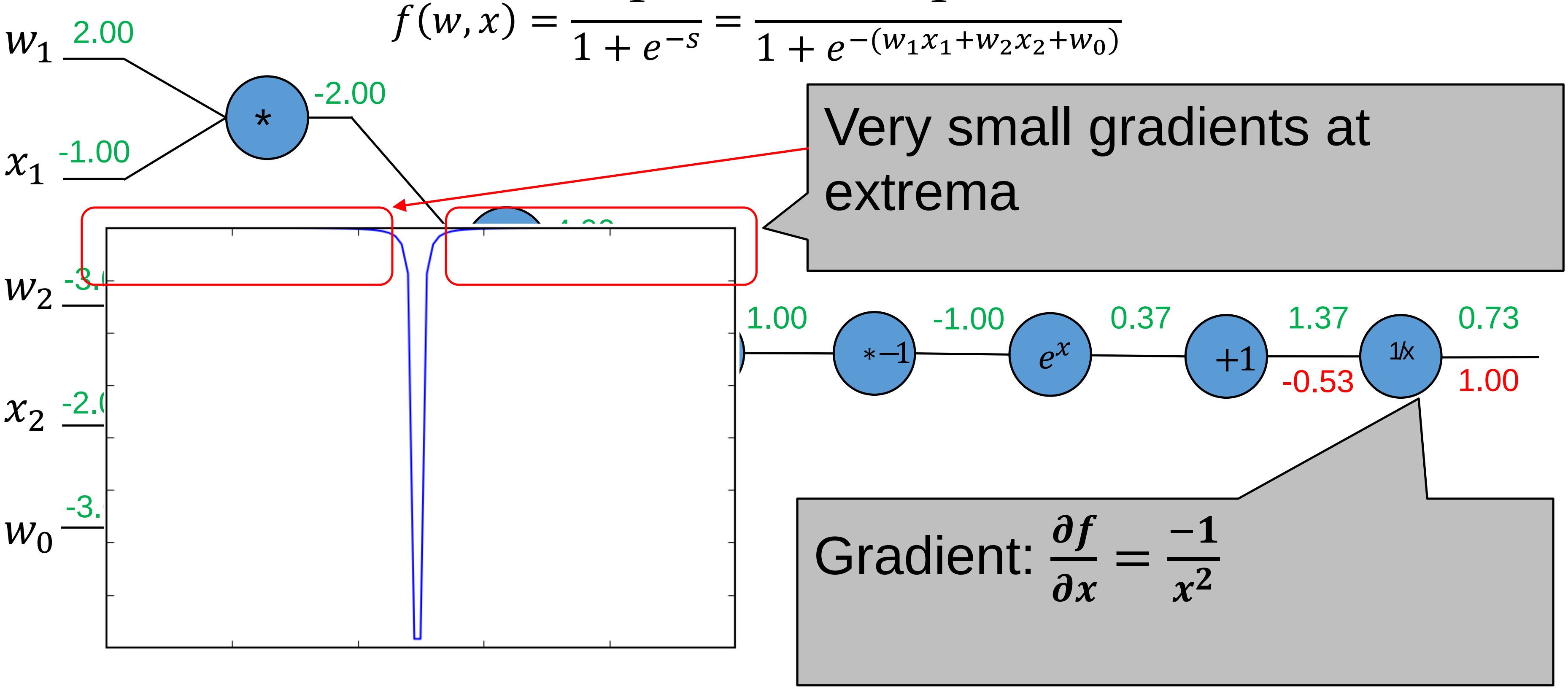
This activation is hyperbolic

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

NORTHWESTERN UNIVERSITY

Backprop in “depth”: Backward pass

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

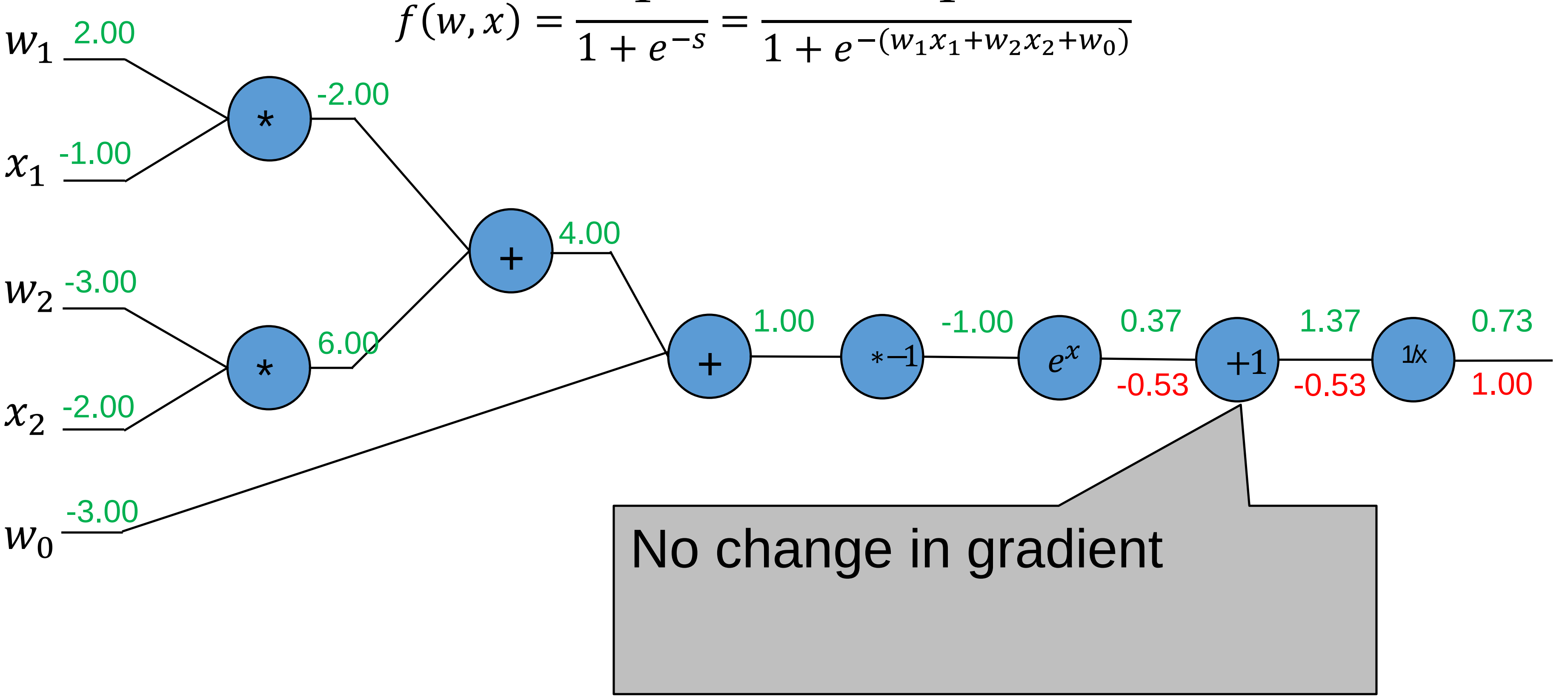


$f(x) = e^x \quad \rightarrow \quad \frac{\partial f}{\partial x} = e^x$	$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{\partial f}{\partial x} = \frac{-1}{x^2}$
$f_a(x) = ax \quad \rightarrow \quad \frac{\partial f}{\partial x} = a$	$f_c(x) = c + x \quad \rightarrow \quad \frac{\partial f}{\partial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

NORTHWESTERN UNIVERSITY

Backprop in "depth": Backward pass

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

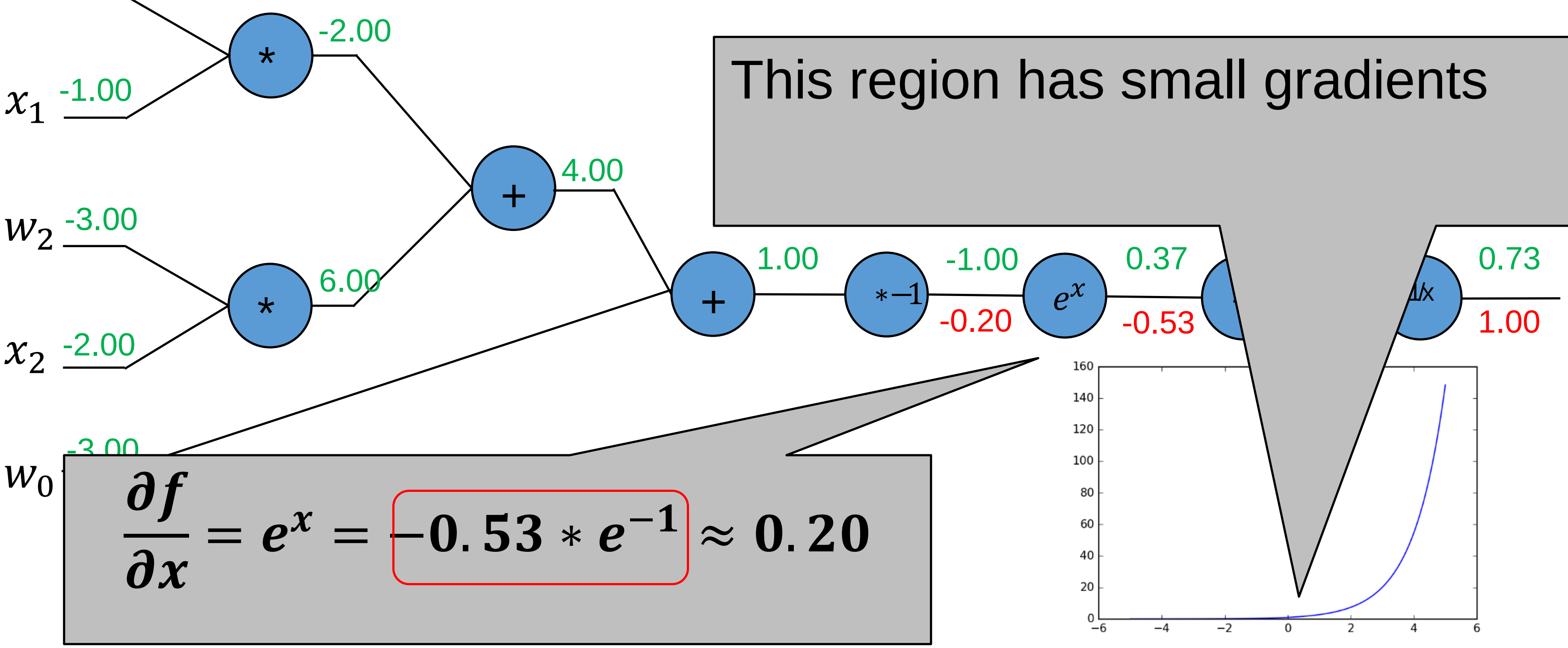


$f(x) = e^x \quad \rightarrow \quad \frac{\partial f}{\partial x} = e^x$	$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{\partial f}{\partial x} = \frac{-1}{x^2}$
$f_a(x) = ax \quad \rightarrow \quad \frac{\partial f}{\partial x} = a$	$f_c(x) = c + x \quad \rightarrow \quad \frac{\partial f}{\partial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

NORTHWESTERN UNIVERSITY

Backprop in “depth”: Backward pass

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



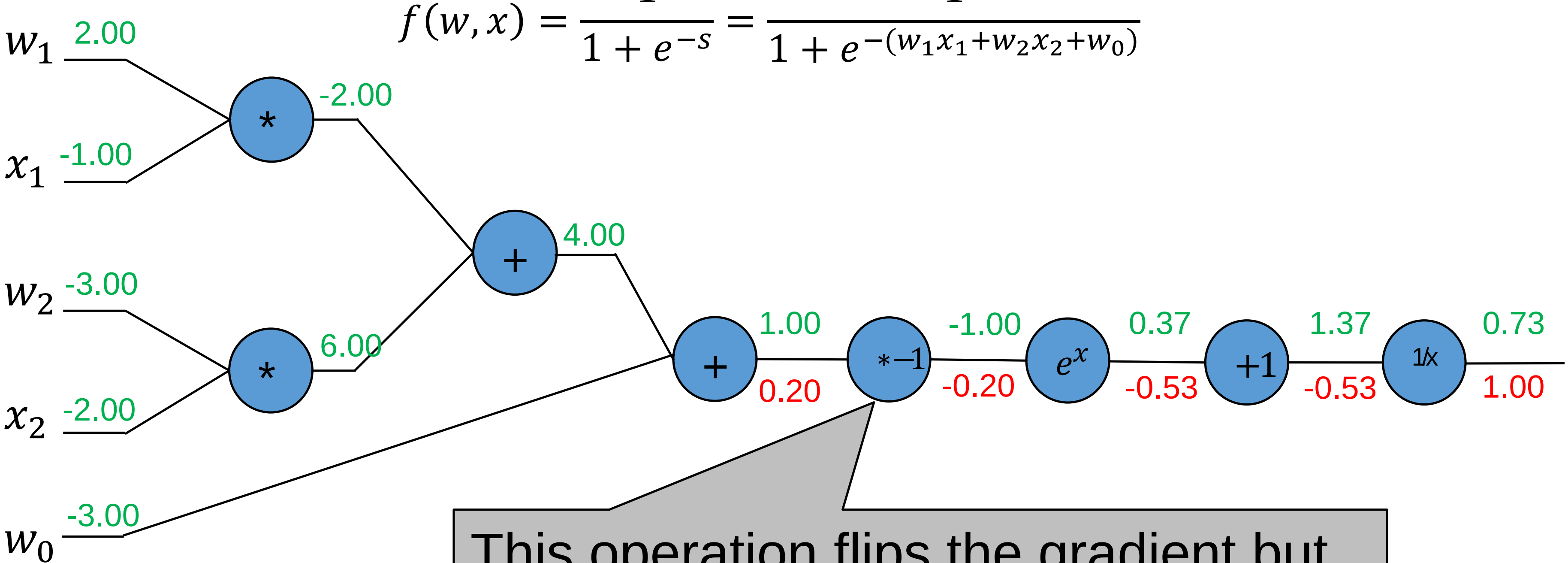
$$\frac{\partial f}{\partial x} = e^x = \boxed{-0.53 * e^{-1}} \approx 0.20$$

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

NORTHWESTERN UNIVERSITY

Backprop in "depth": Backward pass

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



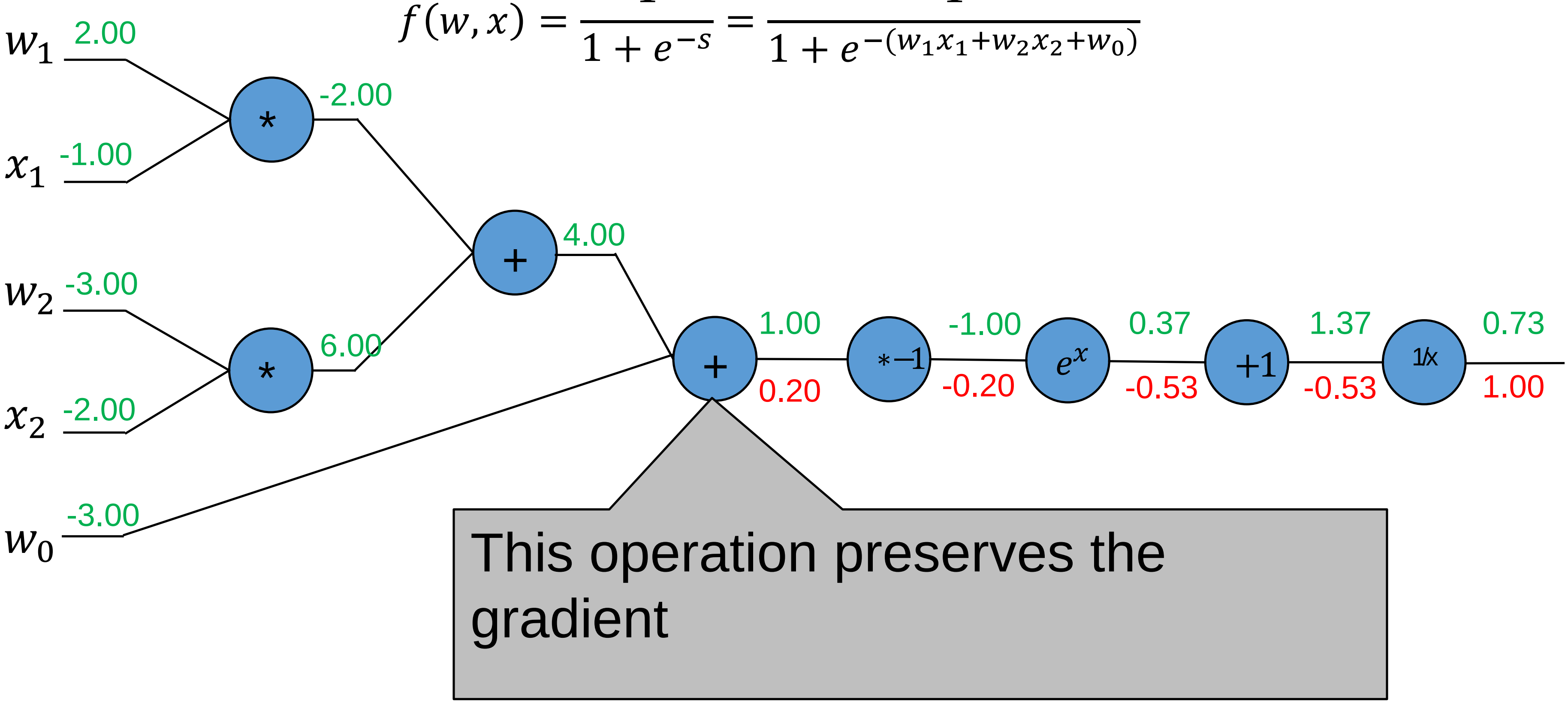
This operation flips the gradient but preserves magnitude

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

NORTHWESTERN UNIVERSITY

Backprop in "depth": Backward pass

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

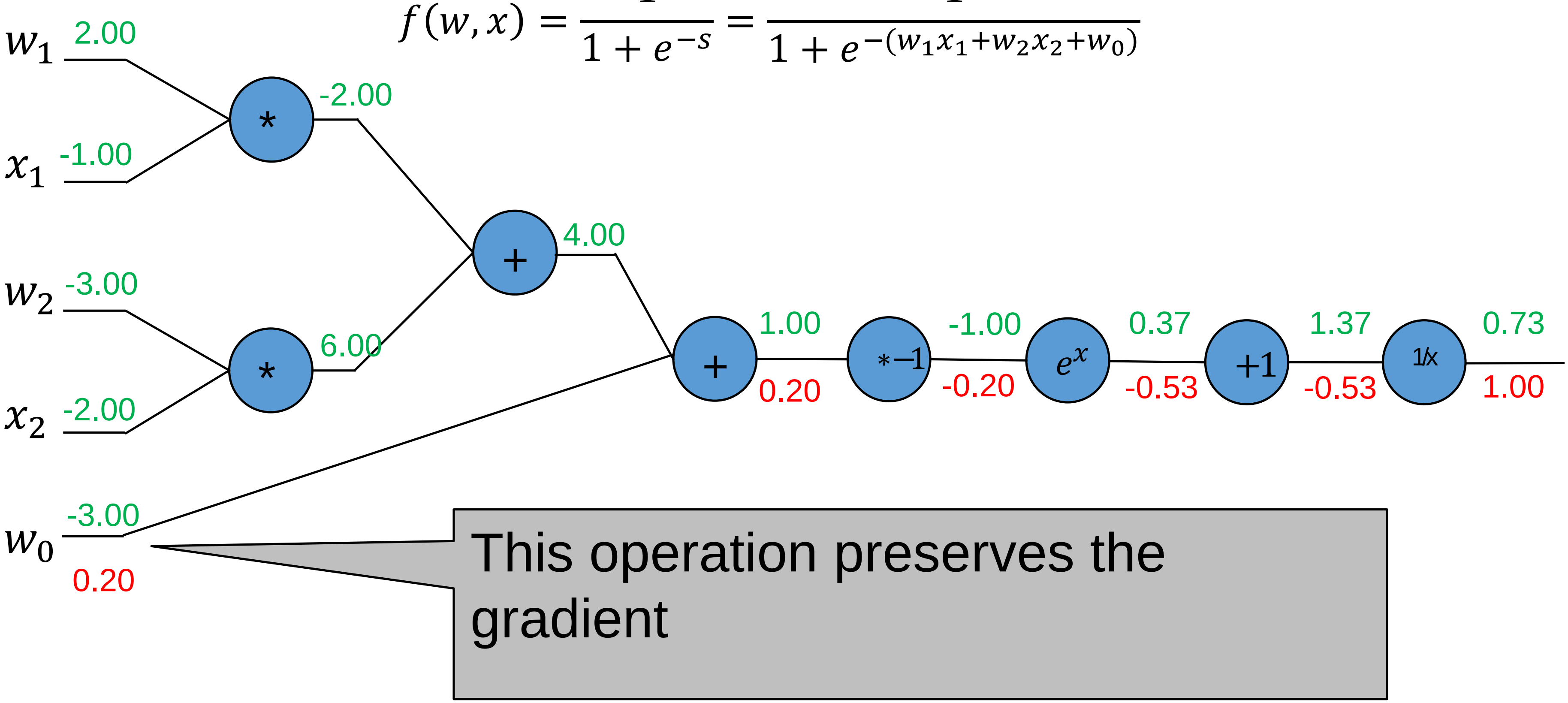


$f(x) = e^x$	$\rightarrow$	$\frac{\partial f}{\partial x} = e^x$	$f(x) = \frac{1}{x}$	$\rightarrow$	$\frac{\partial f}{\partial x} = \frac{-1}{x^2}$
$f_a(x) = ax$	$\rightarrow$	$\frac{\partial f}{\partial x} = a$	$f_c(x) = c + x$	$\rightarrow$	$\frac{\partial f}{\partial x} = 1$

NORTHWESTERN UNIVERSITY

Backprop in "depth": Backward pass

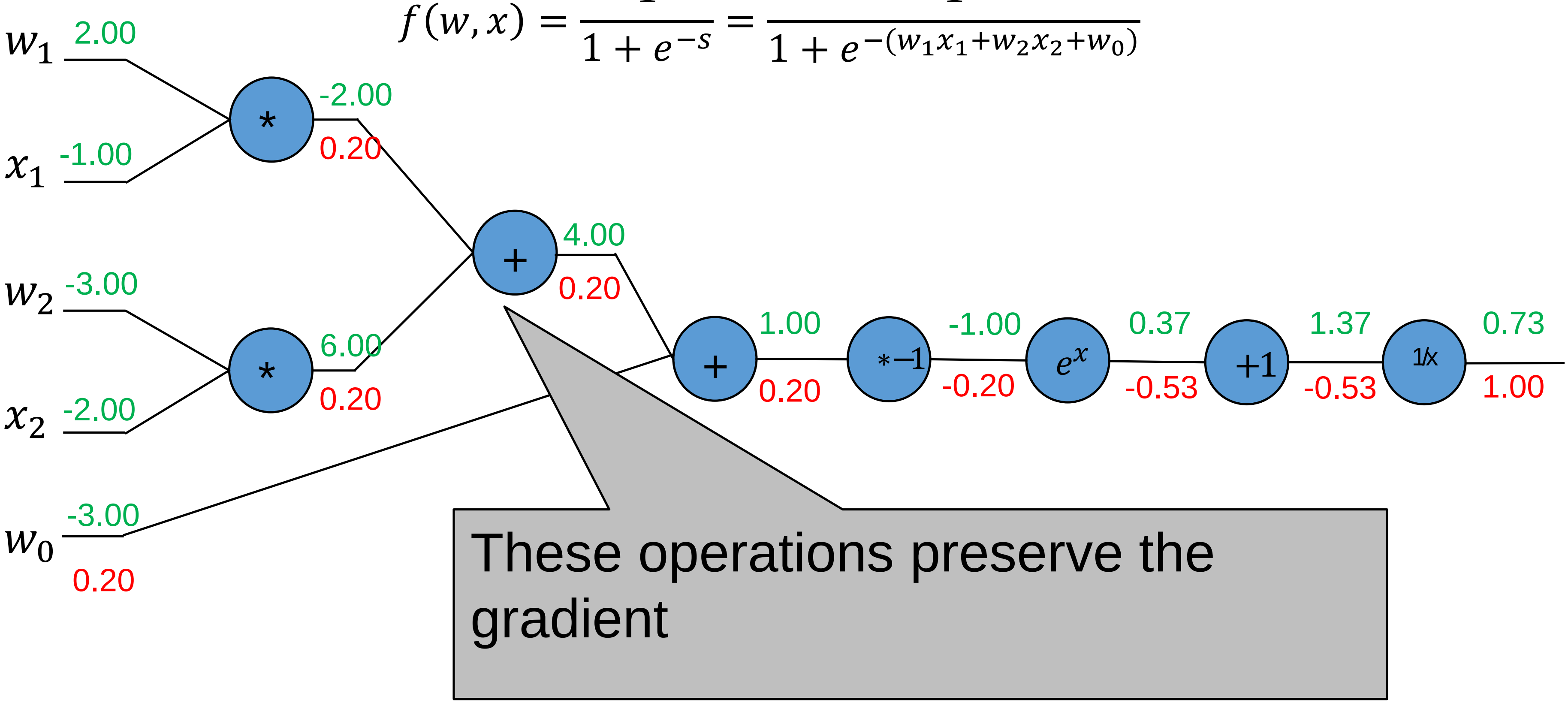
$$f(w,x) = \frac{1}{1+e^{-s}} = \frac{1}{1+e^{-(w_1x_1+w_2x_2+w_0)}}$$



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

Backprop in “depth”: Backward pass

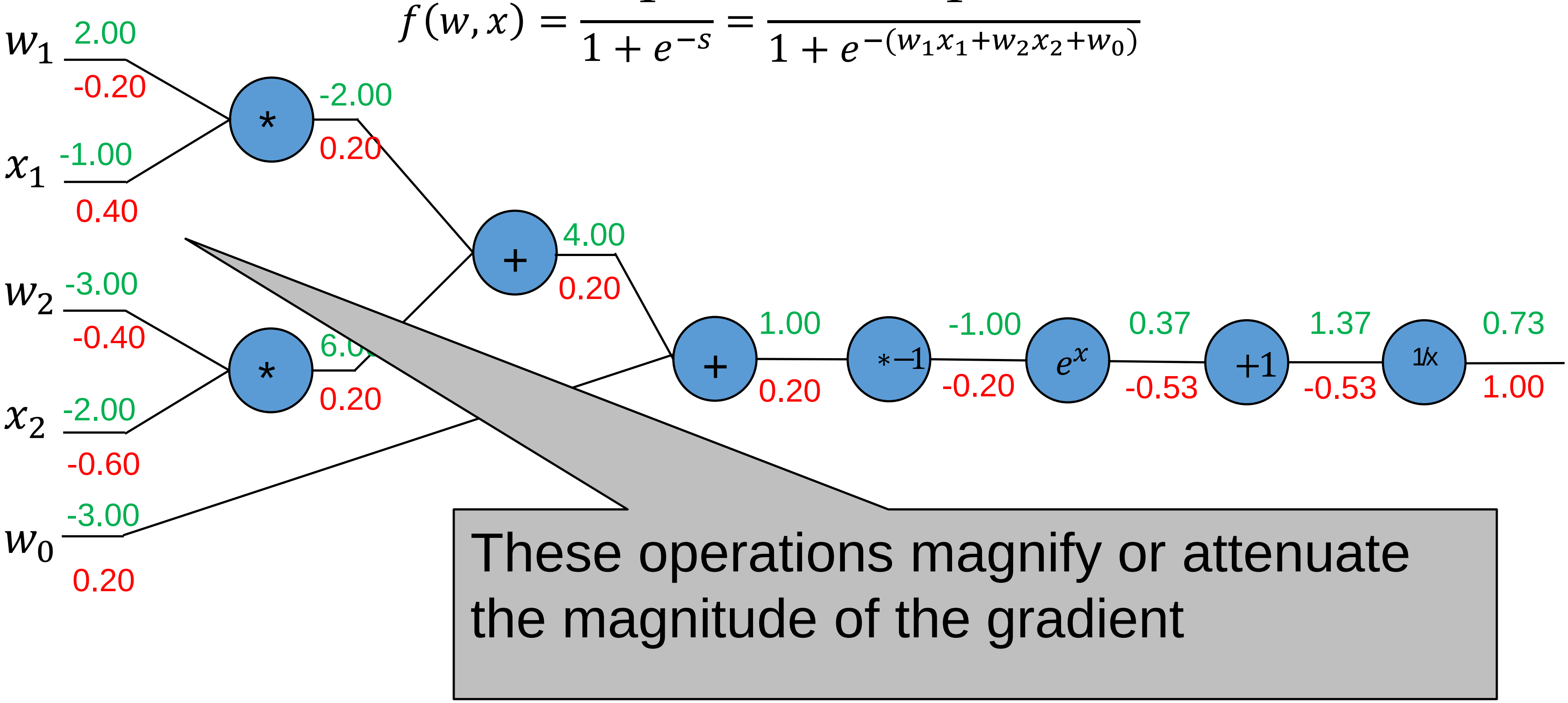
$$f(w,x) = \frac{1}{1+e^{-s}} = \frac{1}{1+e^{-(w_1x_1+w_2x_2+w_0)}}$$



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

Backprop in "depth": Backward pass

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

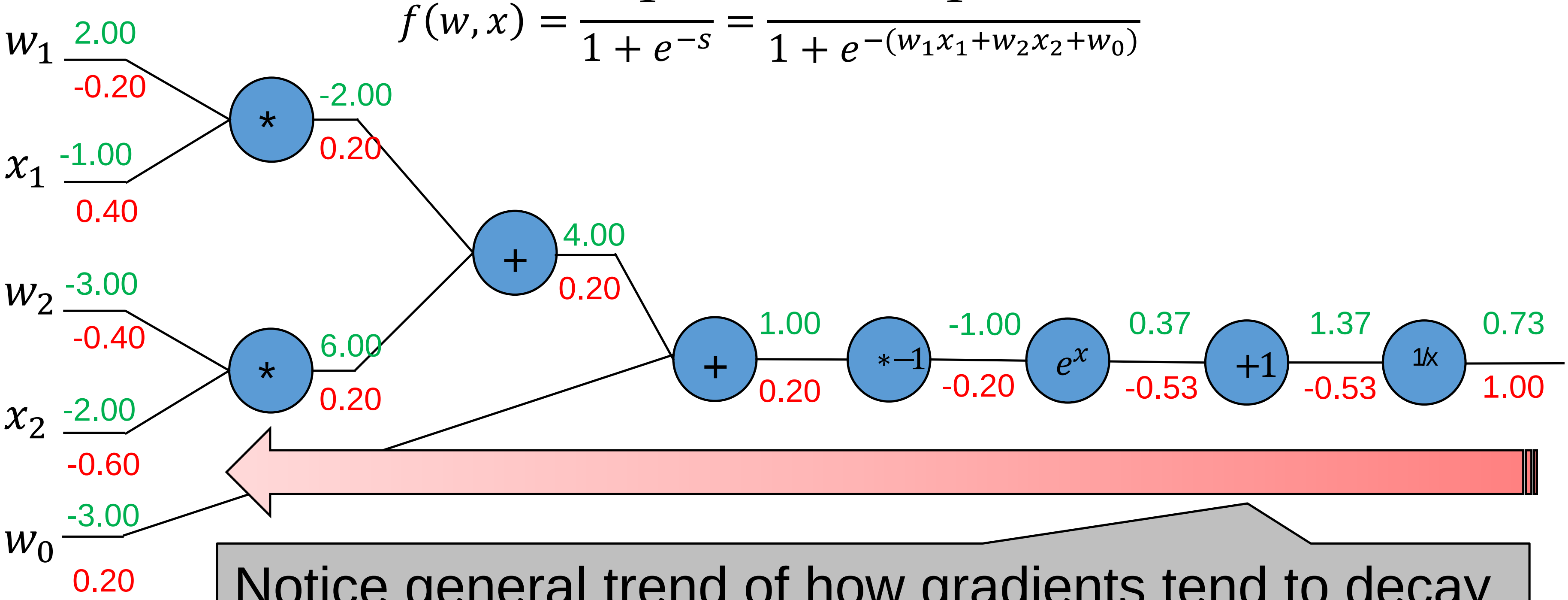


$f(x) = e^x$	$\rightarrow$	$\frac{\partial f}{\partial x} = e^x$	$f(x) = \frac{1}{x}$	$\rightarrow$	$\frac{\partial f}{\partial x} = \frac{-1}{x^2}$
$\boxed{f_a(x) = ax \quad \rightarrow \quad \frac{\partial f}{\partial x} = a}$			$f_c(x) = c + x$	$\rightarrow$	$\frac{\partial f}{\partial x} = 1$

NORTHWESTERN UNIVERSITY

Backprop in “depth”: Backward pass

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



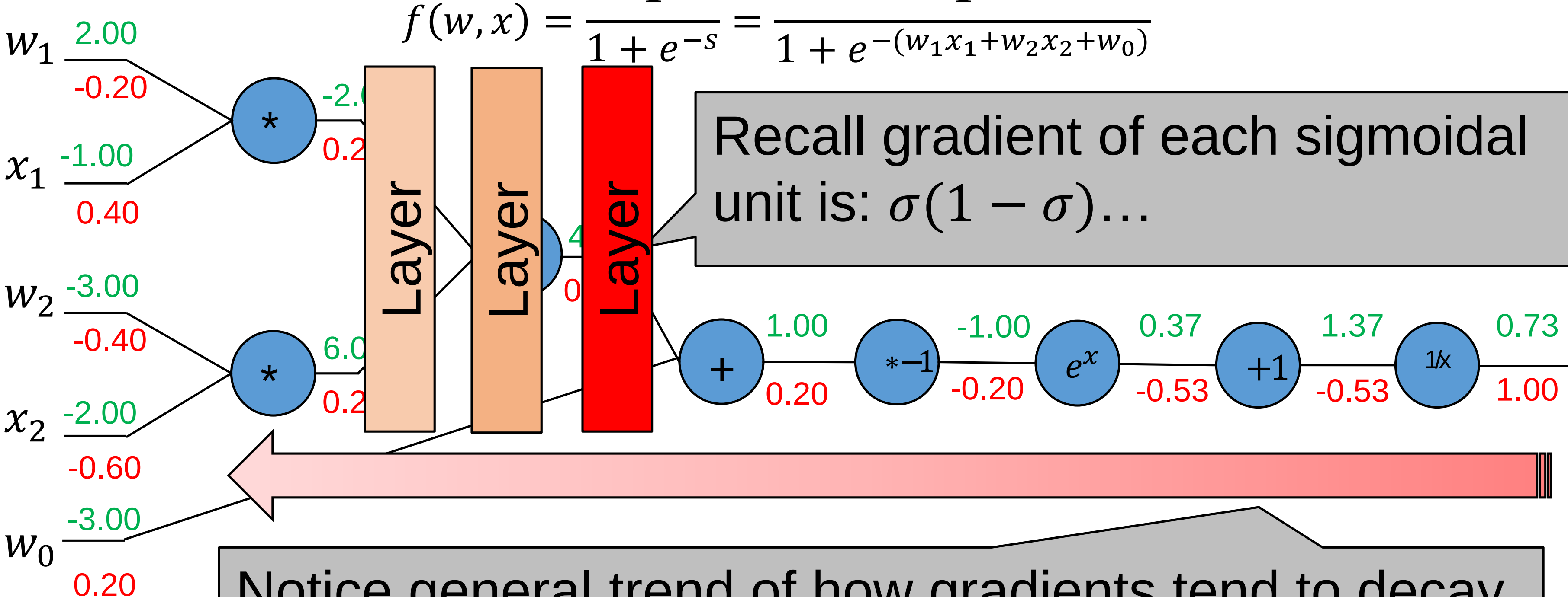
Notice general trend of how gradients tend to decay in magnitude through layers of backprop… why?

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

NORTHWESTERN UNIVERSITY

Backprop in “depth”: Backward pass

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



Recall gradient of each sigmoidal unit is: $\sigma(1-\sigma)$…

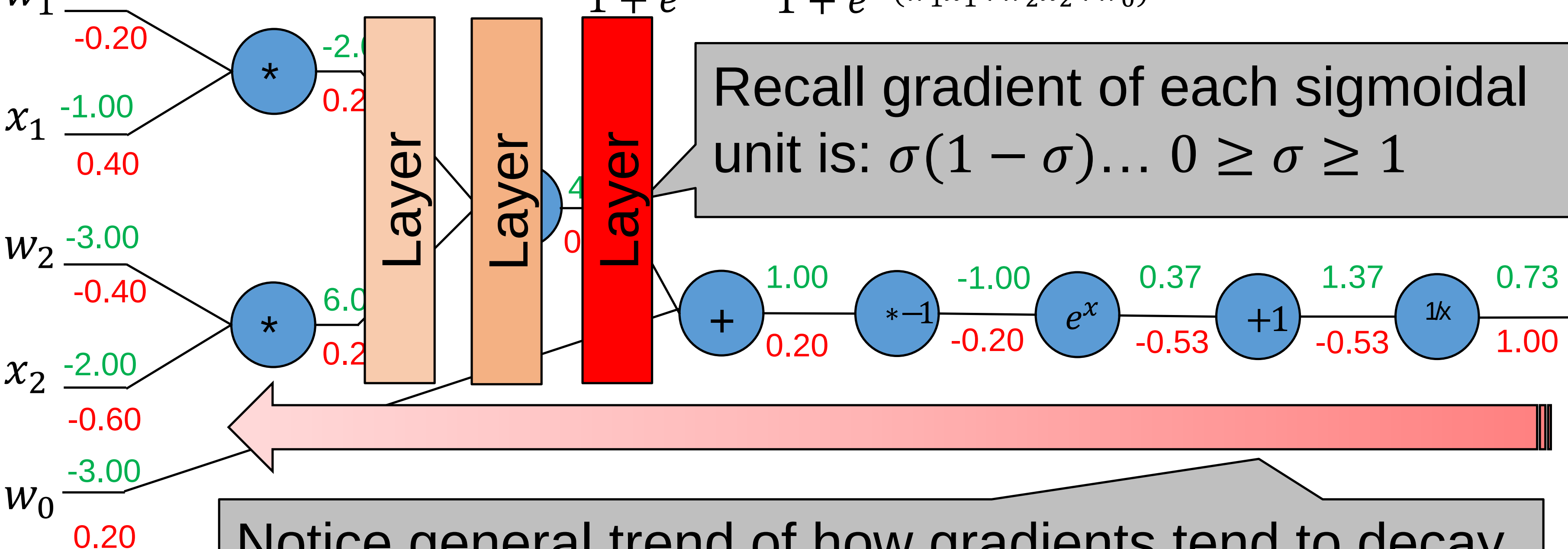
Notice general trend of how gradients tend to decay in magnitude through layers of backprop… why?

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

NORTHWESTERN UNIVERSITY

Backprop in "depth": Backward pass

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



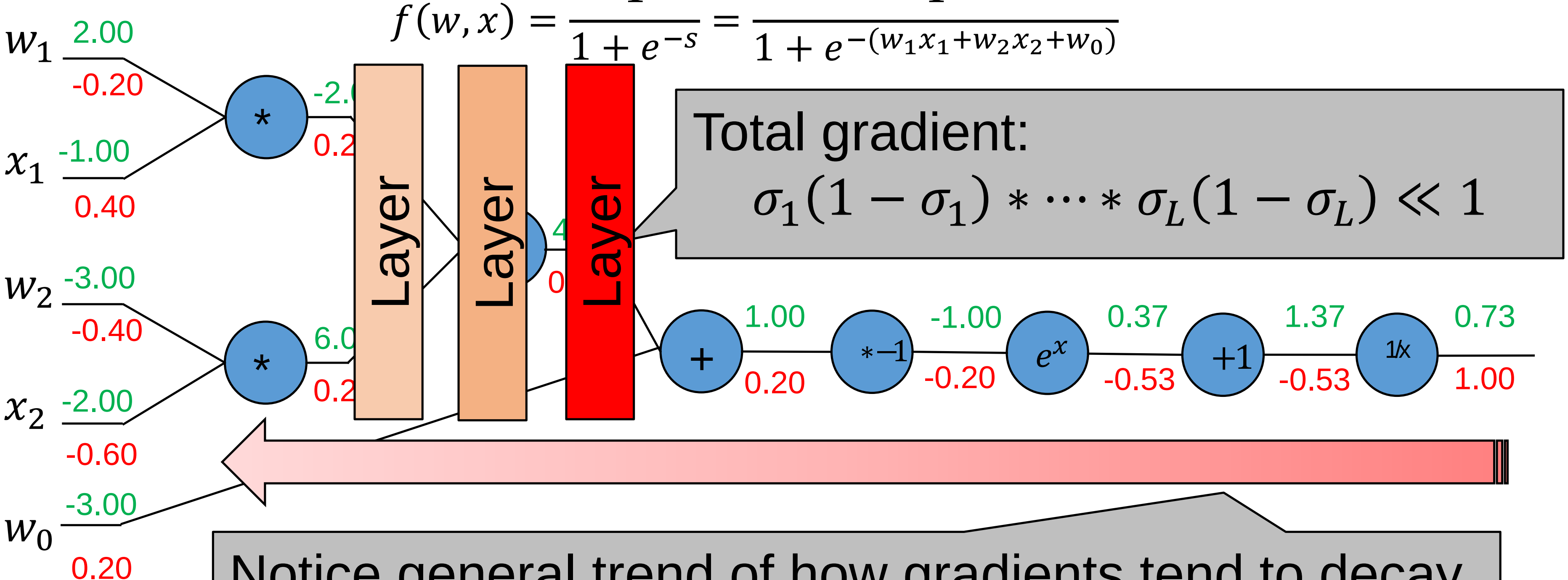
Recall gradient of each sigmoidal unit is: $\sigma(1-\sigma)\ldots\ 0 \geq \sigma \geq 1$

Notice general trend of how gradients tend to decay in magnitude through layers of backprop… why?

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

Backprop in “depth”: Backward pass

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



Total gradient:

$$\sigma_1(1-\sigma_1) * \cdots * \sigma_L(1-\sigma_L) \ll 1$$

Notice general trend of how gradients tend to decay in magnitude through layers of backprop… why?

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

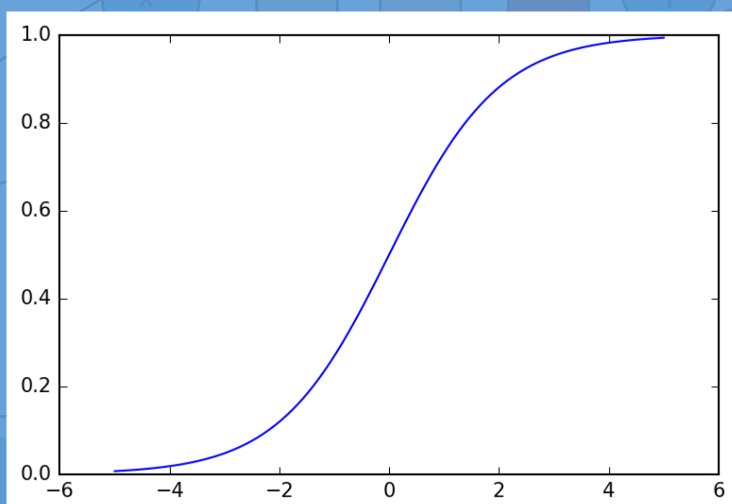
Backprop in "depth": Backward pass

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

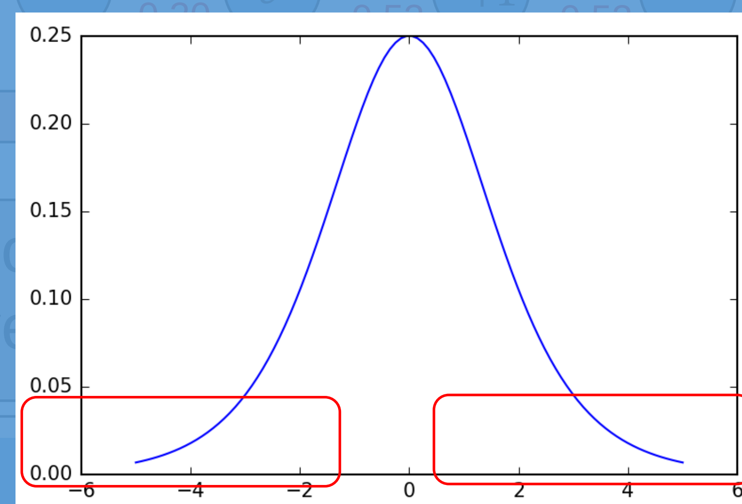
The overall effect is that gradients decay with each layer of sigmoidal activations.

→ Training slows down a lot at extrema

$\sigma(s)$



$\sigma'(s) = \sigma(s)(1-\sigma(s))$



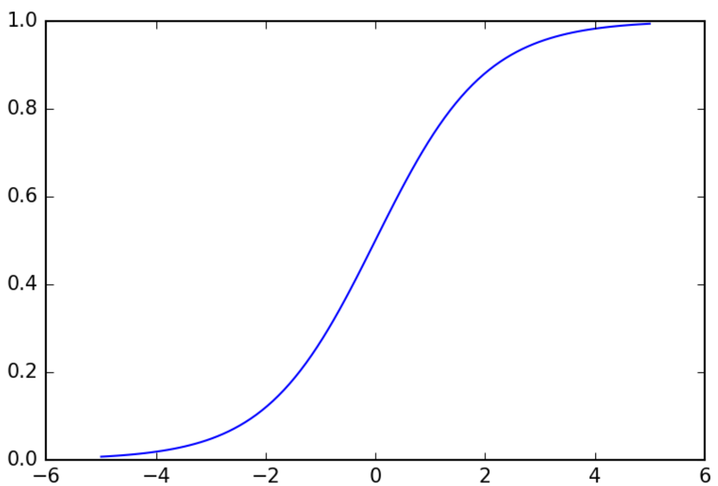
NORTHWESTERN UNIVERSITY

Active regions of Sigmoids

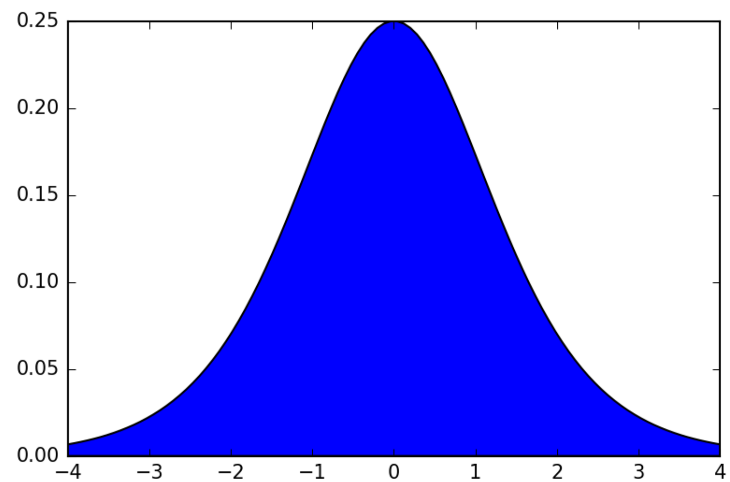
- Sigmoids are relatively effective in active region
- > 95% of gradient density in $\sigma'(s)$ is in [-4,4]

→ Idea: keep sigmoid activations in this active region

$$\sigma(s)$$



$$\sigma'(s) = \sigma(s)(1 - \sigma(s))$$



NORTHWESTERN UNIVERSITY

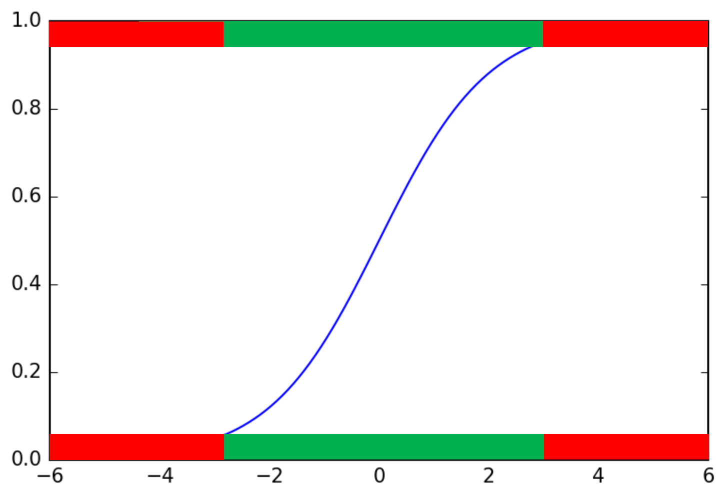
Active regions of Sigmoids

- Idea: keep sigmoid activations in this active region

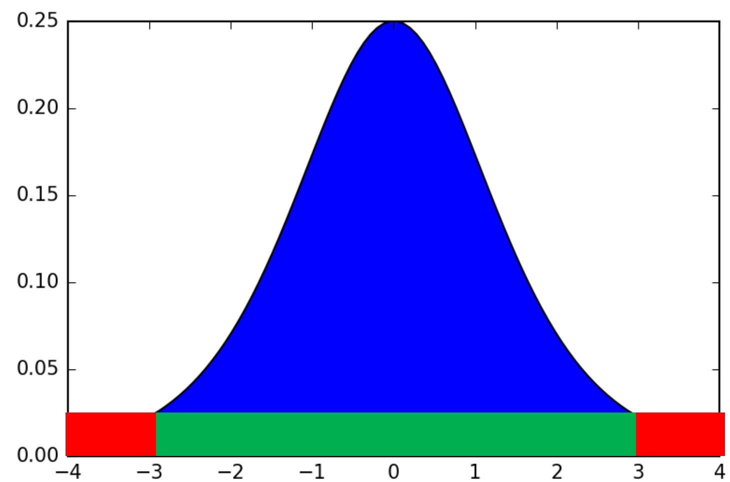
$$f(w,x) = \frac{1}{1+e^{-s}} = \frac{1}{1+e^{-(w_1x_1+w_2x_2+w_0)}}$$

This means that: $-3 \leq w_1x_1 + w_2x_2 + w_0 \leq 3$

$\sigma(s)$

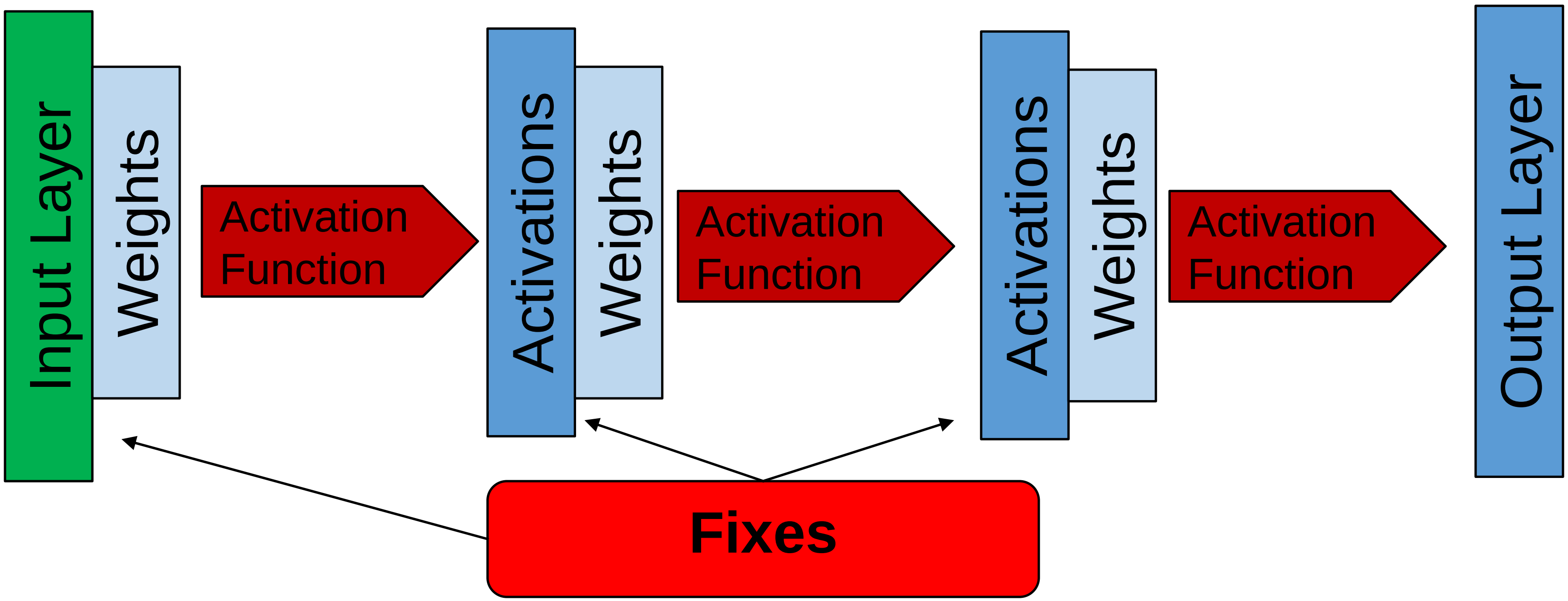


$\sigma'(s) = \sigma(s)(1-\sigma(s))$

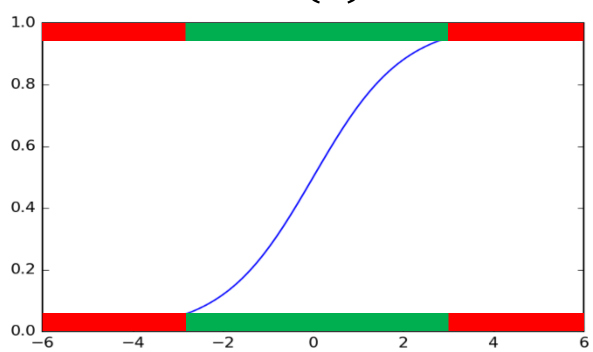


NORTHWESTERN UNIVERSITY

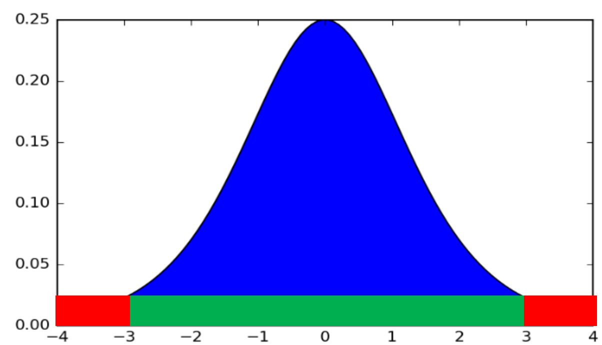
Keeping Sigmoids in active region



$\sigma(s)$



$\sigma'(s) = \sigma(s)(1 - \sigma(s))$



Simple fix: Clip gradients

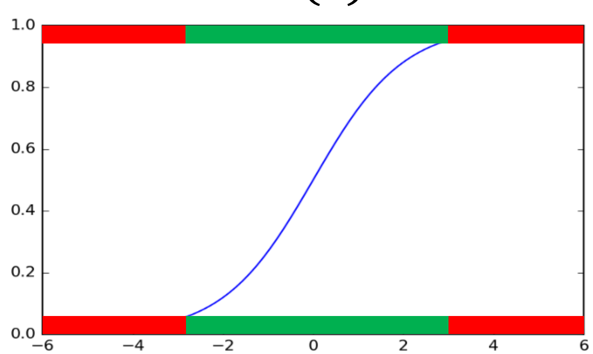
Very simple fix:
If $\sigma'(s)$ < 0.01 → $\sigma'(s) = 0.01$
We can set this limit arbitrarily

→ This ensures that there will always be a non-saturated gradient

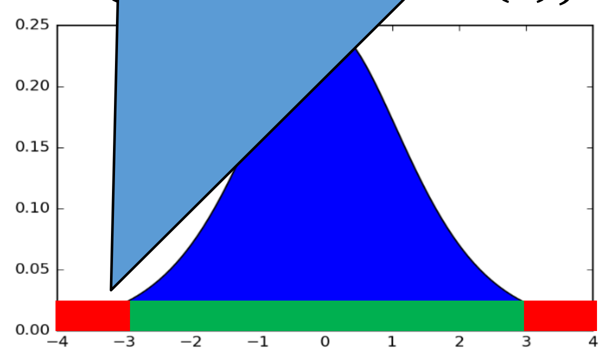
Class: Are there any cons to this approach?

Fixes

$\sigma(s)$



$\sigma'($ … $- \sigma(s))$

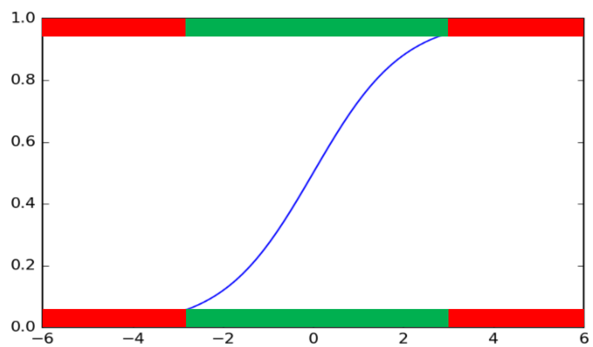


Simple fix: Clip gradients

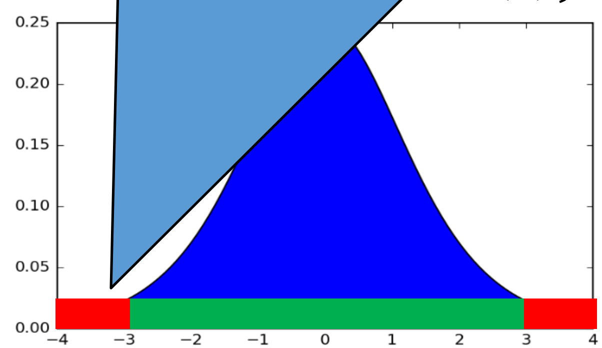
Class: Are there any cons to this approach?

- If the threshold is set too low, we recover the original sigmoid with the vanishing gradients
- If the threshold is set too high, we get only the linear region and lose the benefit of a nonlinearity,
    - E.g. The matrices can be multiplied out into a single linear layer, thus losing effective depth

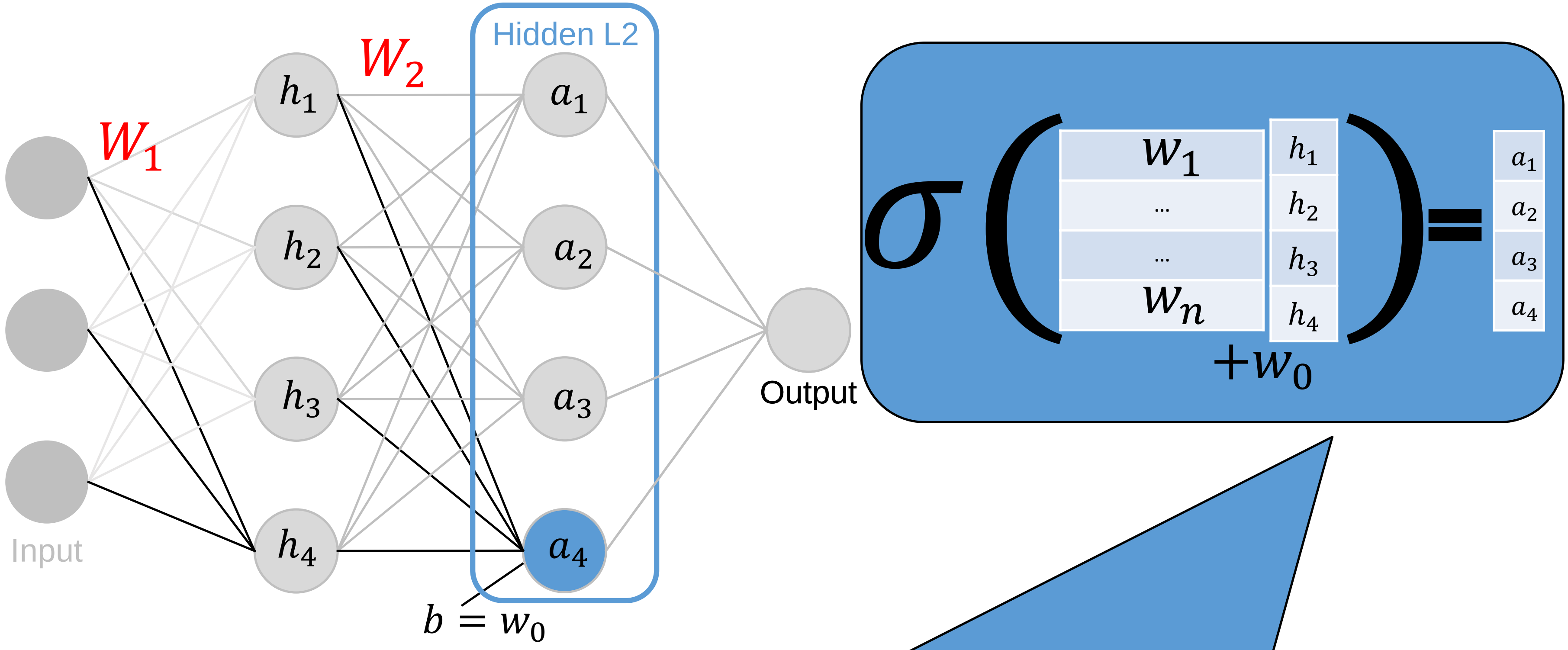
$\sigma(s)$



$\sigma'($ … $- \sigma(s))$



Weight constraints



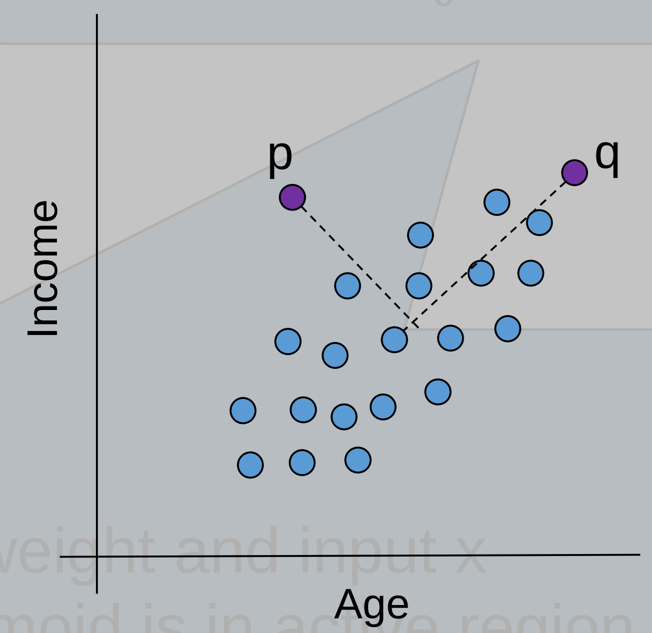
Recall:

- Weights are row vectors
- Each activation is a dot product of weight and input x
- We want $-3 \leq w \cdot x \leq 3$ so that sigmoid is in active region

Bases of a weight matrix

Some insight:

- Weights (rows) of matrix act as a set of basis vectors
- Multiplying ***x*** by the matrix stretches it according to the basis vectors of the matrix



Recall:

- Weights are row vectors
- Each activation is a dot product of weight and input x
- We want $-3 \leq w \cdot x \leq 3$ so that sigmoid is in active region

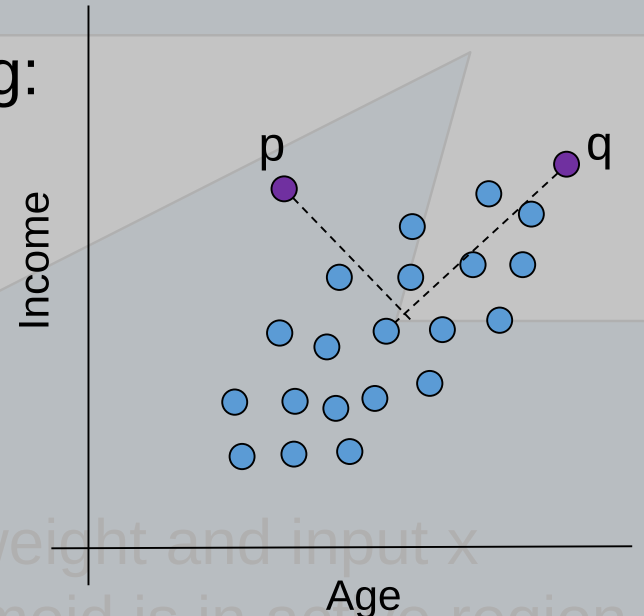
Eigenvectors of W

Some insight:

- Weights (rows) of matrix act as a set of basis vectors
- Multiplying ***x*** by the matrix stretches it according to the basis vectors of the matrix

- We can find these bases by solving:

$Ax = \lambda x$



Recall:

- Weights are row vectors
- Each activation is a dot product of weight and input x
- We want $-3 \leq w \cdot x \leq 3$ so that sigmoid is in active region

NORTHWESTERN UNIVERSITY

Eigenvectors of W

Some insight:

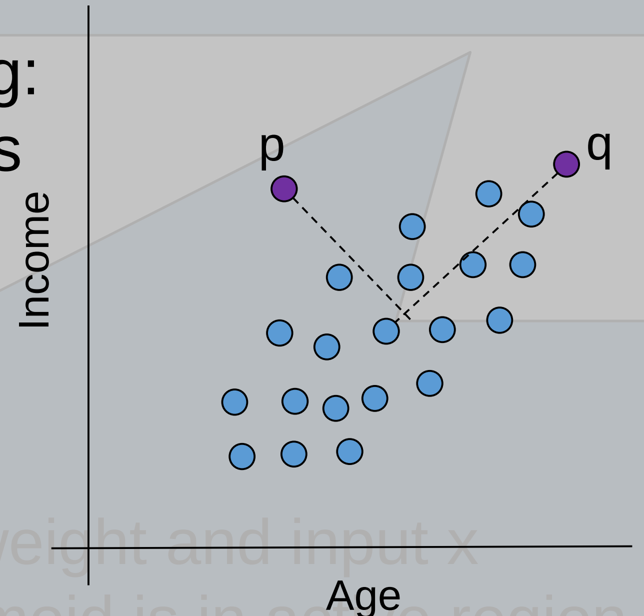
- Weights (rows) of matrix act as a set of basis vectors
- Multiplying ***x*** by the matrix stretches it according to the basis vectors of the matrix

- We can find these bases by solving:

$Ax = \lambda x$ → Eigenvalues, eigenvectors

v_1
…
…
v_n

λ_1			
	λ_2		
		…	
			λ_n



NORTHWESTERN UNIVERSITY

Eigenvectors of W

Some insight:

- Weights (rows) of matrix act as a set of basis vectors
- Multiplying ***x*** by the matrix stretches it according to the basis vectors of the matrix

- We can find these bases by solving:

$Ax = \lambda x$ → Eigenvalues, eigenvectors

$$\begin{bmatrix} v_1 \\ \cdots \\ \cdots \\ v_n \end{bmatrix} \quad \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \cdots & \\ & & & \lambda_n \end{bmatrix}$$

Each λ_i acts as a stretch factor. If $\lambda_i < 1$ or $\lambda_i > 1$, then resulting vector shrinks or grows.

NORTHWESTERN UNIVERSITY

Eigenvectors of W

Some insight:

- Weights (rows) of matrix act as a set of basis vectors
- Multiplying ***x*** by the matrix stretches it according to the basis vectors of the matrix

- We can find these bases by solving:

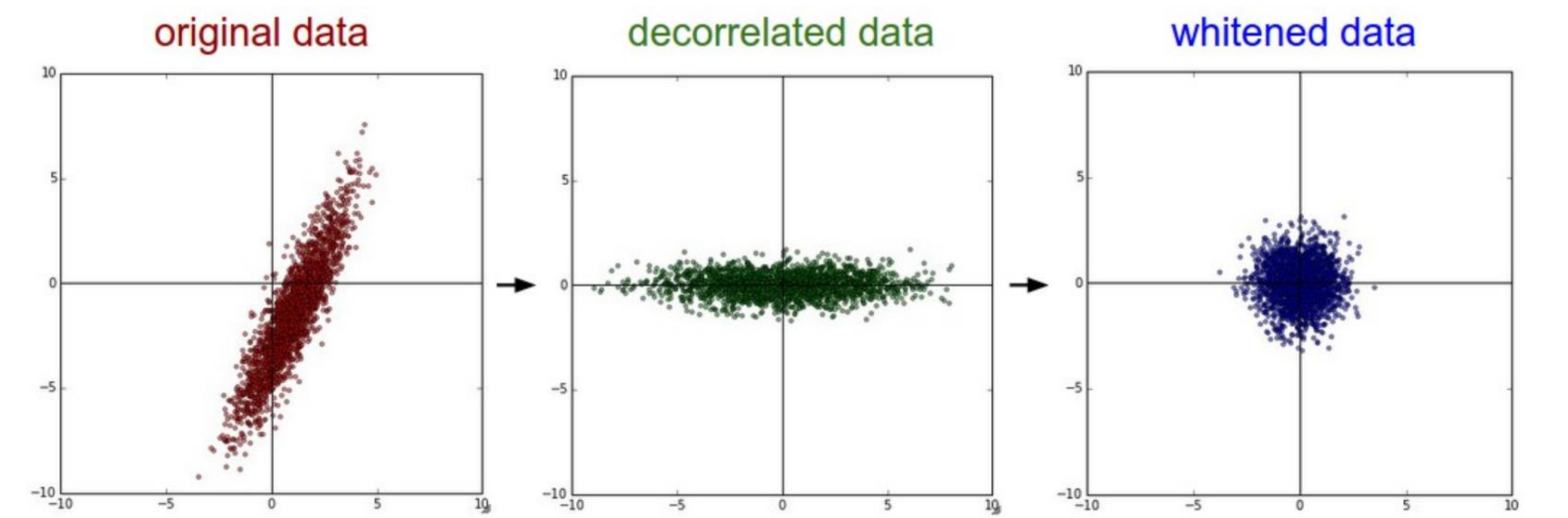
$Ax = \lambda x$ → Eigenvalues, eigenvectors

$$\begin{bmatrix} v_1 \\ \cdots \\ \cdots \\ v_n \end{bmatrix} \quad \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \cdots & \\ & & & \lambda_n \end{bmatrix}$$

Each λ_i acts as a stretch factor. If $\lambda_i < 1$ or $\lambda_i > 1$, then resulting vector shrinks or grows. We want $\lambda_i \approx 1$!

NORTHWESTERN UNIVERSITY

Keeping sigmoids in active region



(data has diagonal covariance matrix)

(covariance matrix is the identity matrix)

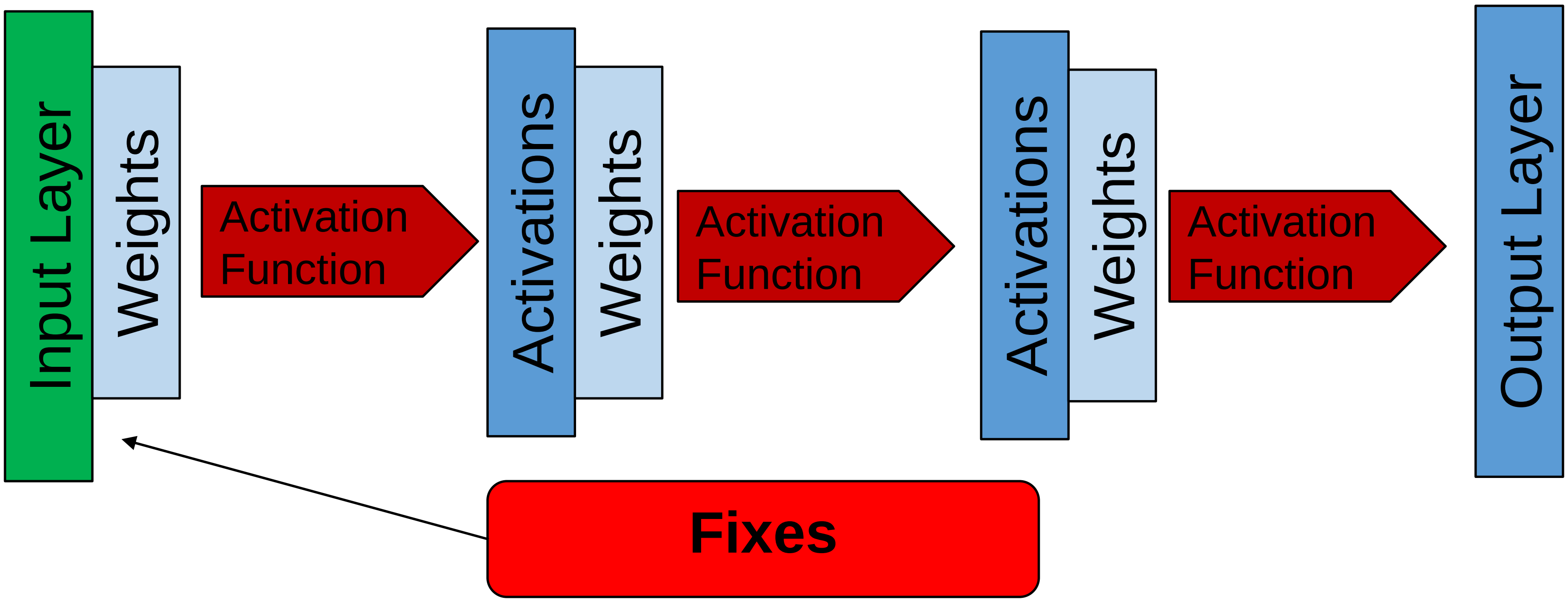
Practical tip:

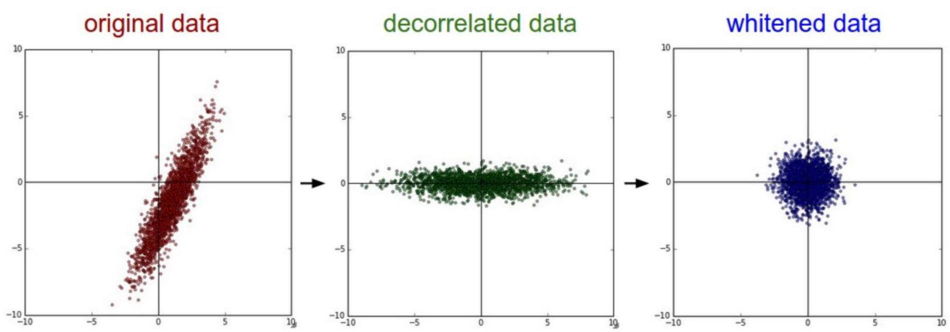
- Subtract data mean(bias=0), Divide by standard deviation
- Whiten data (PCA/ZCA) if possible

See: http://ufldl.stanford.edu/wiki/index.php/Exercise:PCA_and_Whitening

- Keeps activations at each layer in an active region

Keeping sigmoids in active region



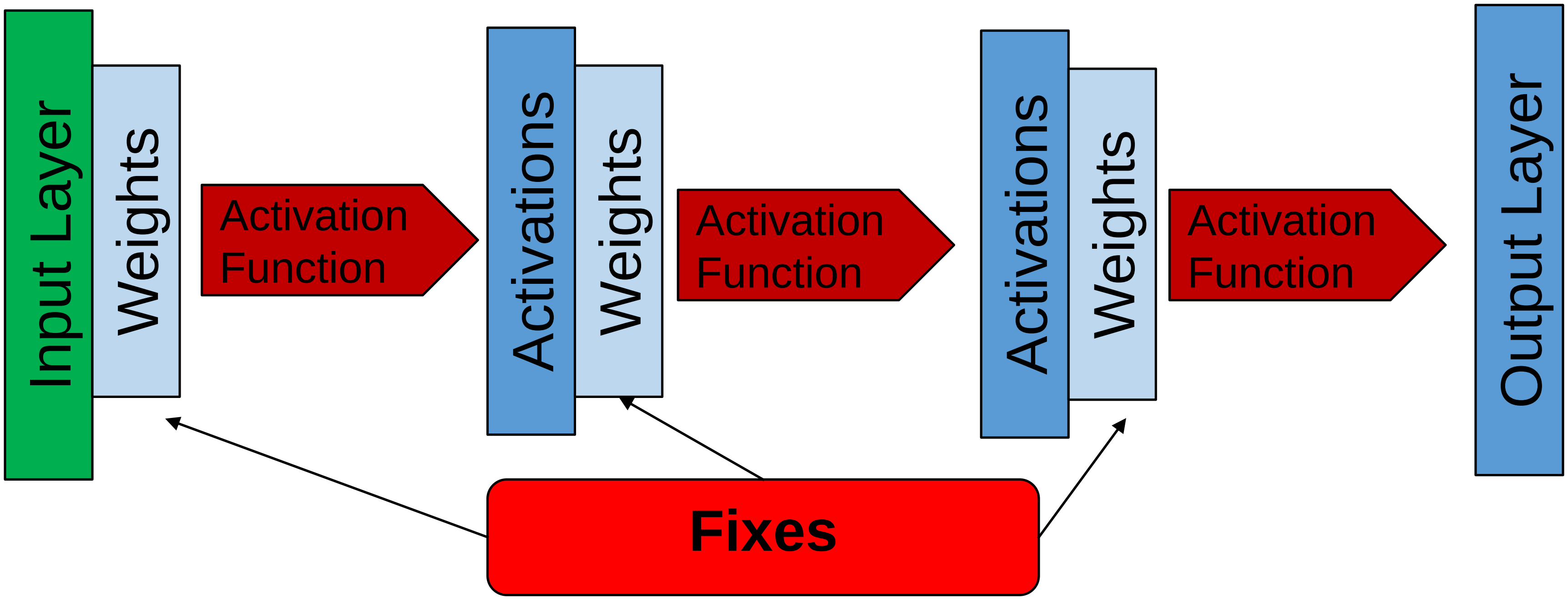


(data has diagonal covariance matrix)

(covariance matrix is the identity matrix)

NORTHWESTERN UNIVERSITY

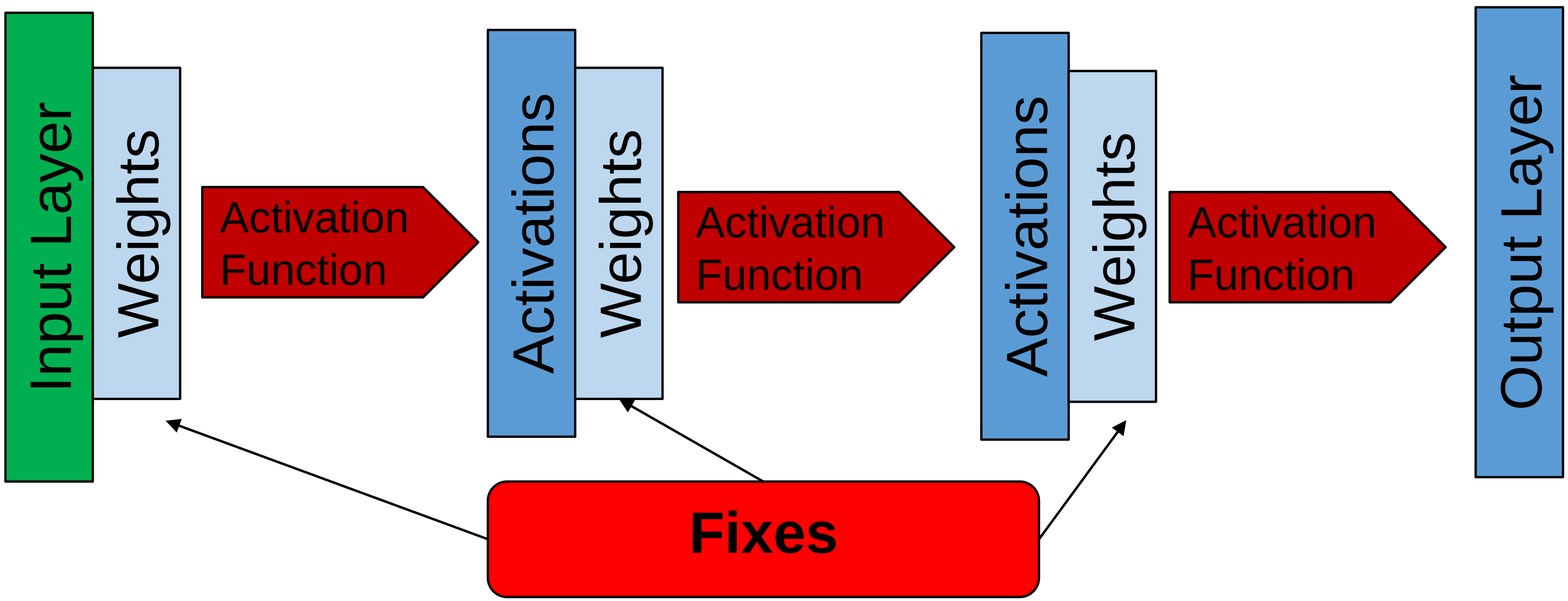
Keeping sigmoids in active region



- What about weights?
- Weights are usually initialized randomly
- Poor initialization can lead to small/large eigenvalues and shrinkage/explosion of gradients
- **Practical fix:** Initialize weights with $\mu = 0, \sigma \ll 1$

NORTHWESTERN UNIVERSITY

Keeping sigmoids in active region



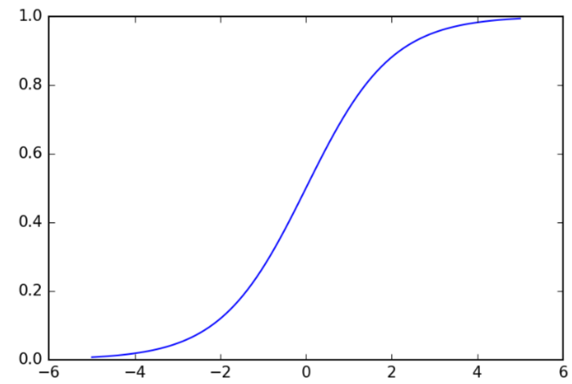
- $f(w,x) = \sum_i w_i x_i$. Scale of output is determined by fan-in. Suppose $w_i x_i$ is Gaussian distributed with μ_i, σ_i. $\sigma^2 = \sum_{i=1}^{n} \sigma_i^2 = n\sigma^2$. We overshot by $\sqrt(n)$.
- **Theory:** Normalize inputs by fan-in $\sigma_i/\sqrt(n)$ to maintain unit variance in output.

NORTHWESTERN UNIVERSITY

Nature of the logistic activation

- $0 \leq \sigma(\sum_i w_i x_i) \leq 1$

$\sigma(s)$

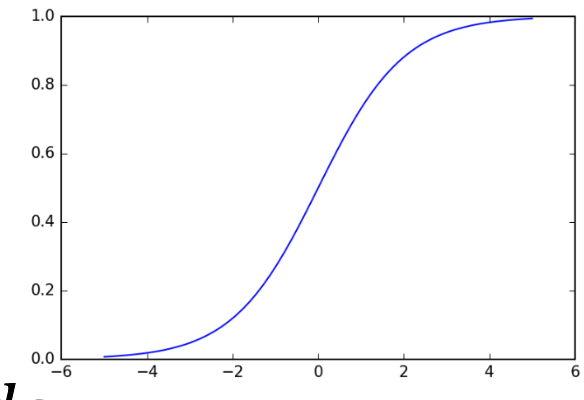


NORTHWESTERN UNIVERSITY

Nature of the logistic activation

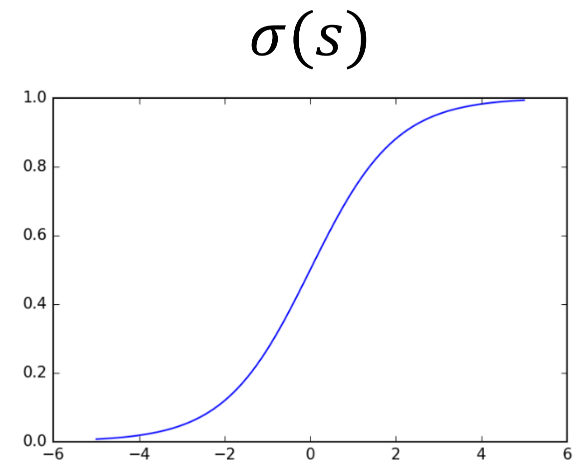
- $0 \leq \sigma(\sum_i w_i x_i) \leq 1$
- Inputs at next layer are always positive
- How does that constrain the gradients? $\sigma'(s) = \sigma(s)\big(1 - \sigma(s)\big)ds$

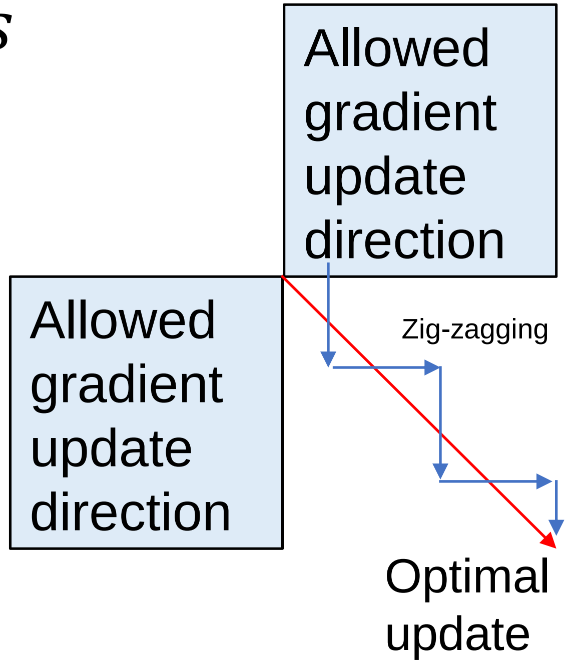
$\sigma(s)$



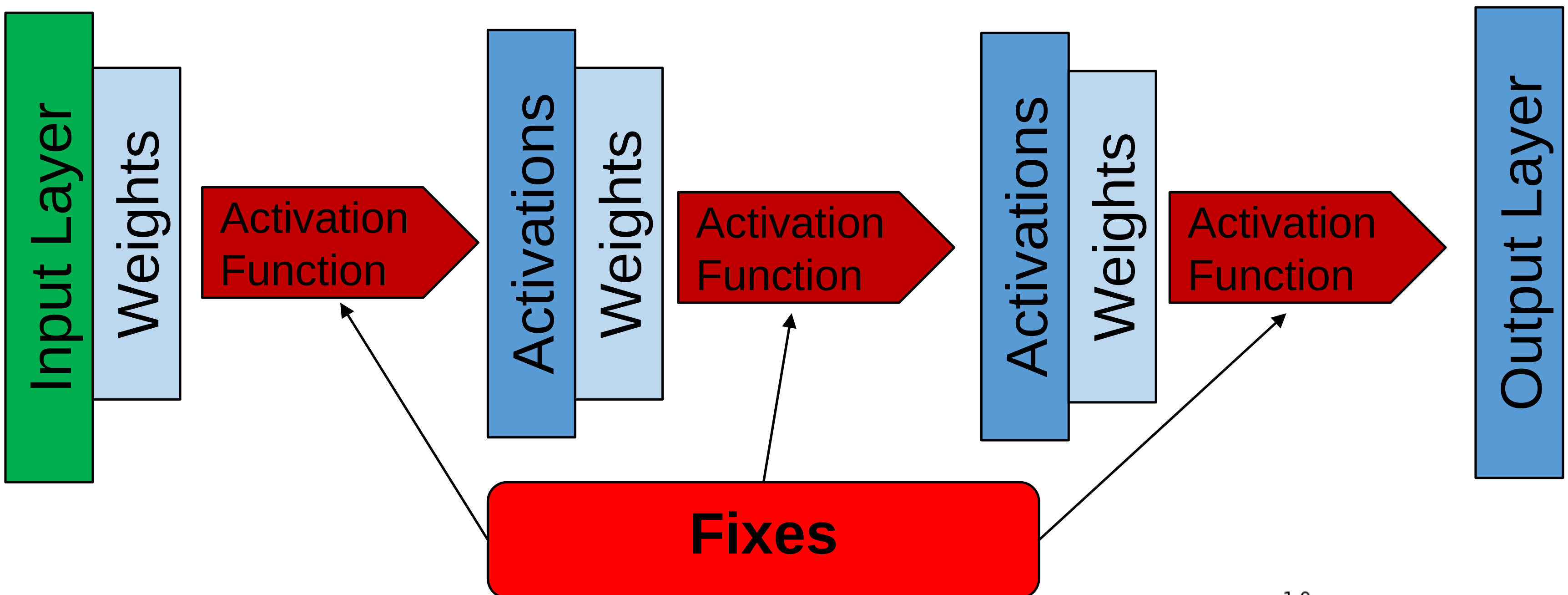
Nature of the logistic activation

- $0 \leq \sigma(\sum_i w_i x_i) \leq 1$
- Inputs at next layer are always positive
- How does that constrain the gradients? $\sigma'(s) = \sigma(s)\big(1 - \sigma(s)\big)ds$
- Depending on the sign of ds, gradients are always all positive or all negative





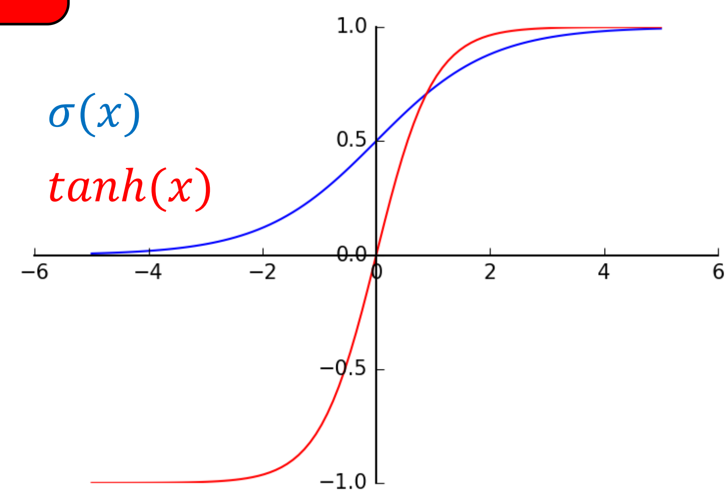
Tanh(x)



Advice: Choose a mean-centered Activation function

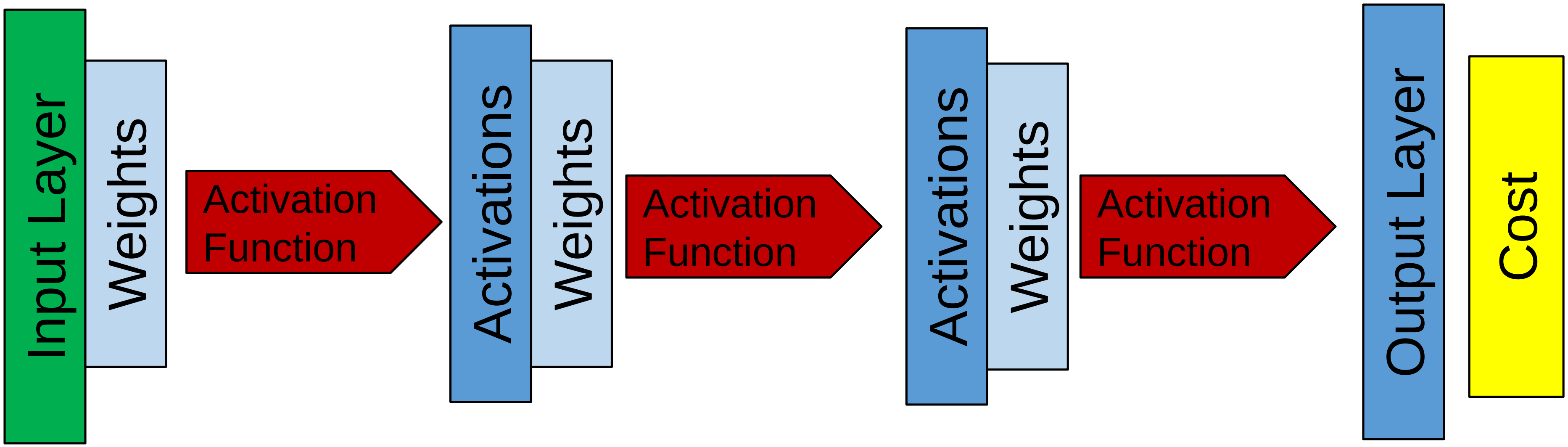
$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = 2\sigma(2x) - 1$$

- Addresses data centering issue (if input is zero mean)
- Addresses zigzag issue (activation can be negative)
- Does not address vanishing gradients



NORTHWESTERN UNIVERSITY

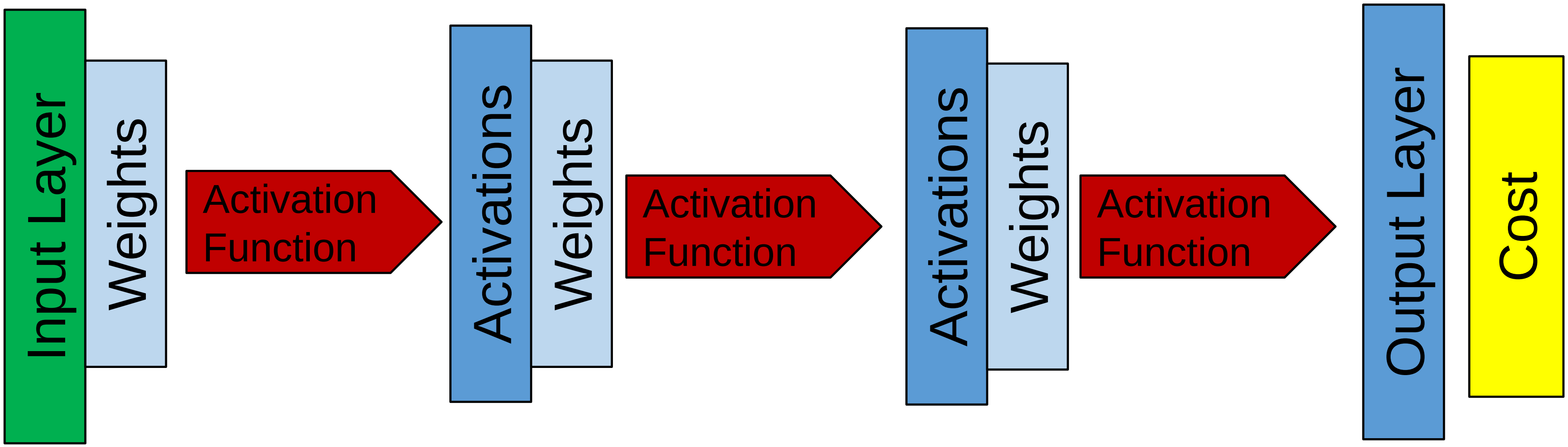
Cost function



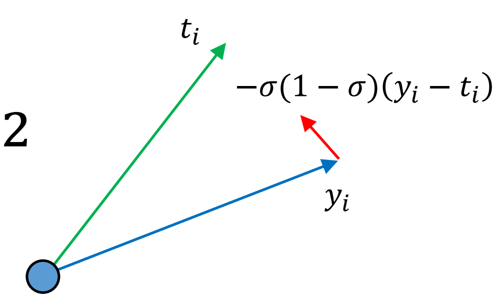
- What about the cost function?
- We’ve been using $MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - t_i)^2$
- Derivative of MSE: $2(y_i - t_i)\ dy$

NORTHWESTERN UNIVERSITY

Cost function



- What about the cost function?
- We've been using $MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - t_i)^2$
- Derivative of MSE: $\boxed{2(y_i - t_i)}\boxed{dy}$

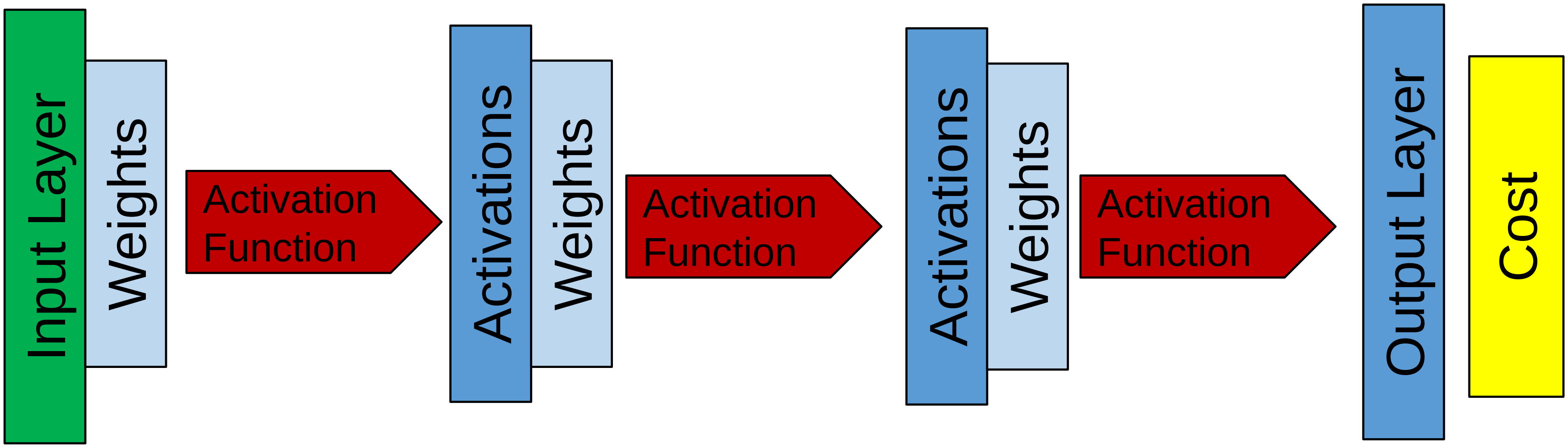


Measure of disagreement between prediction y and target t

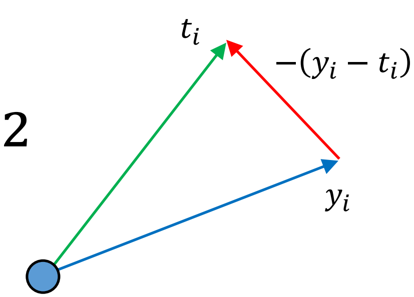
$dy = \sigma(1-\sigma)$
Derivative of activation function. Remember that this is **SLOW**.

NORTHWESTERN UNIVERSITY

Cost function



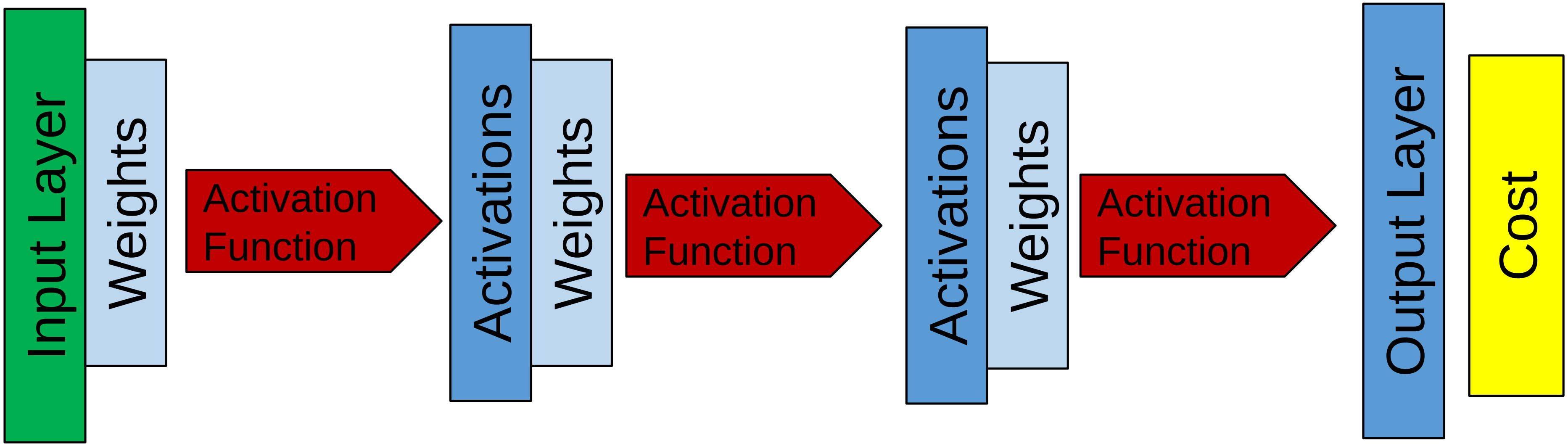
- What about the cost function?
- We've been using $MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - t_i)^2$
- Derivative of MSE: $2(y_i - t_i)\cancel{dy}$



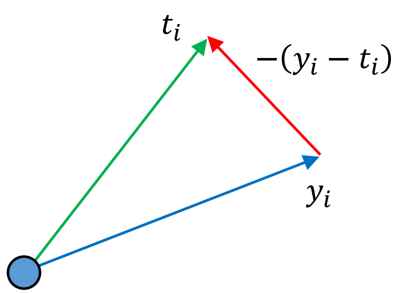
Measure of disagreement between prediction y and target t. Let's KEEP this.

And find a way to eliminate or at least speed up this…

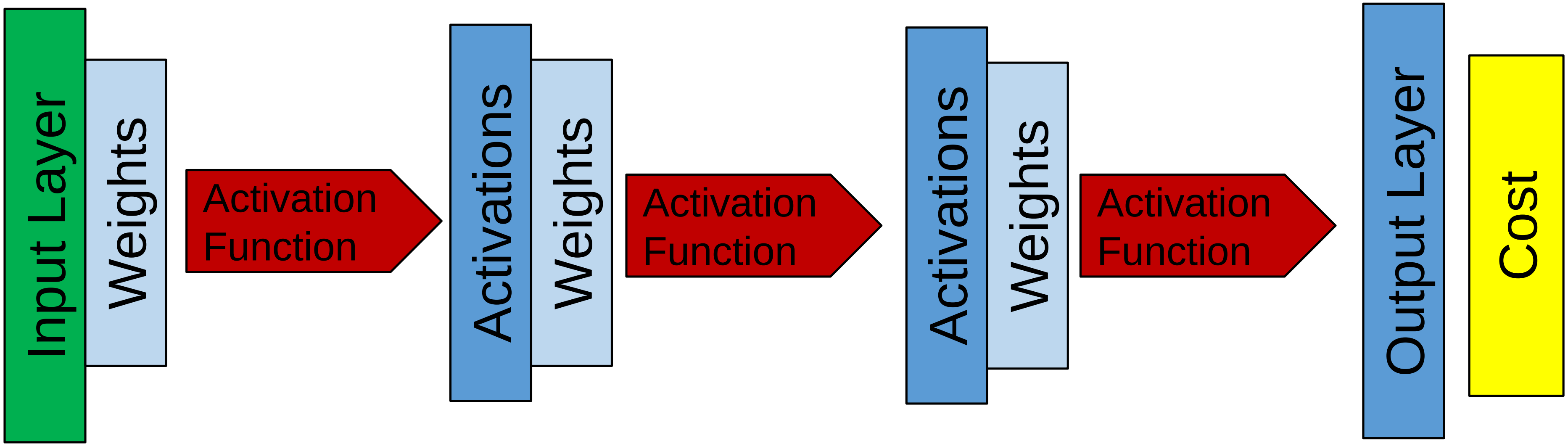
Keeping sigmoids in active region



- Desired gradient: $\partial E = y - t$



Keeping sigmoids in active region



- Desired gradient: $\partial E = y - t$
- Cross entropy cost: $E = -t \ln y \, - (1 - t)\ln(1 - y)$

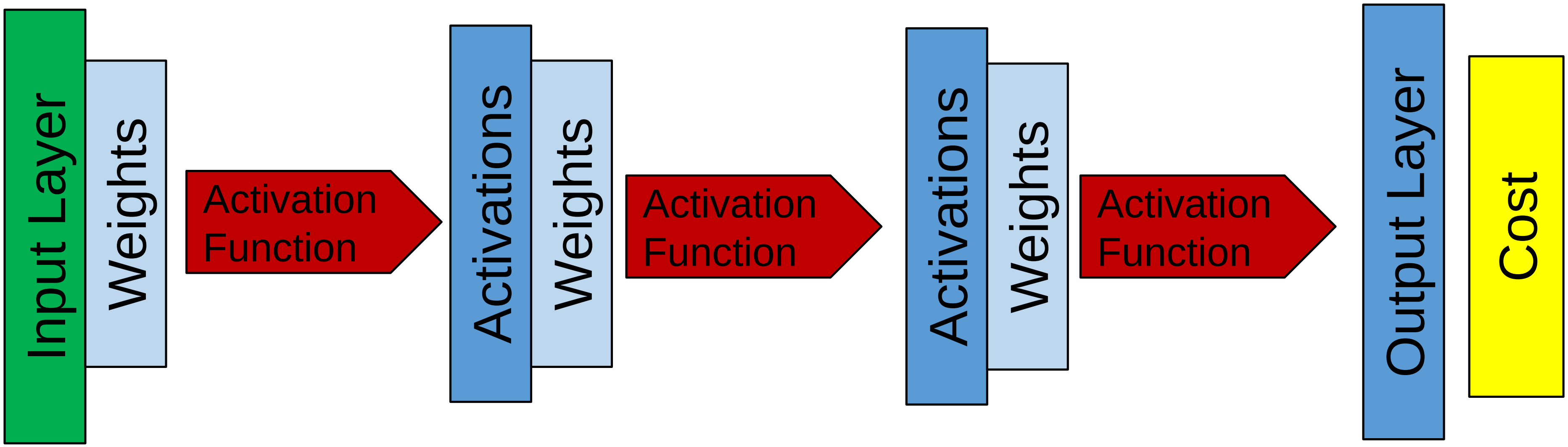
NORTHWESTERN UNIVERSITY

Cross entropy in depth

- $E = -\frac{1}{n}\sum_x t \ln \sigma(z) + (1-t)\ln(1-\sigma(z))$
- $\frac{\partial E}{\partial w_j} = -\frac{1}{n}\sum_x \left(\frac{t}{\sigma(z)} - \frac{1-t}{1-\sigma(z)}\right)\sigma'(z)x_j$
- $= -\frac{1}{n}\sum_x \left(\frac{t(1-\sigma(z))-(1-t)\sigma(z)}{\sigma(z)(1-\sigma(z))}\right)\sigma(z)(1-\sigma(z))x_j$
- $= -\frac{1}{n}\sum_x \left(\frac{t \cancel{-t\sigma(z)} - \sigma(z) + \cancel{t\sigma(z)}}{\cancel{\sigma(z)(1-\sigma(z))}}\right)\cancel{\sigma(z)(1-\sigma(z))}x_j$
- $= \frac{1}{n}\sum_x(\sigma(z)-t)x_j$
- → Slow $\sigma'(z)$ cancels out!
- See Nielsen Ch 3 for details: http://neuralnetworksanddeeplearning.com/chap3.html
- Also read Nielsen Ch 3 to learn where cross entropy comes from

NORTHWESTERN UNIVERSITY

Keeping sigmoids in active region



- Cross entropy cost: $E = -t \ln y - (1-t)\ln(1-y)$
- Cross entropy gradient: $\partial E = y - t$

  MSE gradient: $\partial E = (y-t)\boxed{dy}$

  $dy = \sigma(1-\sigma)$
  Derivative of activation function
- Gradient is proportional to difference
- No learning slowdown due to sigmoid gradients

NORTHWESTERN UNIVERSITY

But how did we get cross entropy?

- Using $\sigma'(z) = \sigma(z)\big(1 - \sigma(z)\big) = a(1 - a)$
- We'd like the $a(1 - a)$ to cancel out of the cost, so
- $\frac{\partial E}{\partial a} = \frac{a-y}{a(1-a)}$

a-y is just the discrepancy between the prediction y and actual value t. a(1-a) cancels out with the derivative.

NORTHWESTERN UNIVERSITY

But how did we get cross entropy?

- Using $\sigma'(z) = \sigma(z)\big(1-\sigma(z)\big) = a(1-a)$
- We'd like the $a(1-a)$ to cancel out of the cost to get the gradient to be $a-t$, so
- $\frac{\partial E}{\partial a} = \frac{a-t}{a(1-a)} = \frac{a+at-at-t}{a(1-a)} = \frac{a(1-t)-t(1-a)}{a(1-a)} = \frac{1-t}{1-a} - \frac{t}{a}$
- $\int \frac{\partial E}{\partial a} da = \int \frac{a-t}{a(1-a)} da = \int \frac{1-t}{1-a} da - \int \frac{t}{a} da$

$= -t \ln a - (1-t)\ln(1-a)$

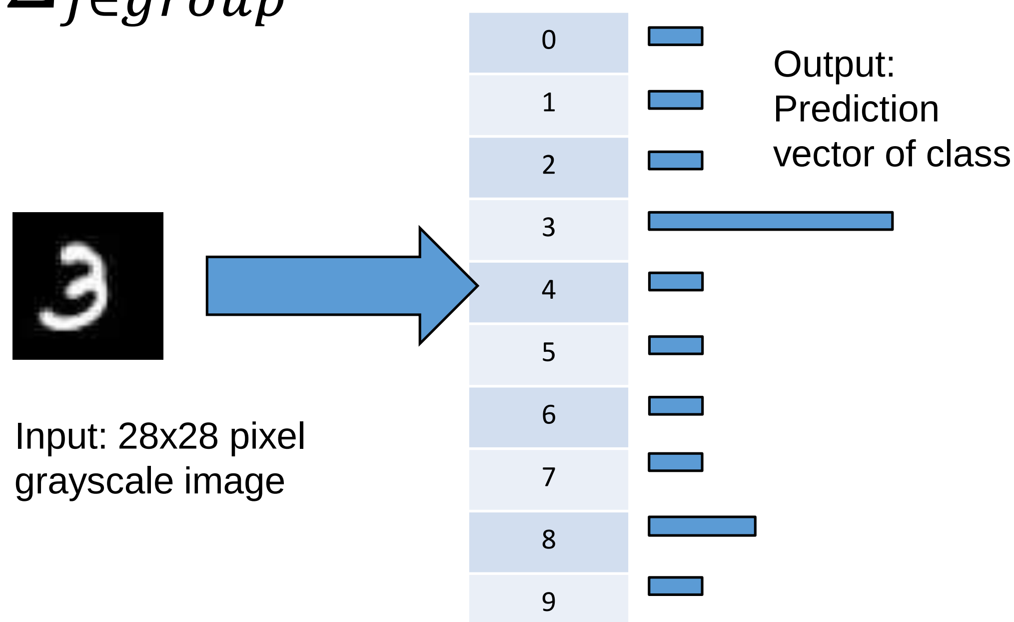
NORTHWESTERN UNIVERSITY

Softmax

- Recall sigmoid function: $\sigma(z) = \frac{1}{1+e^{-z}}$
- Softmax is a generalization of the sigmoid:

$$y_i = \frac{e^{z_i}}{\sum_{j \in group} e^{z_j}}$$

- Softmax generalizes 2-class logistic regression/sigmoids to n classes
- Suitable for mutually exclusive classes

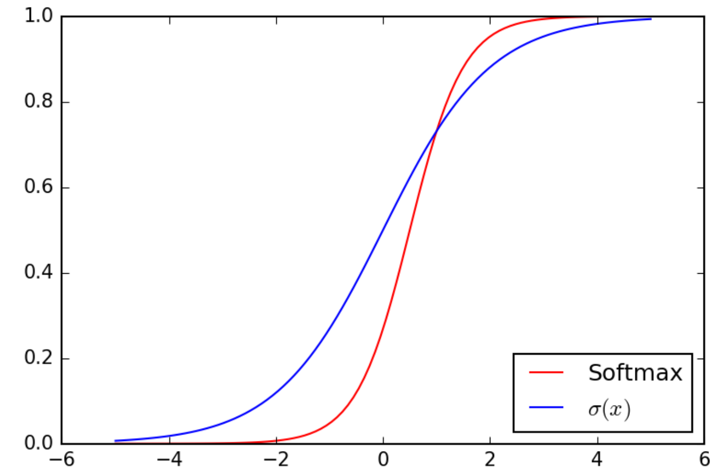


Softmax

- Recall sigmoid function: $\sigma(z) = \frac{1}{1+e^{-z}}$
- Softmax is a generalization of the sigmoid:

$$y_i = \frac{e^{z_i}}{\sum_{j \in group} e^{z_j}}$$

- Consider for two classes with probabilities (z, 1-z):
- $y_i = \frac{e^z}{e^z + e^{1-z}}$



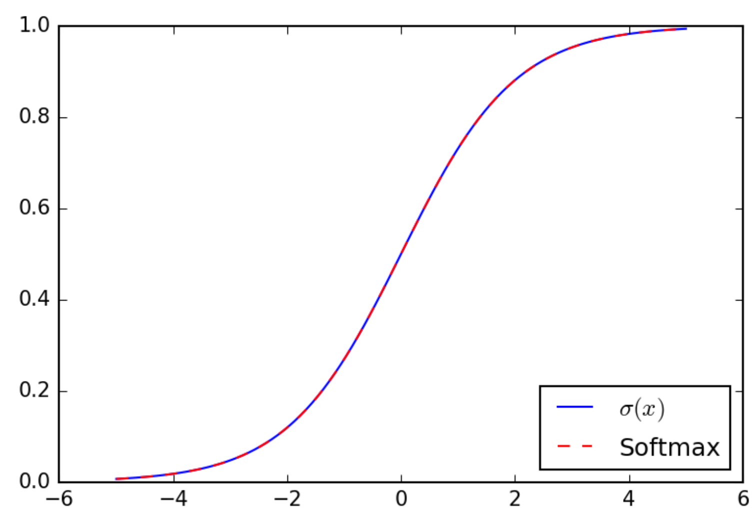
Softmax

- Recall sigmoid function: $\sigma(z) = \frac{1}{1+e^{-z}}$
- Softmax is a generalization of the sigmoid:

$$y_i = \frac{e^{z_i}}{\sum_{j \in group} e^{z_j}}$$

- Consider for two classes with probabilities (z, 1-z):
- $y_i = \frac{e^{z}}{e^{z}+e^{1-z}}$
- With $\hat{z} = (z+1)/2$, we recover the

Sigmoid function.



Softmax

- Recall sigmoid function: $\sigma(z) = \frac{1}{1+e^{-z}}$
- Softmax is a generalization of the sigmoid:

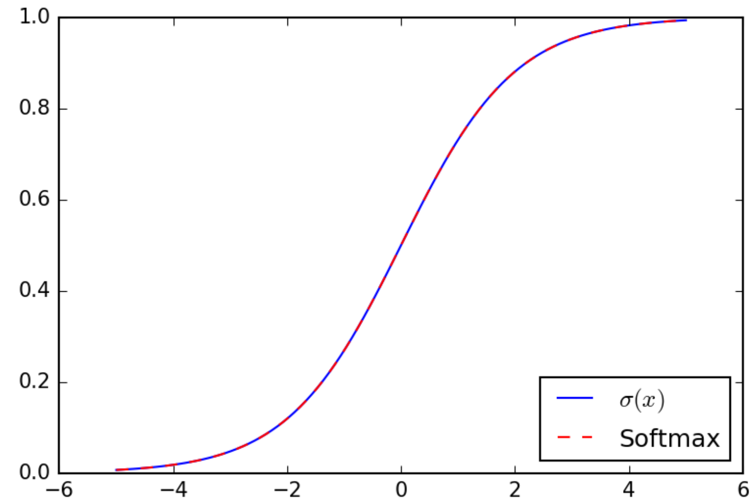
$$y_i = \frac{e^{z_i}}{\sum_{j \in group} e^{z_j}}$$

- Consider for two classes with probabilities (z, 1-z):
- $y_i = \frac{e^z}{e^z + e^{1-z}}$
- With $\hat{z} = (z+1)/2$, we recover the

Sigmoid function.

- As we found with the sigmoid,

$$y_i' = y_i(1 - y_i)$$



Softmax

Cross entropy is the "best" cost function to use for the softmax:

$$C_j = -\sum_j t_j \ln y_j$$

For the familiar two-class case:

$$C = \boxed{-t_j \ln y_j} - \boxed{(1-t)\ln(1-y)}$$

Class 1

Class 2 = 1 - Class 1

Sigmoid function.

- As we found with the sigmoid,

$$y_i' = y_i(1-y_i)$$

- Recall sigmoid function: $\sigma(z) = \frac{1}{1+e^{-z}}$
- Softmax is a generalization of the sigmoid:

$$y_i = \frac{e^{z_i}}{\sum_{j \in group} e^{z_j}}$$

- Consider for two classes with probabilities (z, 1-z):
- $y_i = \frac{e^z}{e^z + e^{1-z}}$
- With z'=(z+1)/2, we recover the

Sigmoid function.

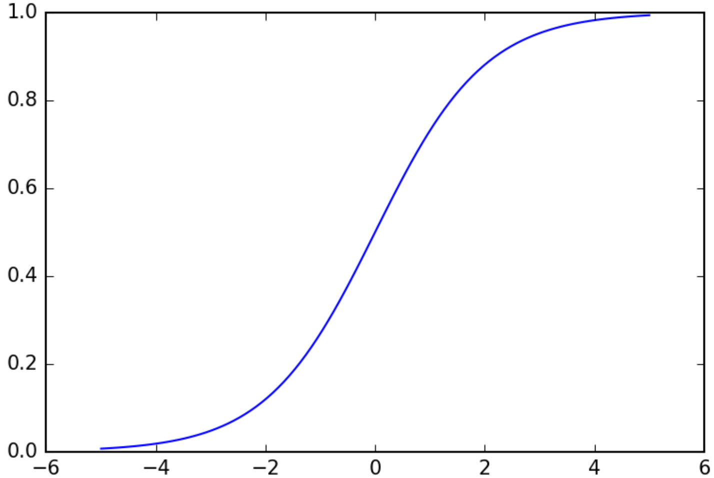
- As we found with the sigmoid,

$$y_i' = y_i(1-y_i)$$

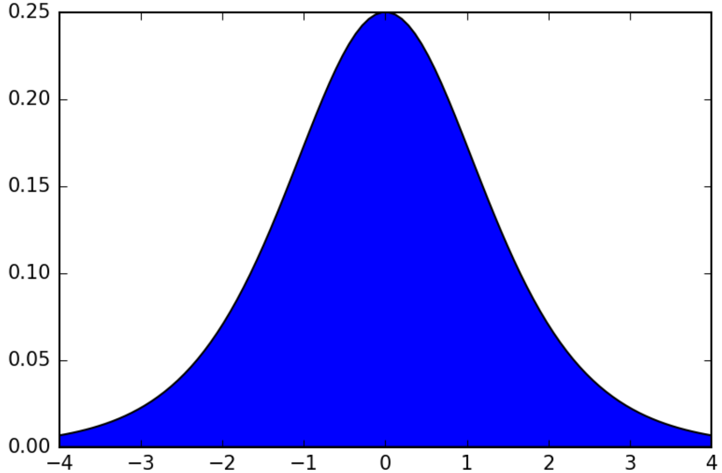
Recall: Trouble in Sigmoid land

- Sigmoids are relatively effective in active region

$$\sigma(s)$$



$$\sigma'(s) = \sigma(s)(1 - \sigma(s))$$



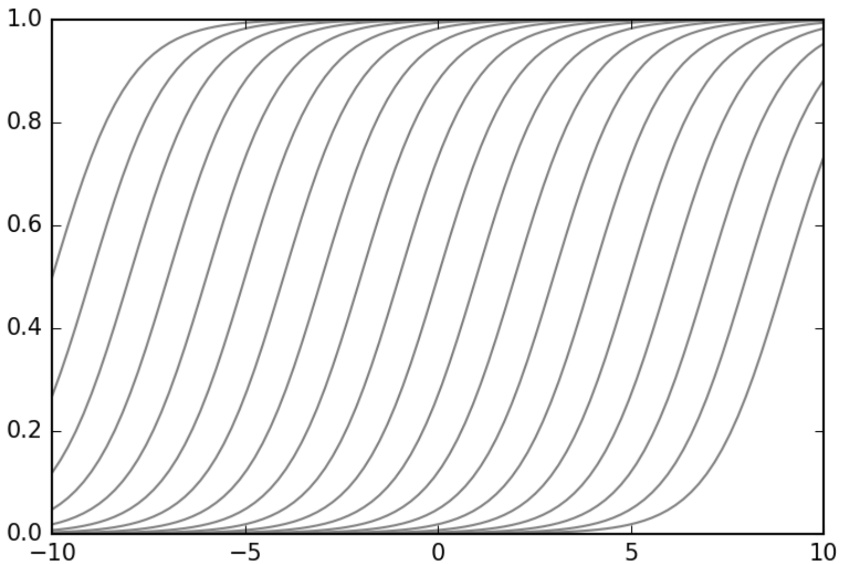
NORTHWESTERN UNIVERSITY

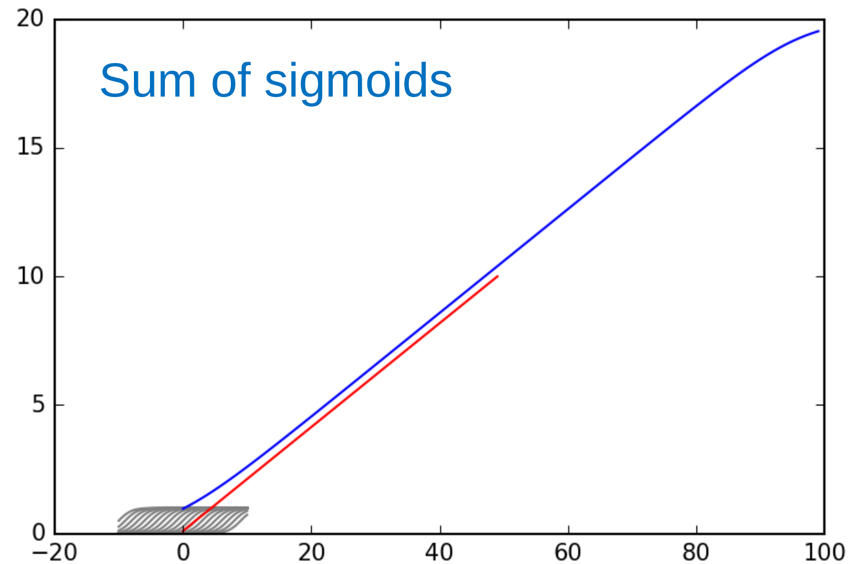
Recall: Trouble in Sigmoid land

- Sigmoids are relatively effective in active region
- What if we replicate the sigmoids with different biases?

$$\sigma(s) + \sigma(s-1) + \sigma(s-i) + \cdots + \sigma(s-n)$$

- And share their weights w_i?





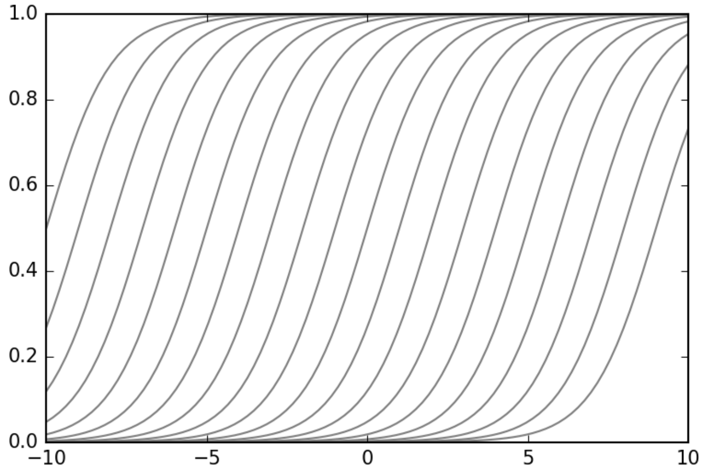
NORTHWESTERN UNIVERSITY

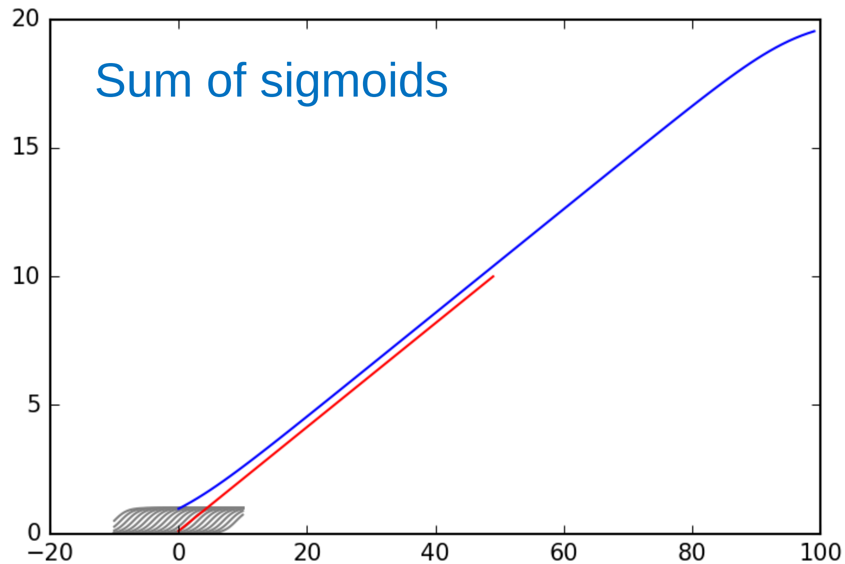
Recall: Trouble in Sigmoid land

- What if we replicate the sigmoids with different biases?

$$\sum_{i=1}^{\infty} \sigma(s + 0.5 - i) \approx \log(1 + e^{s}) \approx \max(0, s + noise)$$

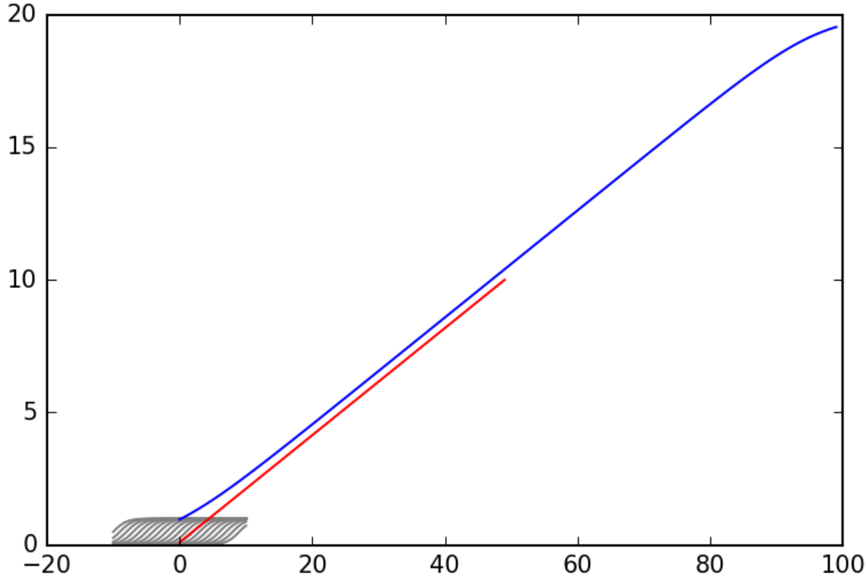
- Called rectified linear unit: ReLU
- Easy to approximate





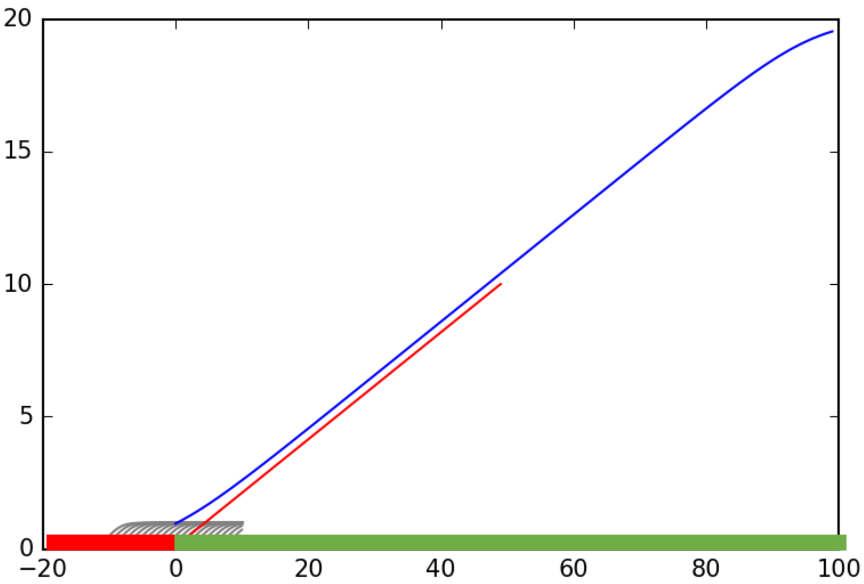
Nice try, but….

- $f(s) = \max(0, s)$
- $\frac{\partial}{\partial s} f = \begin{cases} 0, & s \leq 0 \\ 1 * ds, & s > 0 \end{cases}$



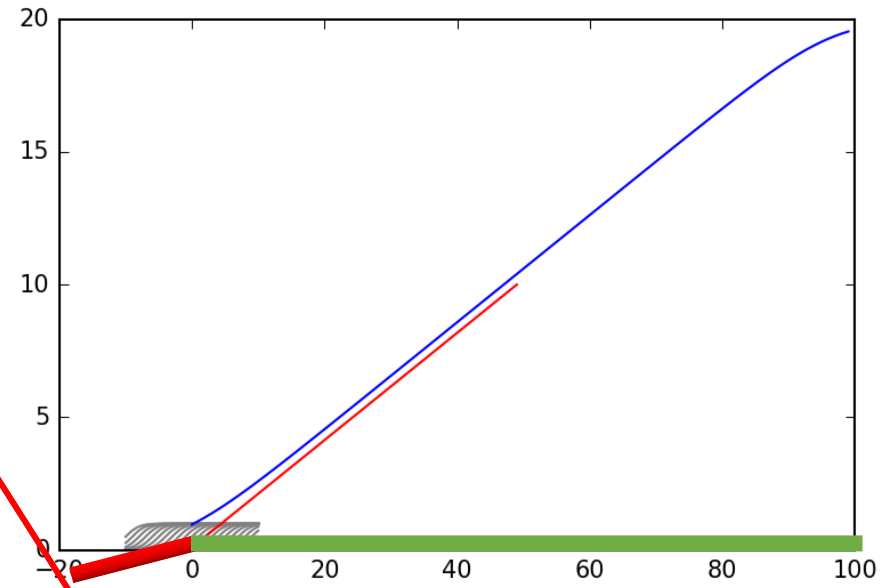
Nice try, but….

- $f(s) = \max(0, s)$
- $\frac{\partial}{\partial s} f = \begin{cases} 0, & s \leq 0 \\ 1 * ds, & s > 0 \end{cases}$
- ReLU neurons also suffer from vanishing gradients
- Neurons can "die" if they get a negative activation and never recover



Nice try, but....

- $f(s) = \max(0, s)$
- $\frac{\partial}{\partial s} f = \begin{cases} \boxed{0.01s, \quad s \le 0} \\ 1 * ds, \quad s > 0 \end{cases}$
- ReLU neurons also suffer from vanishing gradients
- Neurons can "die" if they get a negative activation and never recover



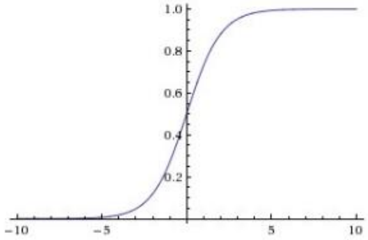
Fix:
Clip 0 gradient to something small so neuron can "recover"
Called: Leaky ReLU

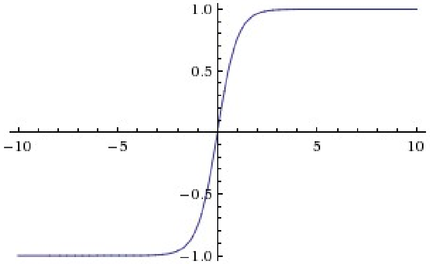
Delving Deep into rectifiers. Kaiming He, MSR 2015

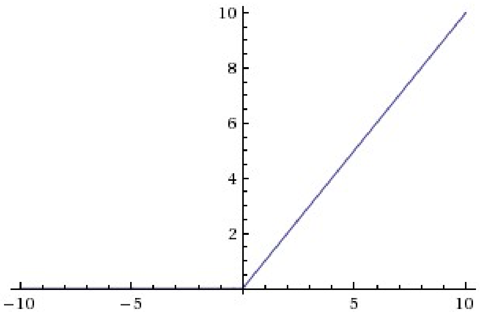
NORTHWESTERN UNIVERSITY

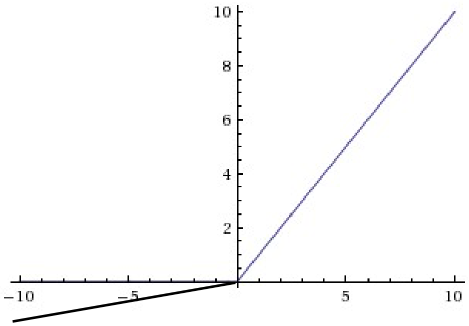
Activation Functions

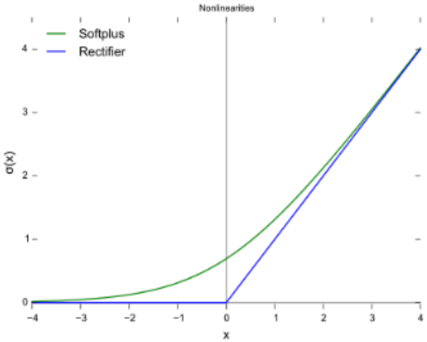
- Sigmoid
- Hyperbolic tangent
- ReLU = max(0,x)
  - Leaky ReLU
  - Soft ReLU – $\ln(1 + e^{x})$









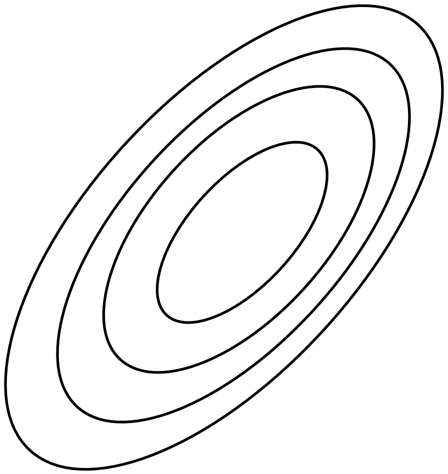


[Mark's paper] Activation ensembles: https://arxiv.org/abs/1702.07790

NORTHWESTERN UNIVERSITY

Optimization Visualized

- To find the best weights, we minimize the cost function using optimization

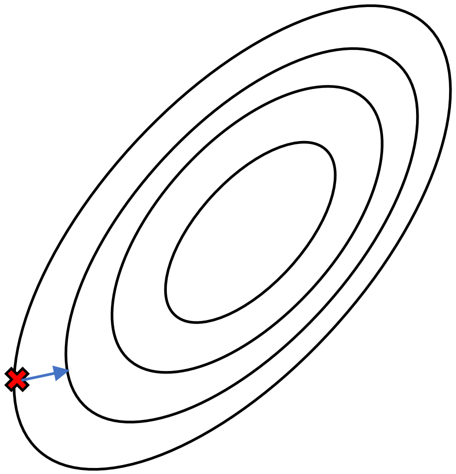


Cost function is usually MSE or cross entropy

NORTHWESTERN UNIVERSITY

Optimization Visualized

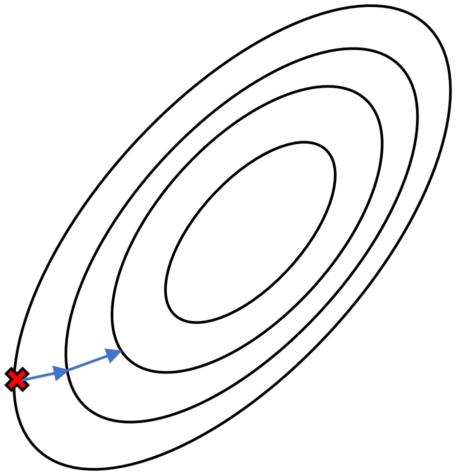
- To find the best weights, we minimize the cost function using optimization
- Optimization starts from an initial point



NORTHWESTERN UNIVERSITY

Optimization Visualized

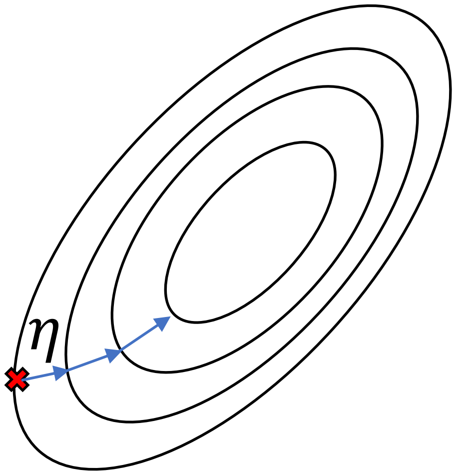
- To find the best weights, we minimize the cost function using optimization
- Optimization starts from an initial point
- Each iteration attempts to make progress towards global minima



NORTHWESTERN UNIVERSITY

Optimization Visualized

- To find the best weights, we minimize the cost function using optimization
- Optimization starts from an initial point
- Each iteration attempts to make progress towards global minima
- Usually the fastest way is to take a step η in the direction of the steepest descent, called the gradient



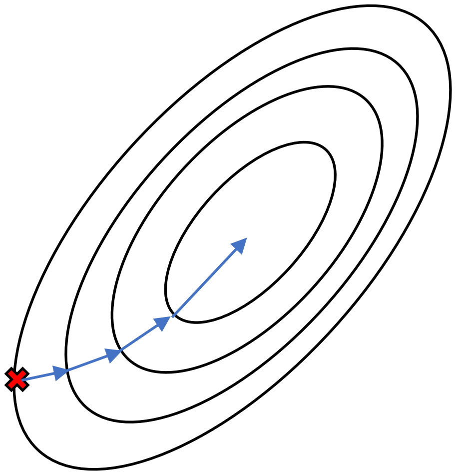
η is known as the learning rate

$$w_{i+1} = w_i - \eta \nabla w_i$$

NORTHWESTERN UNIVERSITY

Optimization Visualized

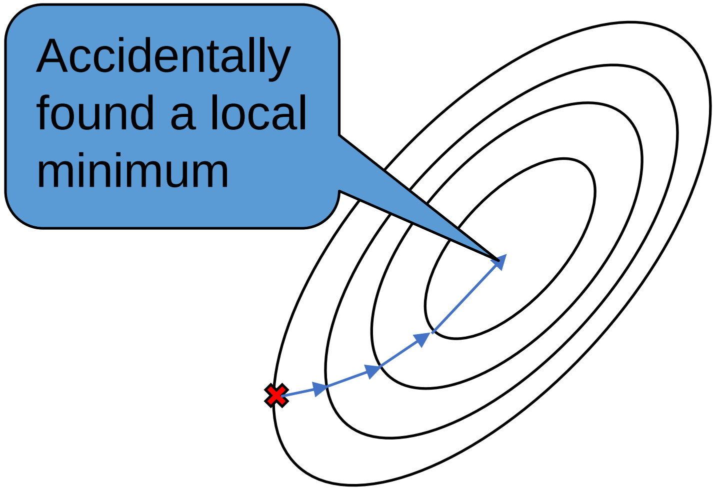
- To find the best weights, we minimize the cost function using optimization
- Optimization starts from an initial point
- Each iteration attempts to make progress towards global minima
- Usually the fastest way is to take a step η in the direction of the steepest descent, called the gradient
- This leads to a path perpendicular to the contours

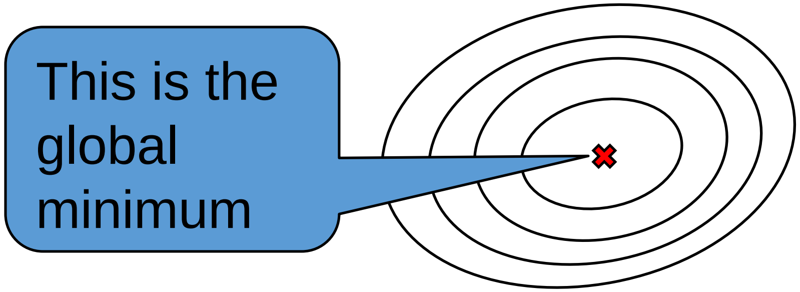


NORTHWESTERN UNIVERSITY

Optimization Visualized

- To find the best weights, we minimize the cost function using optimization
- Optimization starts from an initial point
- Each iteration attempts to make progress towards global minima
- Usually the fastest way is to take a step η in the direction of the steepest descent, called the gradient
- This leads to a path perpendicular to the contours
- However, optimization can have some pitfalls…
  - Local minima

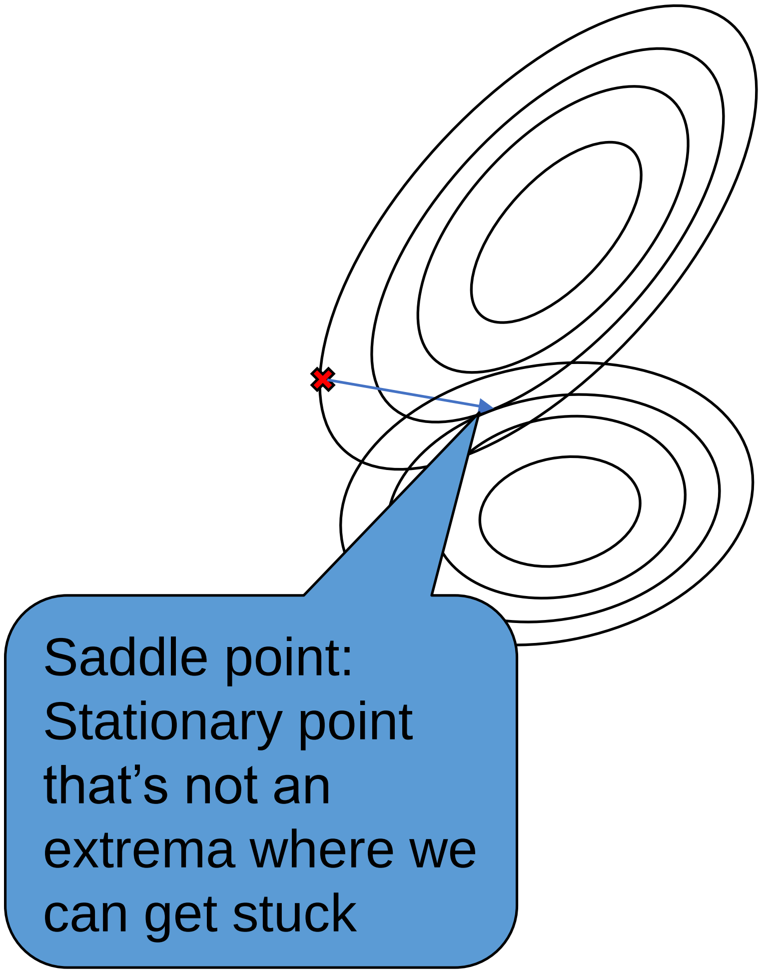




NORTHWESTERN UNIVERSITY

Optimization Visualized

- To find the best weights, we minimize the cost function using optimization
- Optimization starts from an initial point
- Each iteration attempts to make progress towards global minima
- Usually the fastest way is to take a step η in the direction of the steepest descent, called the gradient
- This leads to a path perpendicular to the contours
- However, optimization can have some pitfalls…
  - Local minima
  - Saddle points

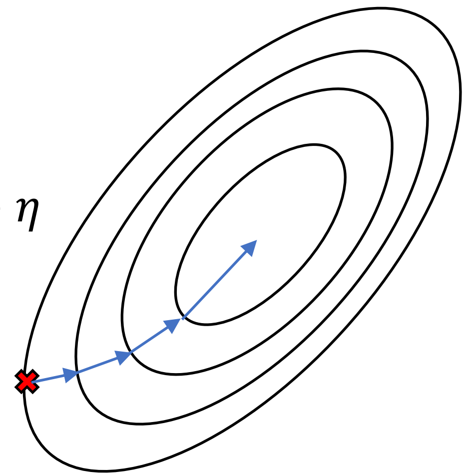


NORTHWESTERN UNIVERSITY

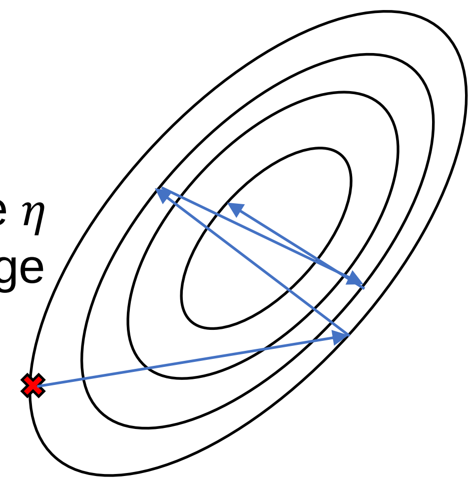
Optimization Visualized

- To find the best weights, we minimize the cost function using optimization
- Optimization starts from an initial point
- Each iteration attempts to make progress towards global minima
- Usually the fastest way is to take a step η in the direction of the steepest descent, called the gradient
- This leads to a path perpendicular to the contours
- However, optimization can have some pitfalls…
  - Local minima
  - Saddle points
  - Bad learning rates η

Step size η is good

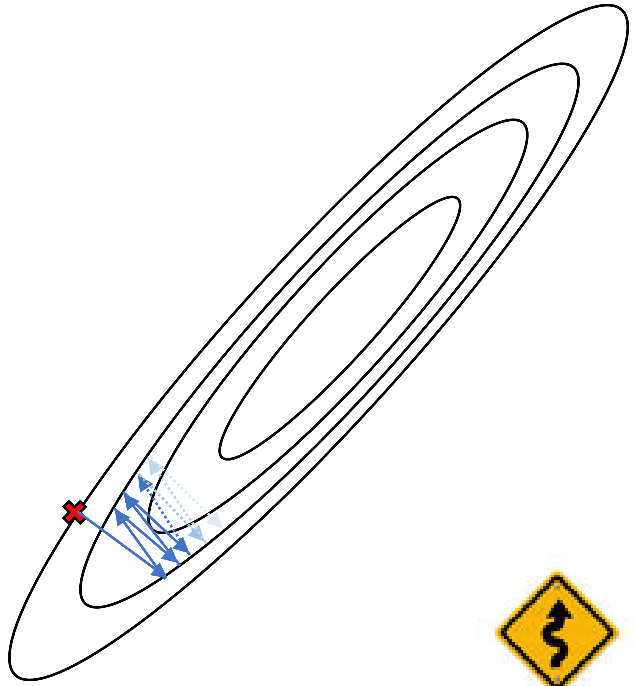


Step size η is too large



Optimization Visualized

- To find the best weights, we minimize the cost function using optimization
- Optimization starts from an initial point
- Each iteration attempts to make progress towards global minima
- Usually the fastest way is to take a step η in the direction of the steepest descent, called the gradient
- This leads to a path perpendicular to the contours
- However, optimization can have some pitfalls…
  - Local minima
  - Saddle points
  - Bad learning rates η

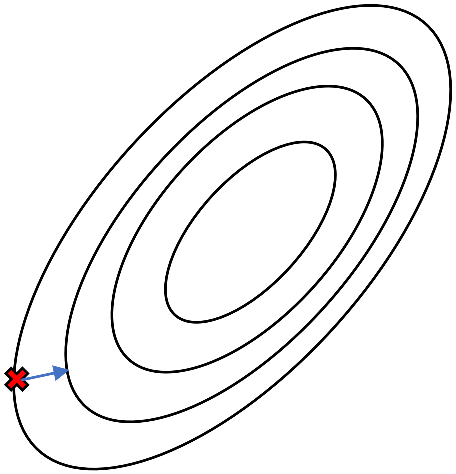


If you forget to normalize your data, weights or activations ;)

NORTHWESTERN UNIVERSITY

Estimating gradients

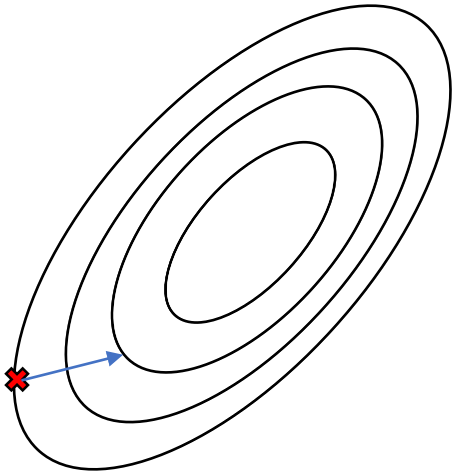
- At each step, the optimizer computes the location of the next point estimate: direction ∇w and step size η



NORTHWESTERN UNIVERSITY

Estimating gradients

- At each step, the optimizer computes the location of the next point estimate: direction ∇w and step size η
- This can be done several ways:
  - Full batch

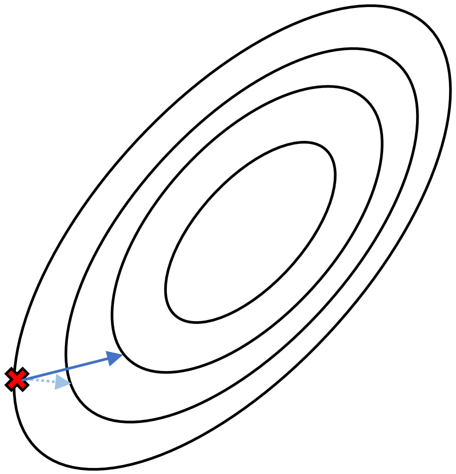


- Full batch means running through all the training data
  - For MNIST, this mean 60,000 samples
  - For some datasets, this could mean terabytes or petabytes of data
  - … if we do that, perhaps we should take a larger step η to justify the cost of computing over all the data
  - Full gradient does the best job in finding the true gradient

NORTHWESTERN UNIVERSITY

Estimating gradients

- At each step, the optimizer computes the location of the next point estimate: direction ∇w and step size η
- This can be done several ways:
  - Full batch
  - Mini-batch



- Mini batch means running through a smaller batch of the training data
  - This is much cheaper than running through all the data. Each batch is called an epoch.
  - However, the gradient estimate is more noisy, so it makes sense to take a smaller step η
  - Depending on the conditioning of the problem, the added cost of small step sizes may outweigh the savings of running a small sample

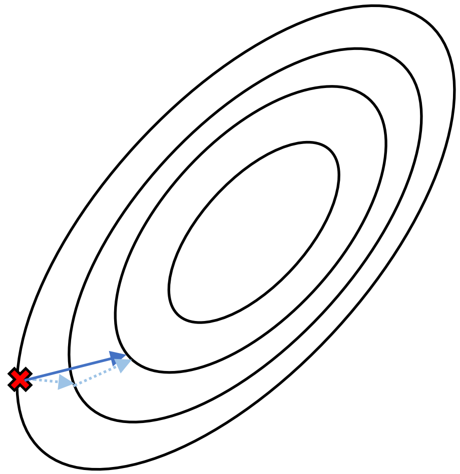
NB: Statisticians often take a smaller sample of the data to build a model. Doing a mini-batch is similar.

NORTHWESTERN UNIVERSITY

Estimating gradients

- At each step, the optimizer computes the location of the next point estimate: direction ∇w and step size η
- This can be done several ways:
  - Full batch
  - Mini-batch

- Mini batch means running through a smaller batch of the training data
  - This is much cheaper than running through all the data. Each batch is called an epoch.
  - However, the gradient estimate is more noisy, so it makes sense to take a smaller step η
  - Depending on the conditioning of the problem, the added cost of small step sizes may outweigh the savings of running a small sample



NB: Statisticians often take a smaller sample of the data to build a model. Doing a mini-batch is similar.

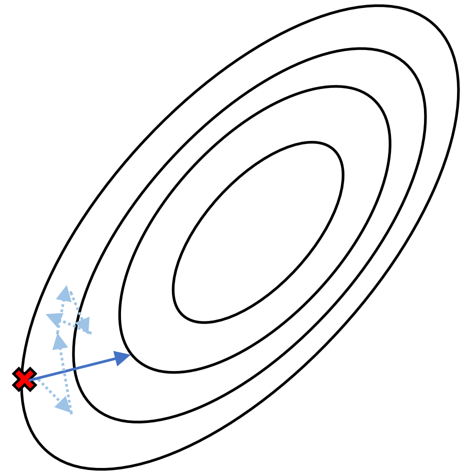
Also, we could take several mini-batches and average over them to get a better estimate

Batch size can make a difference, so choose judiciously

NORTHWESTERN UNIVERSITY

Estimating gradients

- At each step, the optimizer computes the location of the next point estimate: direction ∇w and step size η
- This can be done several ways:
  - Full batch
  - Mini-batch
  - Online

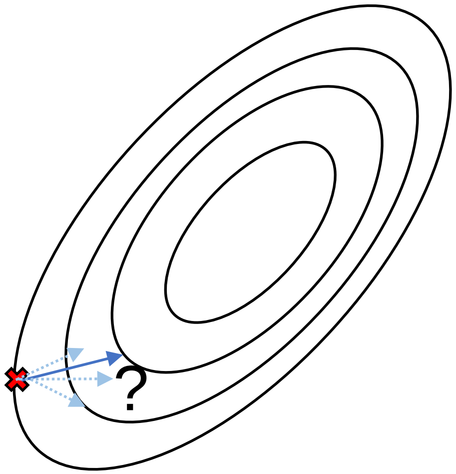


- Online methods update after each training case
  - The gradient can be very noisy due to the lack of averaging
  - Smaller steps are advised
  - This could work well for streaming data sets where it's impractical to store the incoming data

NORTHWESTERN UNIVERSITY

Estimating gradients

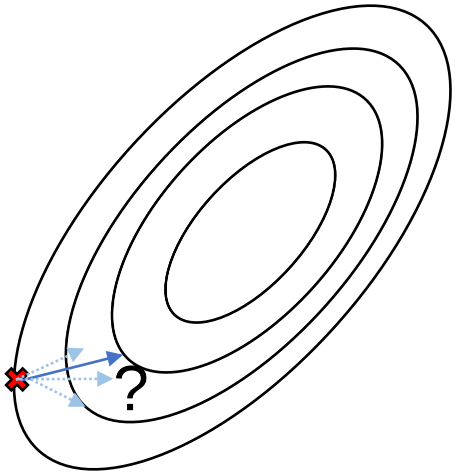
- At each step, the optimizer computes the location of the next point estimate: direction ∇w and step size η
- This can be done several ways:
  - Full batch
  - Mini-batch
  - Online



- How big should the step size η be?

Estimating gradients

- At each step, the optimizer computes the location of the next point estimate: direction ∇w and step size η
- This can be done several ways:
  - Full batch
  - Mini-batch
  - Online

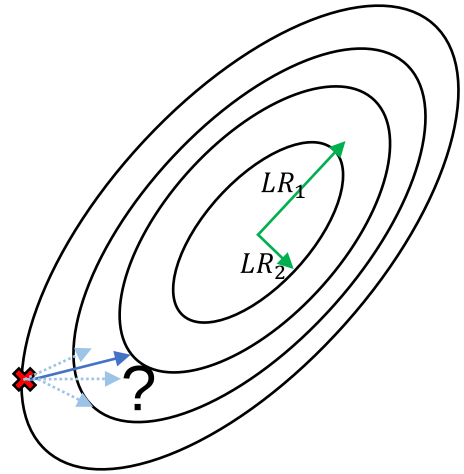


- How big should the step size η be?
  - Fixed global learning rate?
  - Adaptive global learning rate?
  - Learning rate per direction?

NORTHWESTERN UNIVERSITY

Estimating gradients

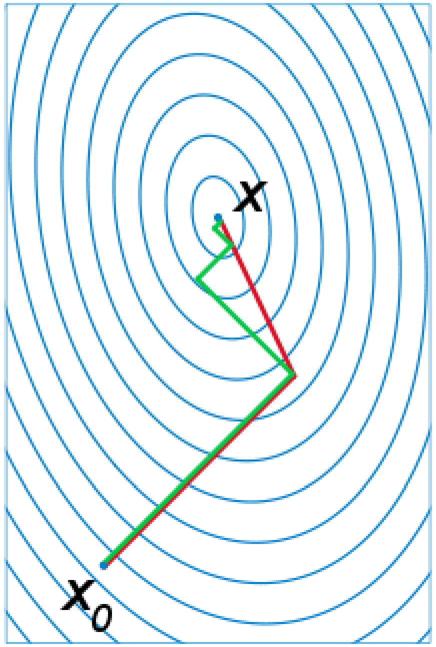
- At each step, the optimizer computes the location of the next point estimate: direction ∇w and step size η
- This can be done several ways:
  - Full batch
  - Mini-batch
  - Online



- How big should the step size η be?
  - Fixed global learning rate?
  - Adaptive global learning rate?
  - Learning rate per direction?

Estimating gradients

- At each step, the optimizer computes the location of the next point estimate: direction ∇w and step size η
- This can be done several ways:
  - Full batch
  - Mini-batch
  - Online
- How big should the step size η be?
  - Fixed global learning rate?
  - Adaptive global learning rate?
  - Learning rate per direction?
  - Using something other than steepest descent?
    - E.g. Conjugate gradient



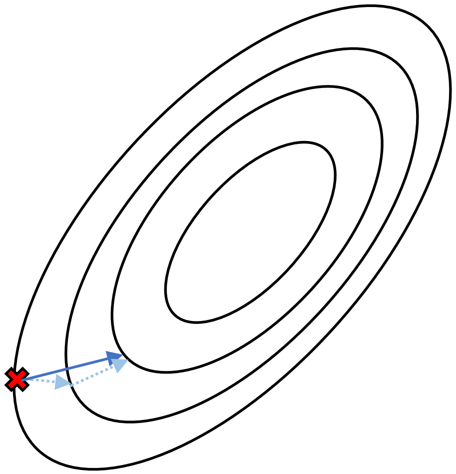
Conjugate gradient [Wikipedia]
Green – gradient descent
Red - Conjugate gradient

Conjugate gradient means choosing a new direction that's conjugate to the previous directions to not "mess up" the work of previous iterations.

NORTHWESTERN UNIVERSITY

Speeding up mini batches

- Each mini-batch epoch estimates gradient ∇w and step size η using a random subset of the data
- Since the gradient is noisy, it's usually necessary to take smaller steps

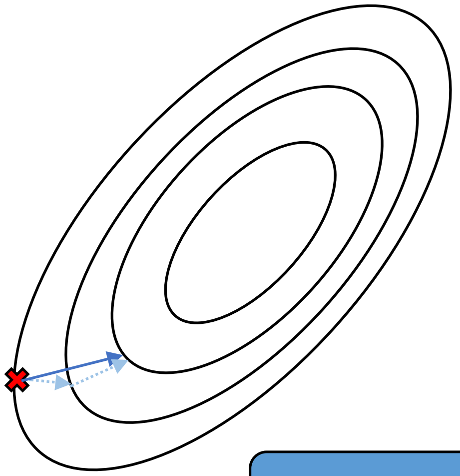


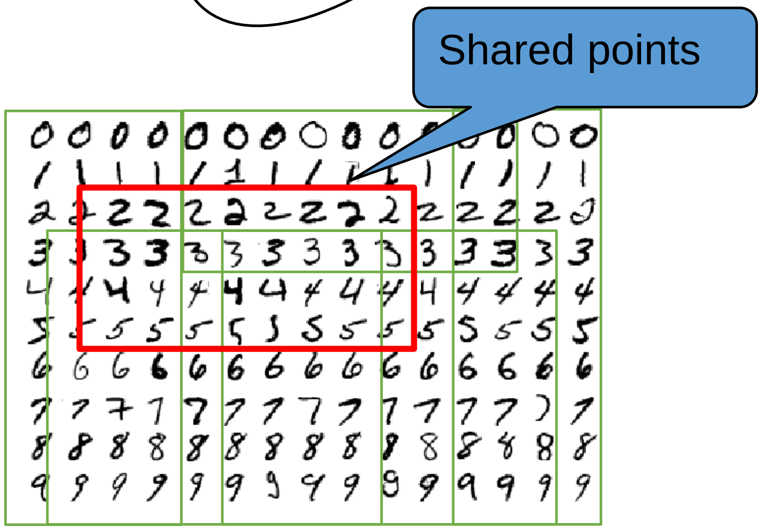
Various ways to slice data for minibatching

NORTHWESTERN UNIVERSITY

Speeding up mini batches

- Each mini-batch epoch estimates gradient ∇w and step size η using a random subset of the data
- Since the gradient is noisy, it's usually necessary to take smaller steps
- However, some of the minibatches share training examples, and hence share some part of their gradient



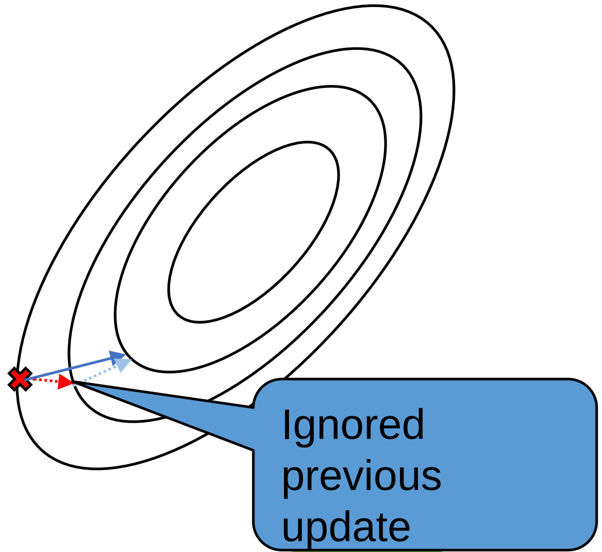


Various ways to slice data for minibatching

NORTHWESTERN UNIVERSITY

Speeding up mini batches

- Each mini-batch epoch estimates gradient ∇w and step size η using a random subset of the data
- Since the gradient is noisy, it's usually necessary to take smaller steps
- However, some of the minibatches share training examples, and hence share some part of their gradient
- Also, information about previous directions of travel are ignored…

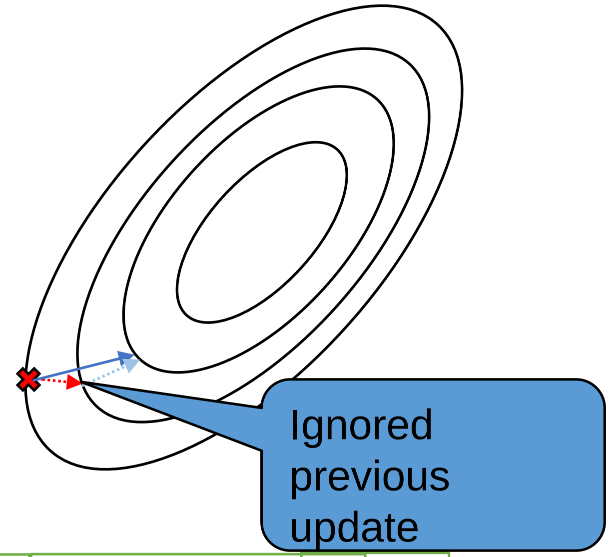


Various ways to slice data for minibatching

NORTHWESTERN UNIVERSITY

Speeding up mini batches

- Each mini-batch epoch estimates gradient ∇w and step size η using a random subset of the data
- Since the gradient is noisy, it's usually necessary to take smaller steps
- However, some of the minibatches share training examples, and hence share some part of their gradient
- Also, information about previous directions of travel are ignored…
- We can exploit these properties to speed up mini-batches by making a better estimate of the gradient to take larger steps with more confidence

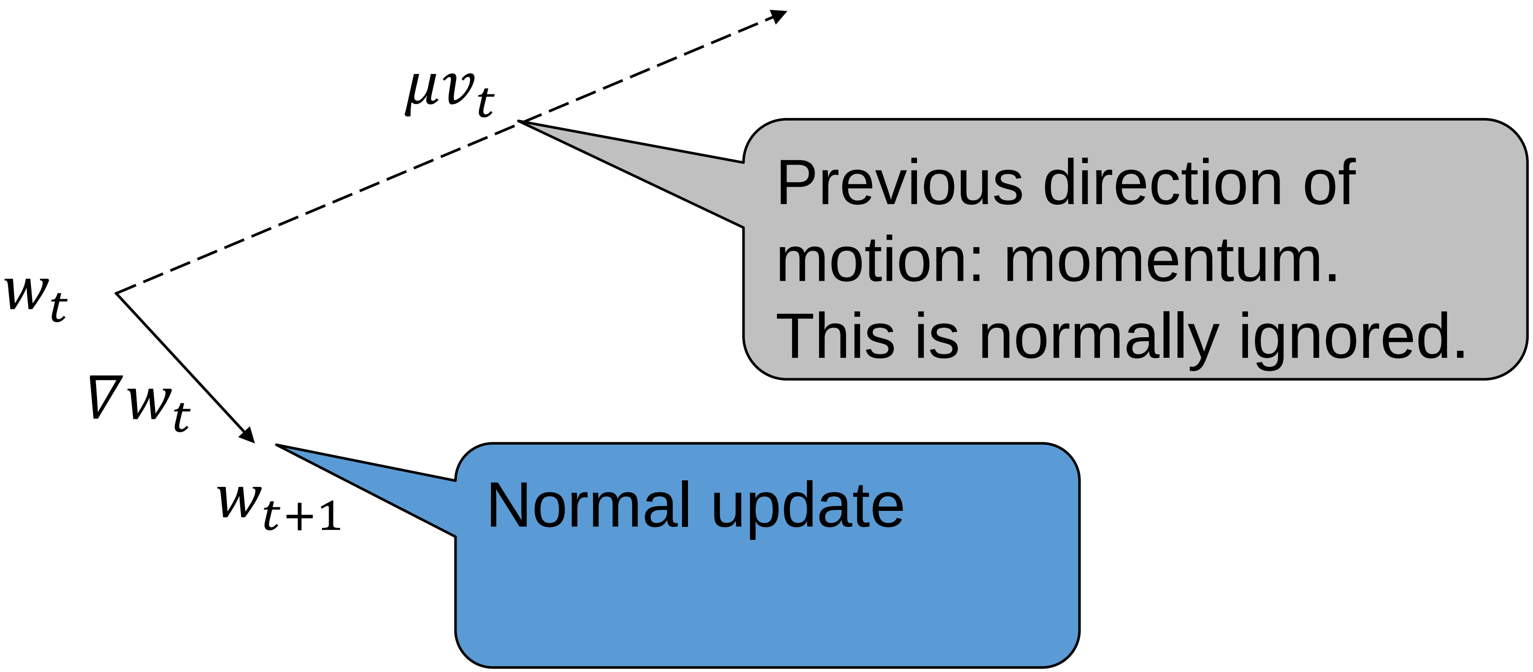


Various ways to slice data for minibatching

Momentum

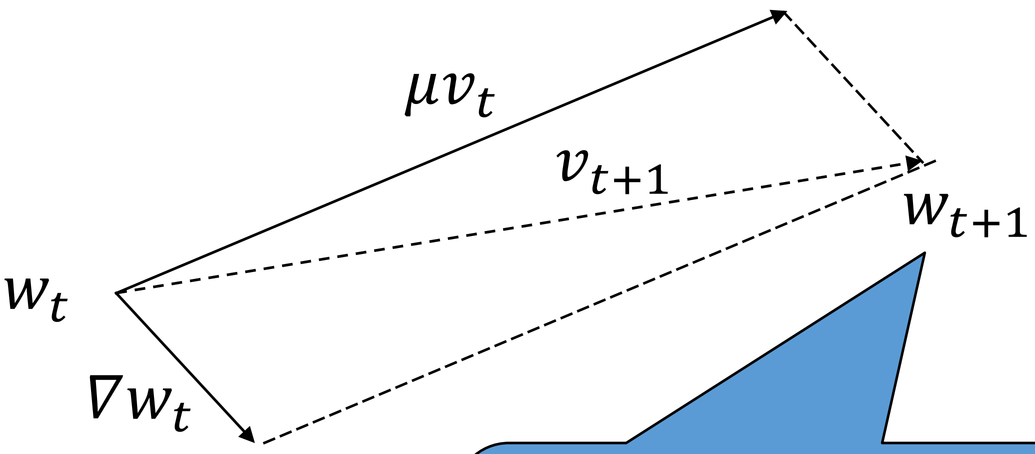
Momentum

$$v_{t+1} = \mu v_t - \eta \nabla w_t$$
$$w_{t+1} = w_t + v_{t+1}$$



Momentum

$$v_{t+1} = \mu v_t - \eta \nabla w_t$$
$$w_{t+1} = w_t + v_{t+1}$$

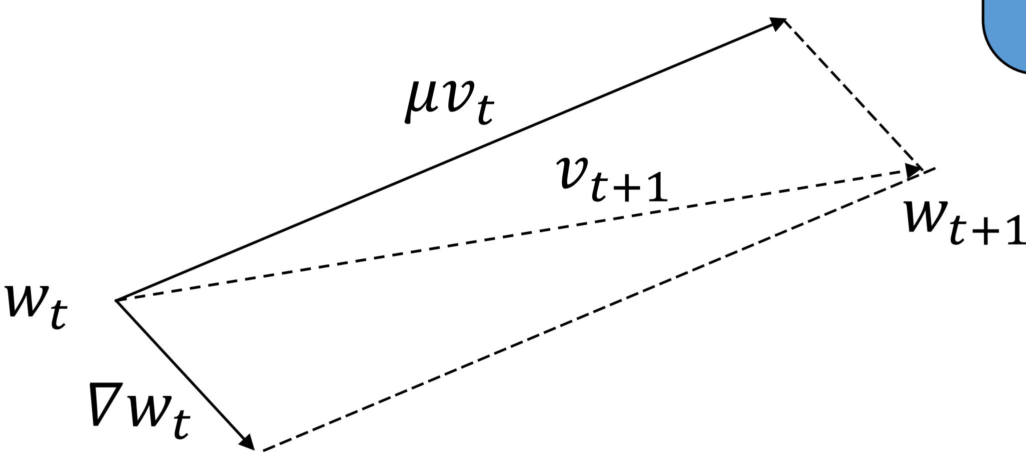


Momentum-adjusted update. Note how momentum is favored over noisy gradient estimate.

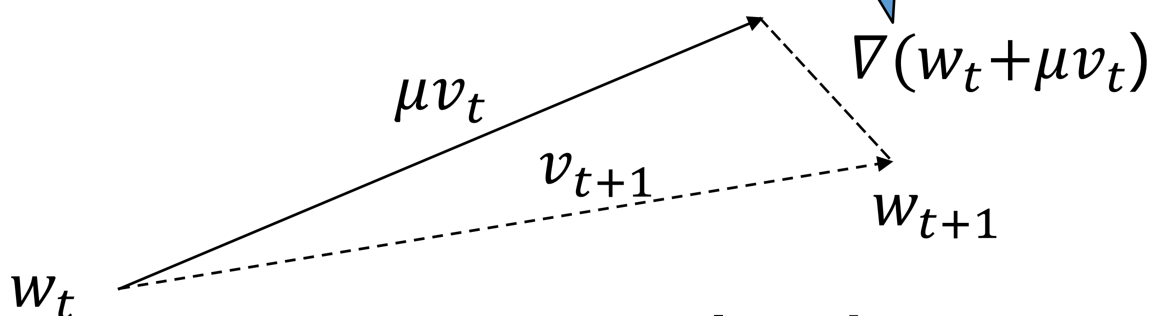
Momentum and acceleration

Momentum

$$v_{t+1} = \mu v_t - \eta \nabla w_t$$
$$w_{t+1} = w_t + v_{t+1}$$



Acceleration (2nd derivative) is now accounted for



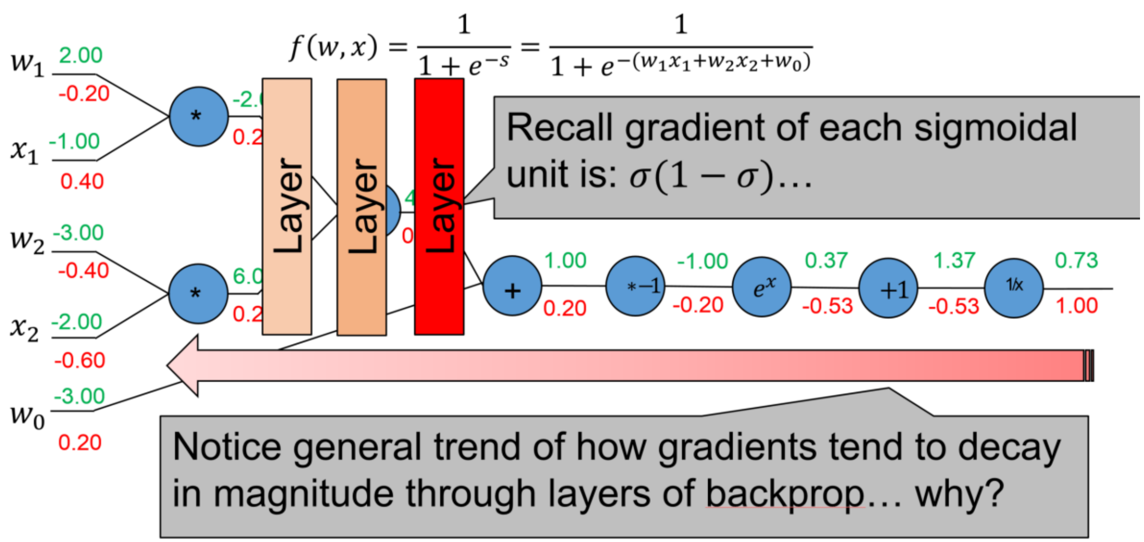
Nesterov Acceleration

Sutskever, ICML 2013

NORTHWESTERN UNIVERSITY

Other ways to speed up mini batches

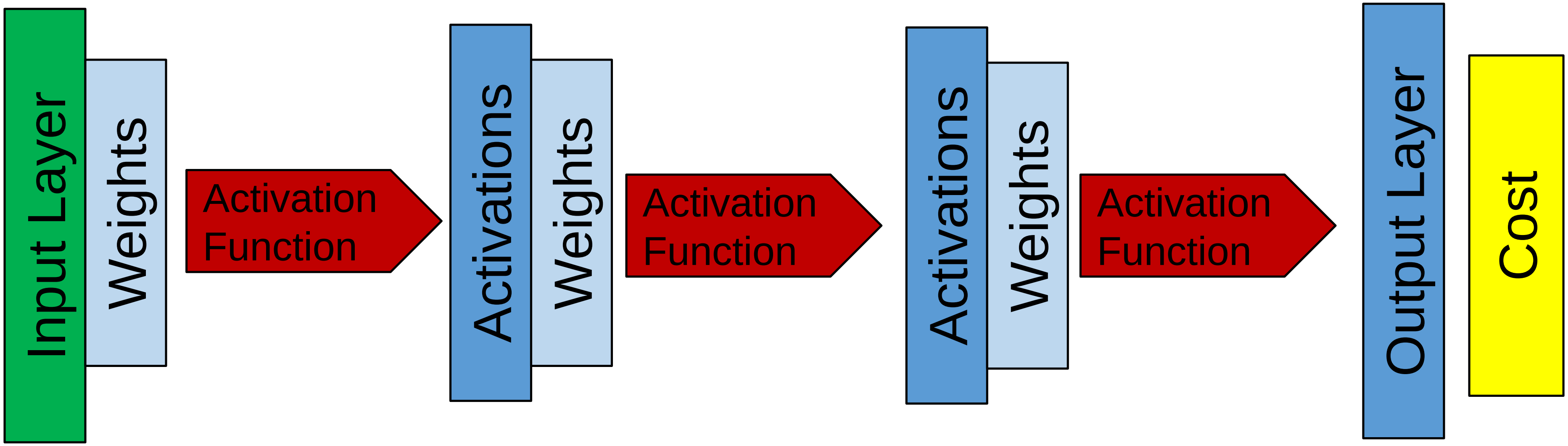
- Adaptive learning rates
  - Remember that earlier layers tend to have smaller gradients



  - Use an adaptive multiplier to boost/attenuate gradients for each layer/weight
  - RMSprop: Keep a moving average of the squared gradient of each weight, divide gradient by this to normalize

NORTHWESTERN UNIVERSITY

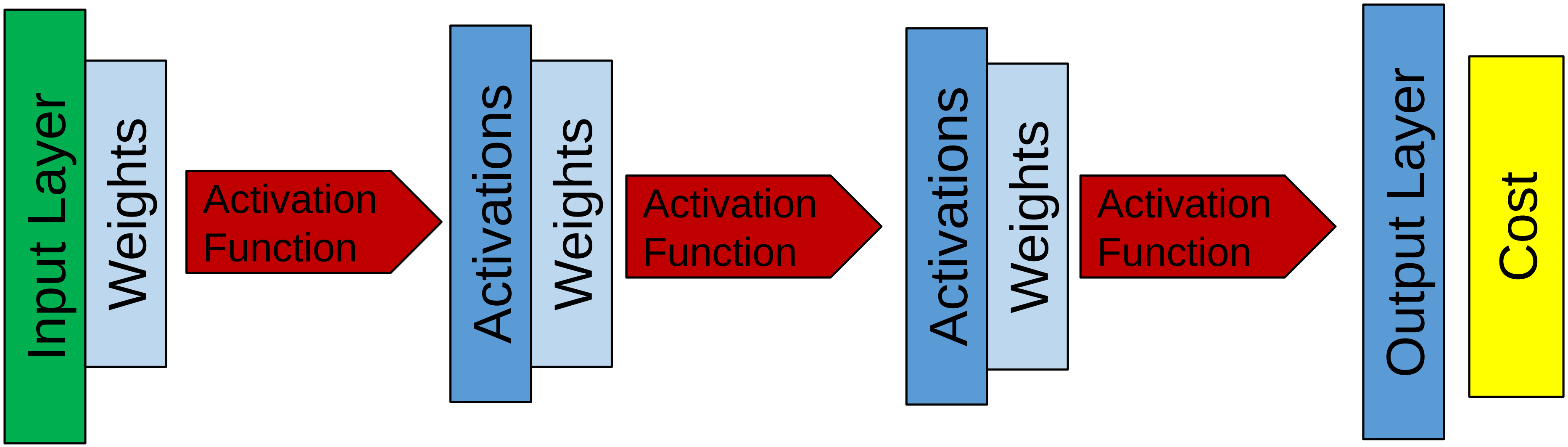
Brute force



- Brute force methods

NORTHWESTERN UNIVERSITY

Brute force



- Brute force methods
- Commercial networks can have billions of parameters, > 30 levels, can take weeks to months to train
- MSR: 152 layers
  - http://www.wired.com/2016/01/microsoft-neural-net-shows-deep-learning-can-get-way-deeper/

NORTHWESTERN UNIVERSITY

GPUs for Deep Learning

NVIDIA Digits
Devbox ~$10k

Four TITAN X GPUs with 7 TFlops of single precision, 336.5 GB/s of memory bandwidth, and 12 GB of memory per board.

VGG net takes ~2-3 weeks to train on 4 GPUs

~$1k per Titan X

Source: NVIDIA

NORTHWESTERN UNIVERSITY

GPUs

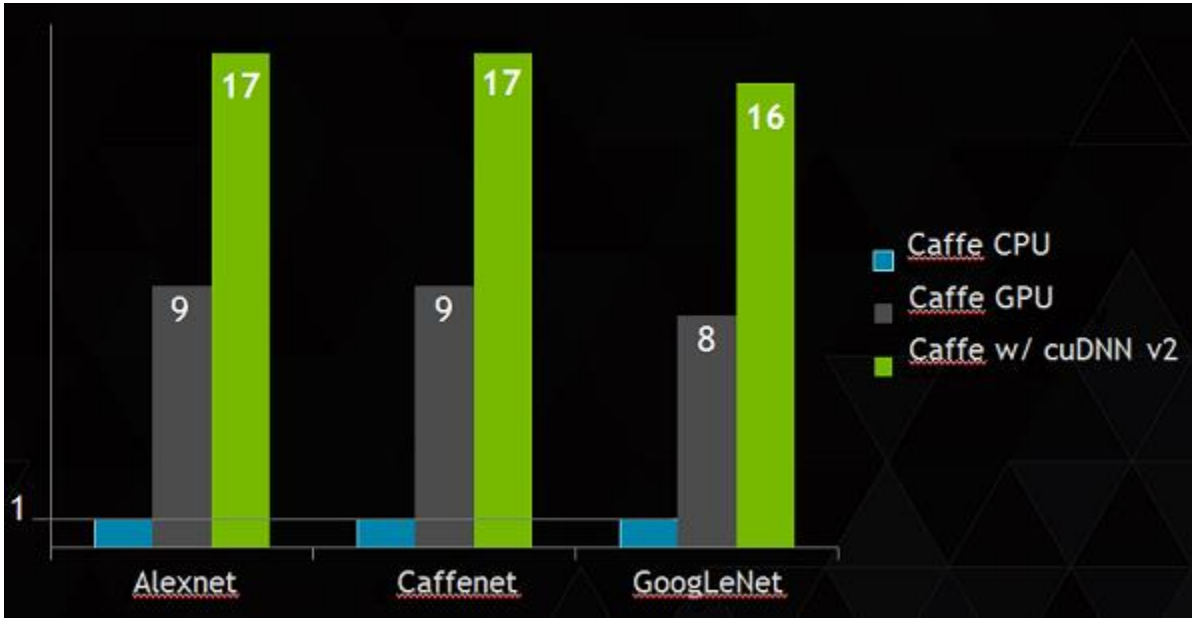
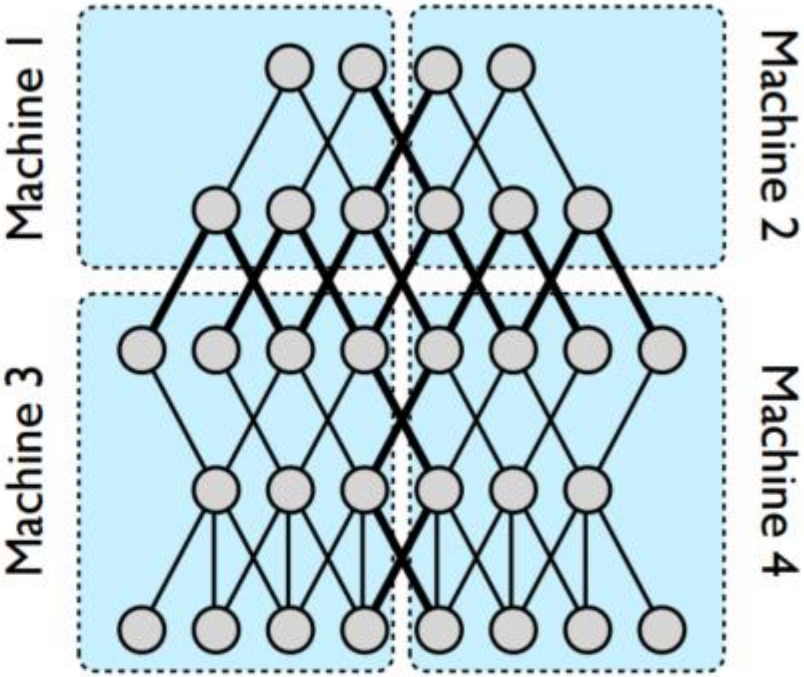
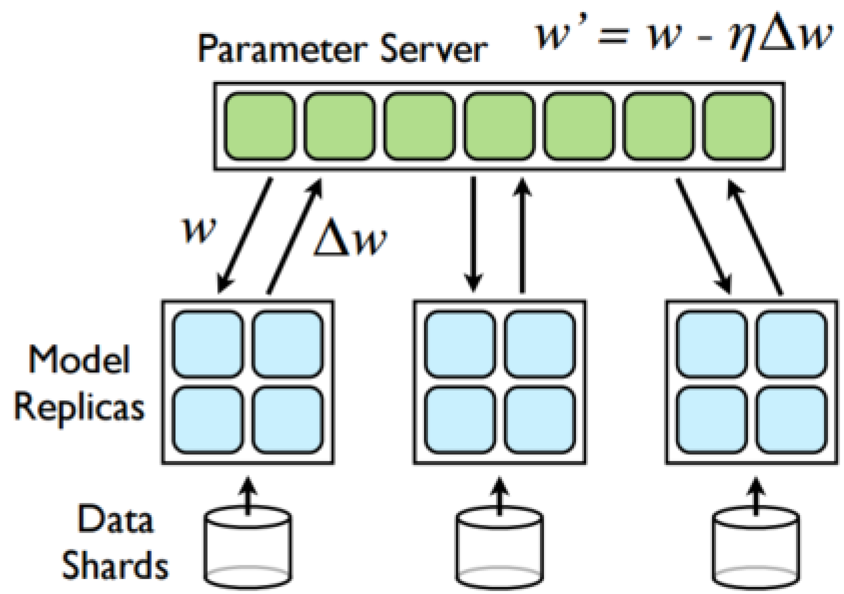


Figure 1: cuDNN performance comparison with CAFFE, using several well known networks. CPU is 16-core Intel Haswell E5-2698 2.3 GHz with 3.6 GHz Turbo. GPU is NVIDIA GeForce GTX TITAN X.

Source: NVIDIA https://devblogs.nvidia.com/parallelforall/cudnn-v2-higher-performance-deep-learning-gpus/

Deep learning parallelized



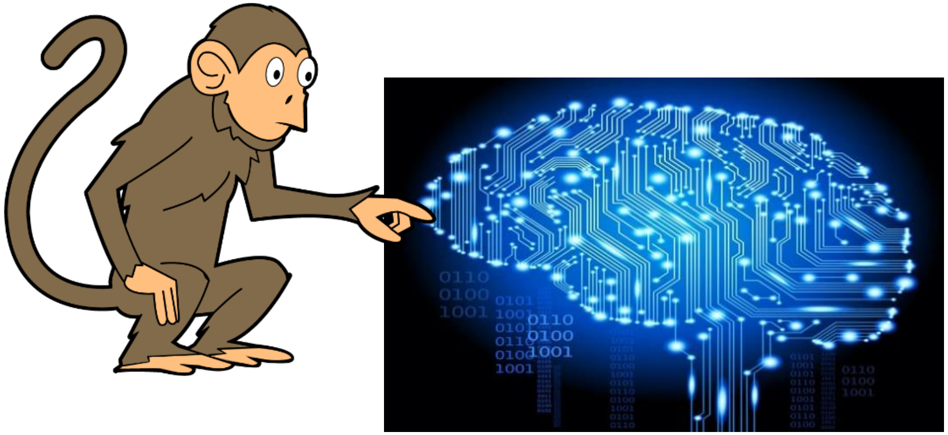


- Divide model across machines
- Sync model periodically
- Out of sync states act as noise → SGD can handle this

Large Scale Distributed Deep Networks. Dean, 2012.

In practice (for this class)

- Training models takes a lot of GPU/CPU and memory
- Luckily, for students like us, there's:
  - A set of pretrained Keras/Tensorflow models we can play with



NORTHWESTERN UNIVERSITY

Summary

- **Data prep:**
  - Remember to clean data by subtracting the mean and normalizing inputs
  - If possible, try to decorrelate inputs

- **Network architecture:**
  - Use ReLU activations instead of sigmoid or tanh
  - Use softmax or cross entropy if modeling mutually exclusive classes: e.g. digits
- **Training:**
  - Try using minibatch training with momentum
  - Be careful about learning rates

Assignment

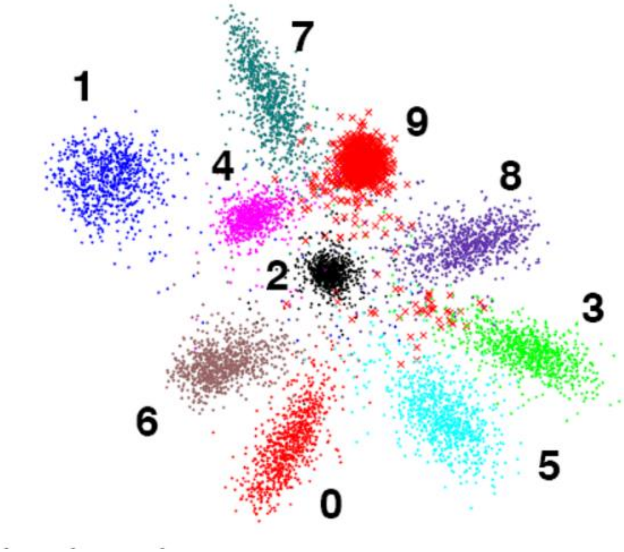
- Assignment
- Quizzes
- Project

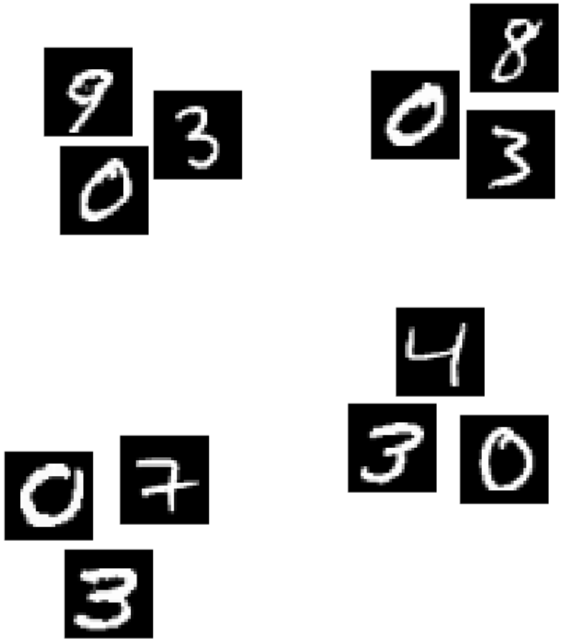
NORTHWESTERN UNIVERSITY

Assignment

- Discuss assignment
- Random labels & NN capacity

Random labels





Non-random labels

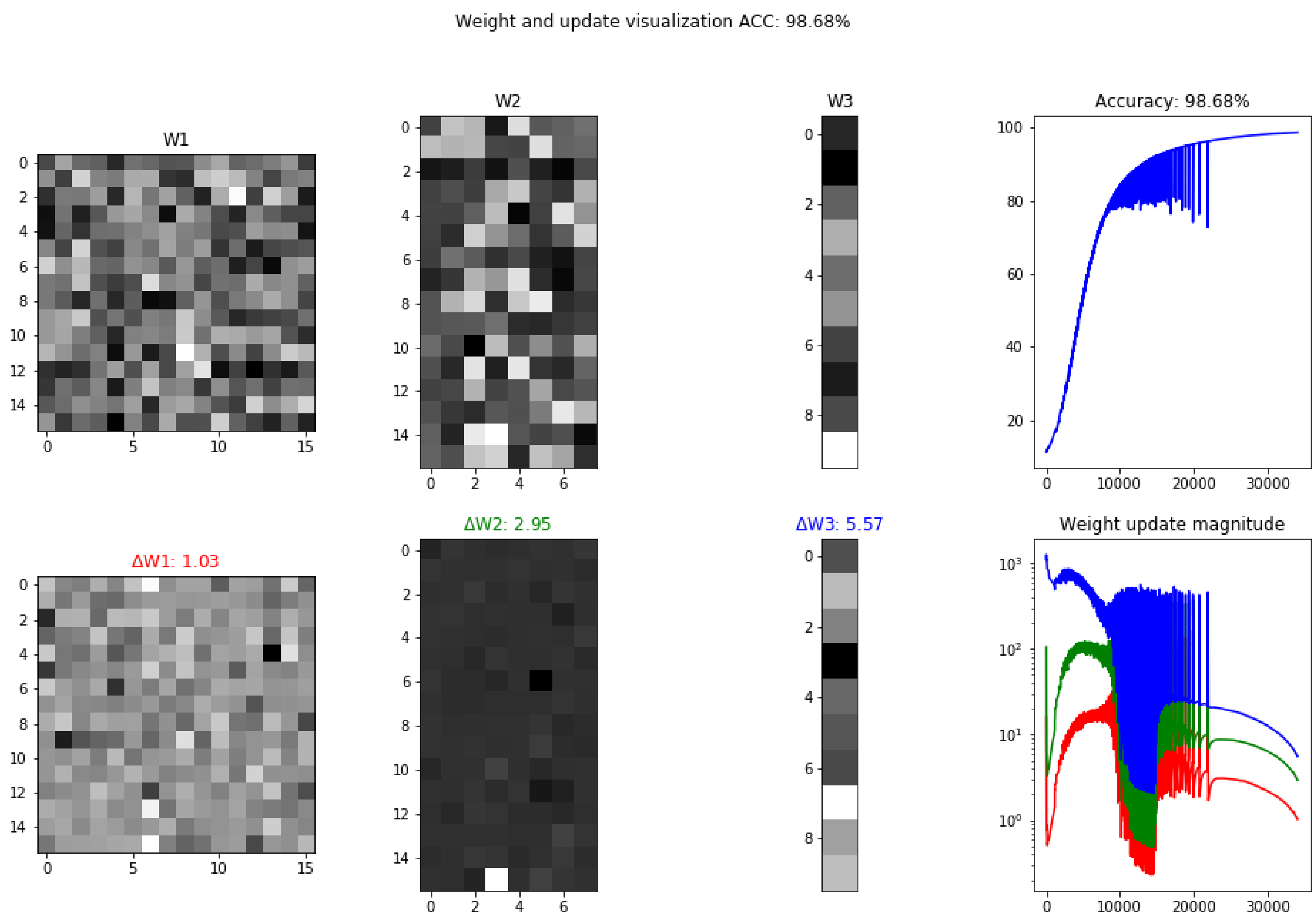
Show some spatial consistency in high dimensional space (clustering)

Random labels

Show no spatial consistency in high dimensional space. Relies on model overfitting.

NORTHWESTERN UNIVERSITY

Random labels



Further reading

- Batch Normalization: http://arxiv.org/pdf/1502.03167.pdf
- 1-bit gradient updates: http://arxiv.org/pdf/1602.02830v1.pdf
- Drop-in for training deep nets: http://arxiv.org/pdf/1511.06951v1.pdf
- Weight normalization
- https://arxiv.org/abs/1602.07868

NORTHWESTERN UNIVERSITY

Optional Reading

- Learning to learn gradient descent by gradient descent: https://arxiv.org/abs/1606.04474
- Synthetic gradient: https://arxiv.org/abs/1608.05343
- Learning to reinforcement learn: https://arxiv.org/abs/1611.05763
- Batch Normalization: https://arxiv.org/pdf/1502.03167.pdf
- Data-dependent init: https://arxiv.org/pdf/1511.06856.pdf

NORTHWESTERN UNIVERSITY

Optional Reading

- Continuum of activations: http://uaf46365.ddns.uark.edu/continuum.pdf
- Designing NN using RL: https://arxiv.org/abs/1611.02167
- Neural arch search using RL: https://openreview.net/forum?id=r1Ue8Hcxg
- Sacred metaoptimizer: https://github.com/IDSIA/sacred
- What is softmax? http://www.kdnuggets.com/2016/07/softmax-regression-related-logistic-regression.html
- Overcoming forgetting: https://arxiv.org/abs/1612.00796
- Local minima: https://arxiv.org/abs/1611.06310

NORTHWESTERN UNIVERSITY

Optional Reading

- Hyperopt:
  - https://github.com/hyperopt/hyperopt-sklearn
  - http://optunity.readthedocs.io/en/latest/
- Tanh paper: [2010] http://jmlr.org/proceedings/papers/v9/glorot10a/glorot10a.pdf - ReLU invented in 2011 (Wikipedia)
- Overview of gradient descent algs: http://sebastianruder.com/optimizing-gradient-descent/

NORTHWESTERN UNIVERSITY

Optional Reading

- More on training and tanh: http://jmlr.org/proceedings/papers/v9/glorot10a/glorot10a.pdf
- More on ReLU: http://www.jmlr.org/proceedings/papers/v15/glorot11a/glorot11a.pdf