Backpropagation in Neural Nets

Materials from

- Intel Deep Learning https://www.intel.com/content/www/us/en/developer/learn/course-deep-learning.html

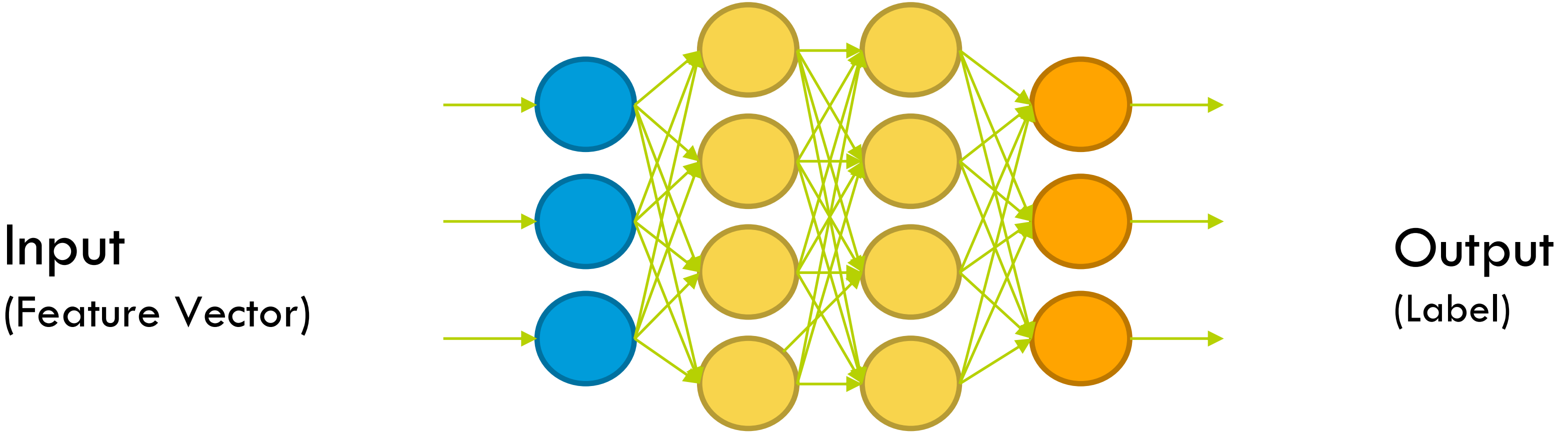
Legal Notices and Disclaimers

This presentation is for informational purposes only. INTEL MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS SUMMARY.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. Check with your system manufacturer or retailer or learn more at intel.com.

This sample source code is released under the Intel Sample Source Code License Agreement.

How to Train a Neural Net?



- Put in Training inputs, get the output
- Compare output to correct answers: Look at loss function J
- Adjust and repeat!
- Backpropagation tells us how to make a single adjustment using calculus.

How have we trained before?

LOSS function : focus only 1 iteration
Loss Cost : focus on the whole

There’s more of the detail in initializing the models

- Gradient Descent!

1. Make prediction
2. Calculate Loss
3. Calculate gradient of the loss function w.r.t. parameters
4. Update parameters by taking a step in the opposite direction
5. Iterate

How have we trained before?

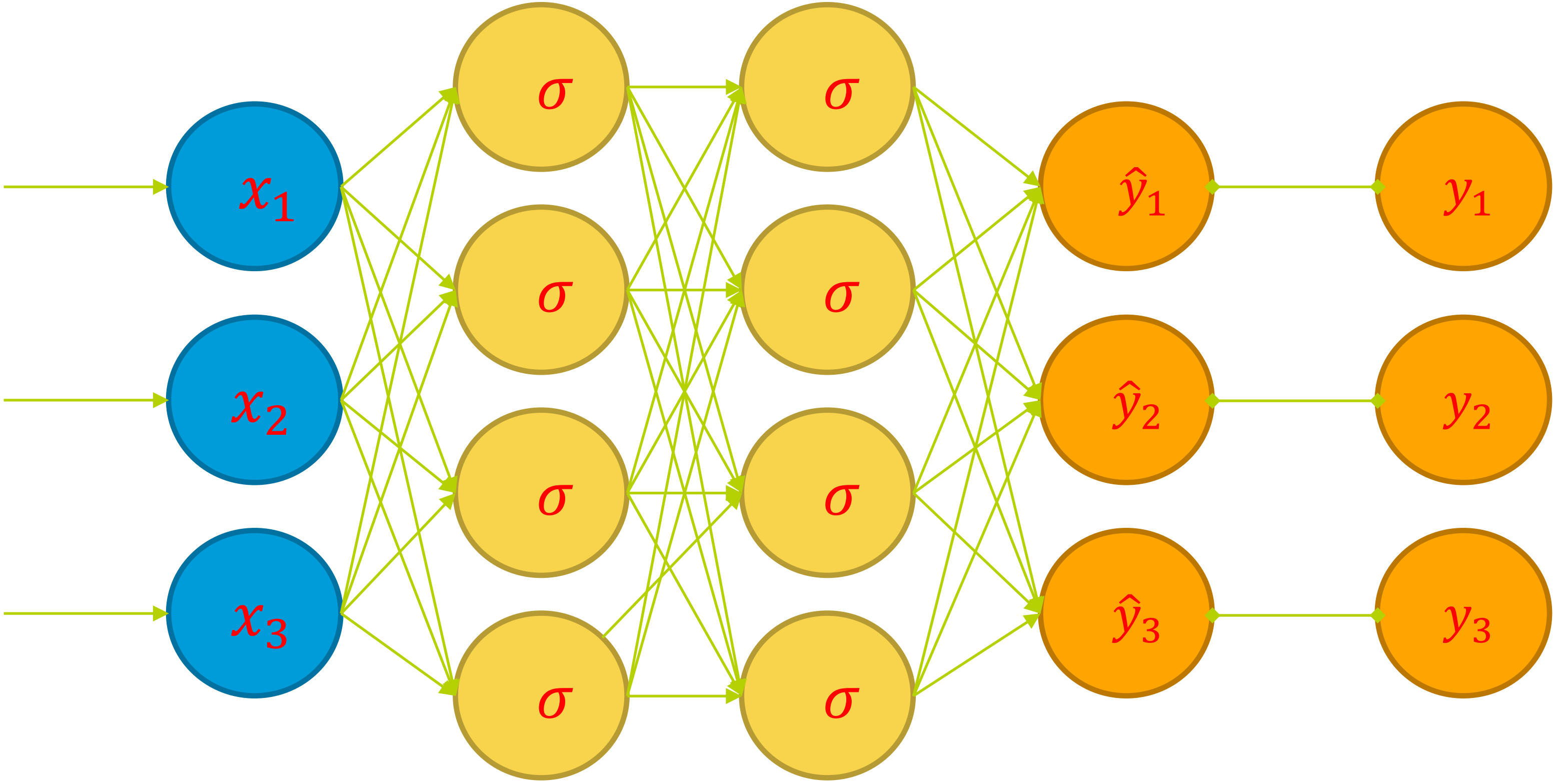
Hyper parameter : the upper layer of parameter for improving the lower layer parameter

parameter for training model
- Loss function
- Learning rate
- epoch
- batch_size
- optimizer

- Gradient Descent!

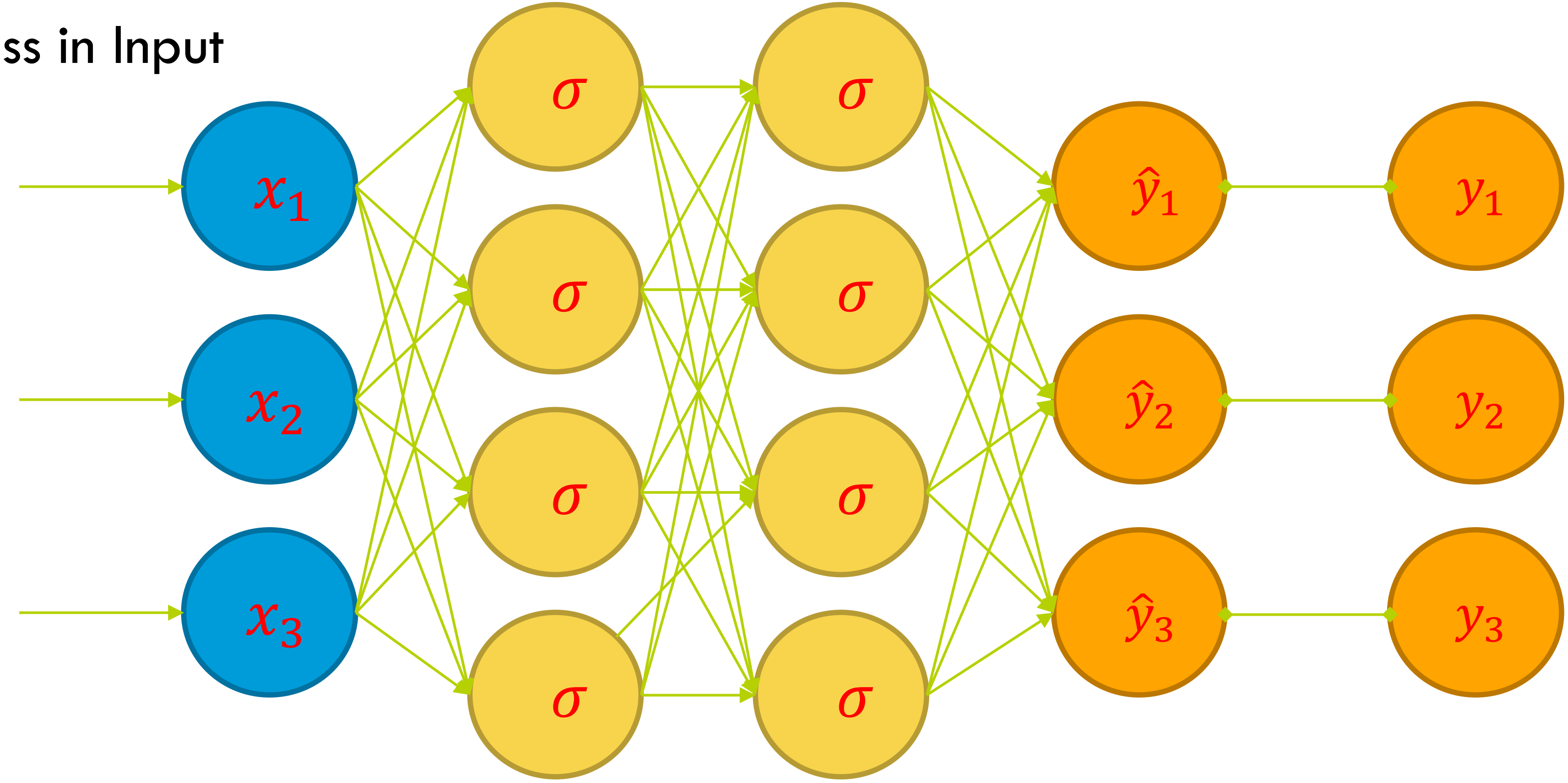
1. Make prediction
2. Calculate Loss
3. Calculate gradient of the loss function w.r.t. parameters
4. Update parameters by taking a step in the opposite direction
5. Iterate

Feedforward Neural Network



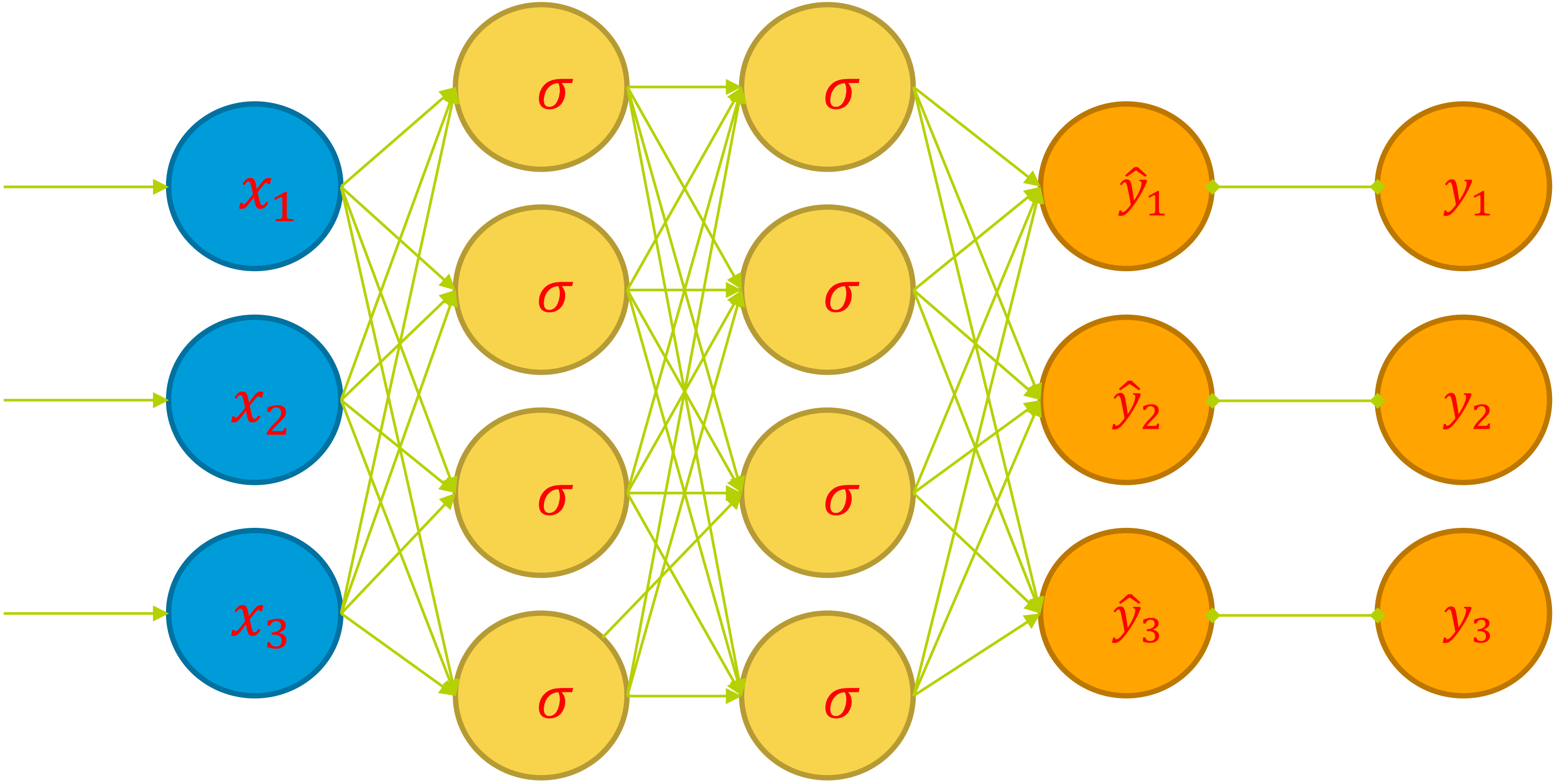
Forward Propagation

Pass in Input

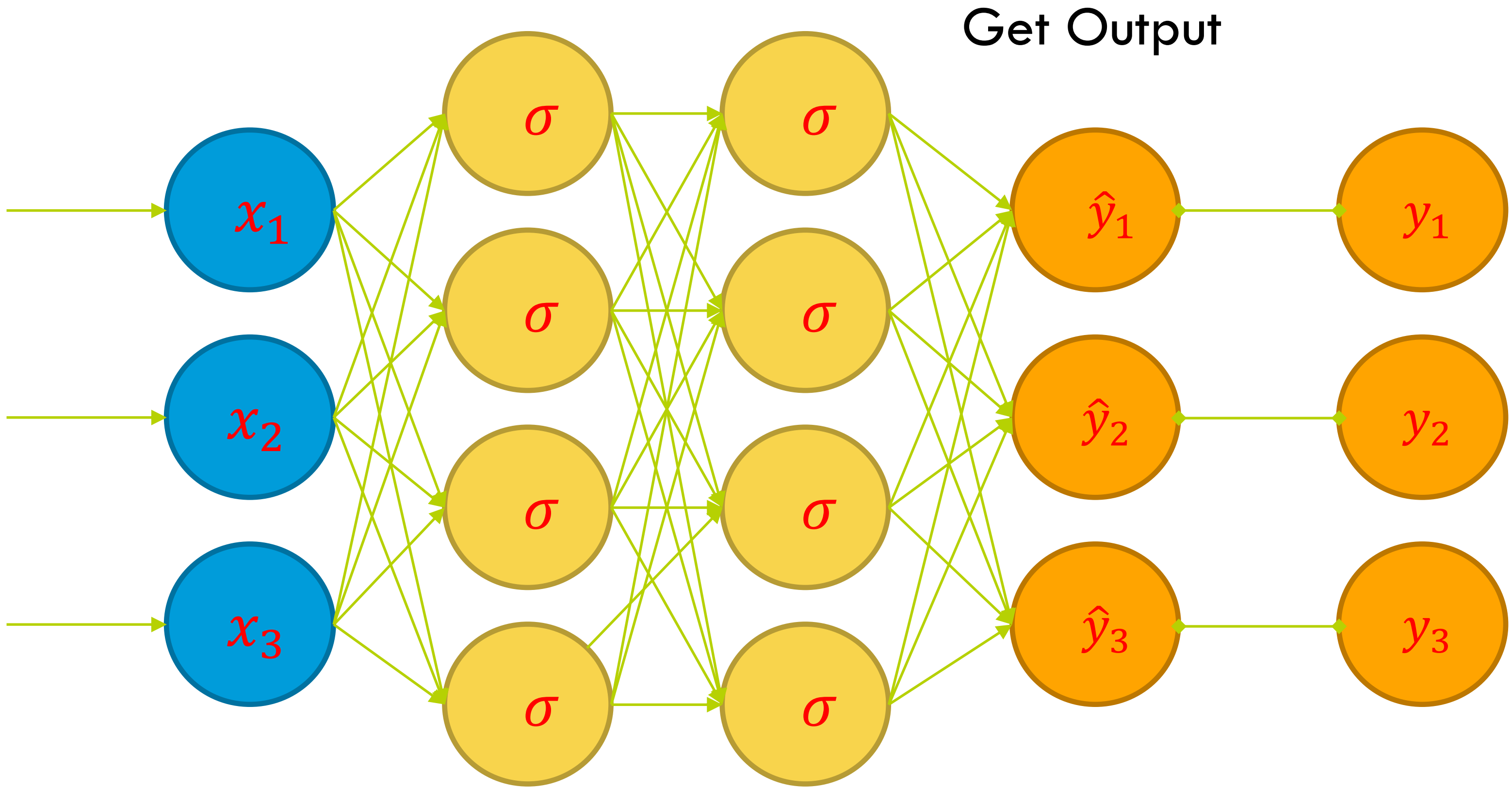


Forward Propagation

Calculate each Layer



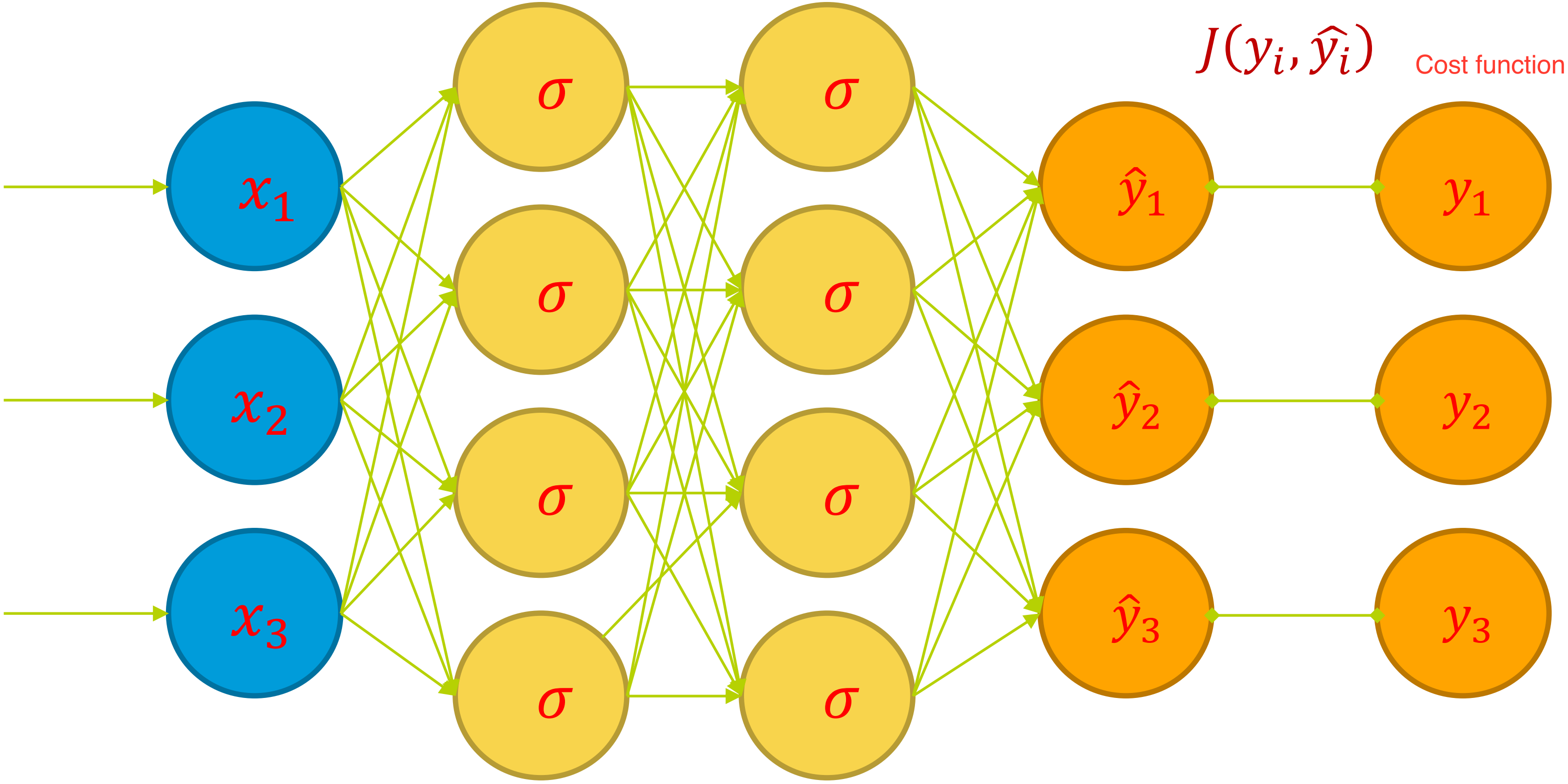
Forward Propagation



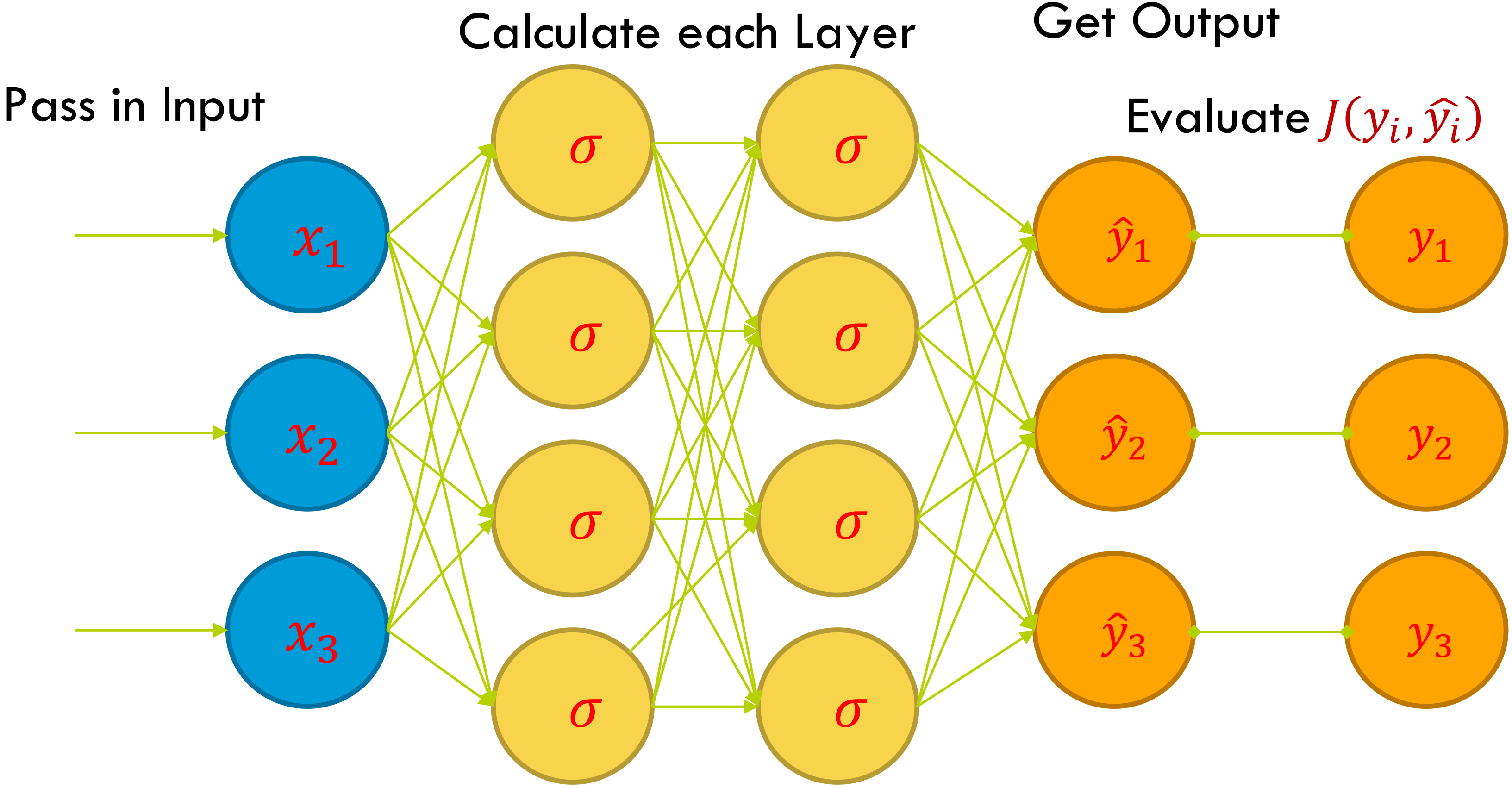
Loss Calculation

Evaluate:

$J(y_i, \hat{y_i})$ Cost function



Loss Calculation



How have we trained before?

- Gradient Descent!

1. Make prediction
2. Calculate Loss
3. Calculate gradient of the loss function w.r.t. parameters
4. Update parameters by taking a step in the opposite direction
5. Iterate

How to calculate gradient?

- Chain rule

How to Train a Neural Net?

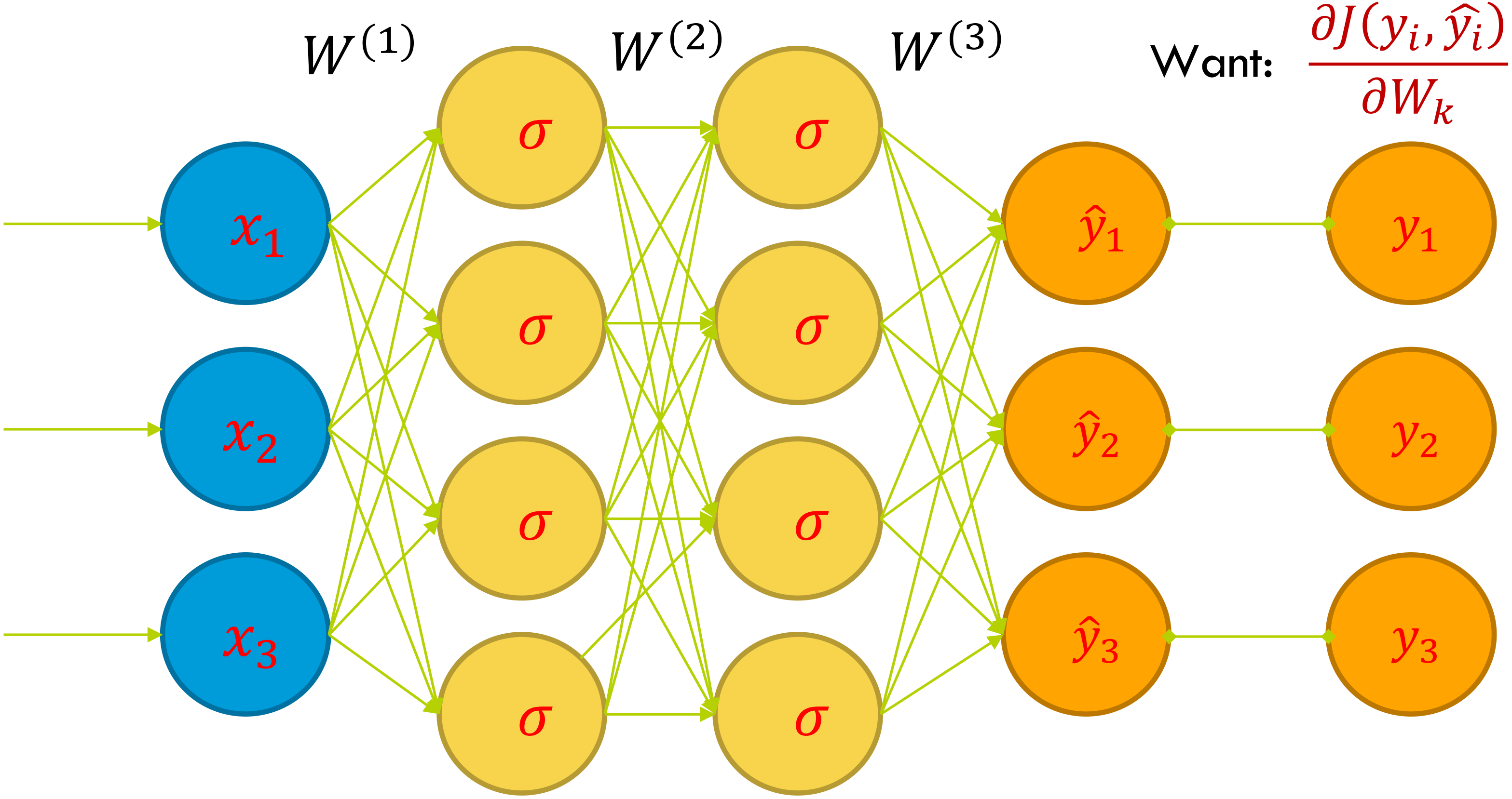
- How could we change the weights to make our Loss Function lower?
- Think of neural net as a function $F: X \longrightarrow Y$ func that input x and output y
- F is a complex computation involving many weights W_k
- Given the structure, the weights "define" the function F (and therefore define our model)
- Loss Function is $J(y, F(x))$

How to Train a Neural Net?

find the slope to fix the weight to the value that make the loss function of that value going down

- Get $\frac{\partial J}{\partial W_k}$ for every weight in the network.
- This tells us what direction to adjust each W_k if we want to lower our loss function.
- Make an adjustment and repeat!

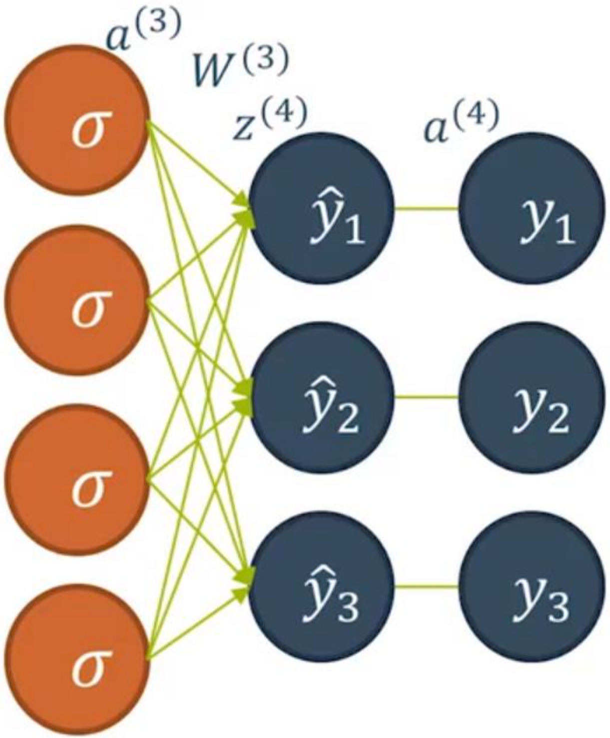
Feedforward Neural Network



Calculus to the Rescue

- Use calculus, chain rule, etc. etc.
- Functions are chosen to have “nice” derivatives
- Numerical issues to be considered

How to calculate gradient



Calculate $\frac{\partial J}{\partial W^{(3)}}$

Where:

$$\frac{\partial J}{\partial W^{(3)}} = \frac{\partial J}{\partial a^{(4)}} \frac{\partial a^{(4)}}{\partial z^{(4)}} \frac{\partial z^{(4)}}{\partial W^{(3)}}$$

How to calculate gradient

Need to Calculate three pieces for $\frac{\partial J}{\partial W^{(3)}}$

y hat

1. $J = (1/2)\,(a^{(4)} - y)^2$

$$\Rightarrow \frac{\partial J}{\partial a^{(4)}} = 2 * (1/2)(a^{(4)} - y) * 1$$

2. $a^{(4)} = z^{(4)} \quad \Rightarrow. \quad \frac{\partial a^{(4)}}{\partial z^{(4)}} = 1$ simply activation

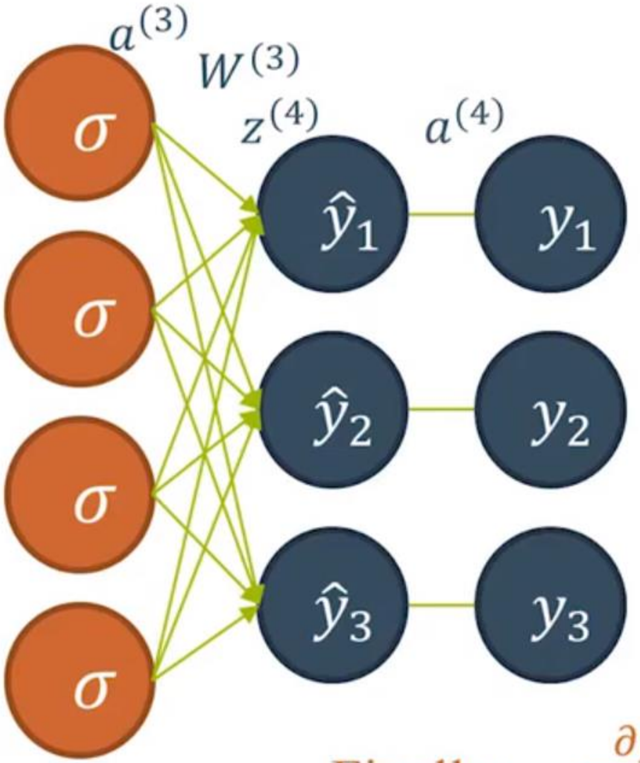
3. $z^{(4)} = a^{(3)} W^{(3)} \quad \Rightarrow \quad \frac{\partial z^{(4)}}{\partial W^{(3)}} = a^{(3)}$

Calculate $\frac{\partial J}{\partial W^{(3)}}$

Where:

$$\frac{\partial J}{\partial W^{(3)}} = \frac{\partial J}{\partial a^{(4)}} \frac{\partial a^{(4)}}{\partial z^{(4)}} \frac{\partial z^{(4)}}{\partial W^{(3)}}$$

How to calculate gradient



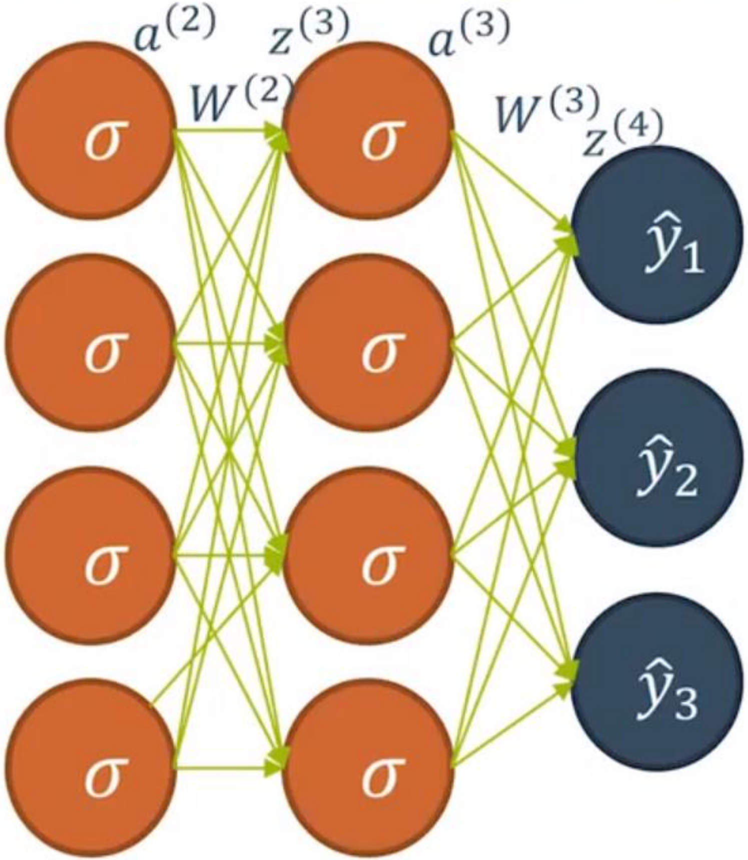
Calculate $\frac{\partial J}{\partial W^{(3)}}$

Where:

$$\frac{\partial J}{\partial W^{(3)}} = \frac{\partial J}{\partial a^{(4)}} \frac{\partial a^{(4)}}{\partial z^{(4)}} \frac{\partial z^{(4)}}{\partial W^{(3)}}$$

Finally: $\frac{\partial J}{\partial W^{(3)}} = (a^{(4)} - y) * a^{(3)}$

How to calculate gradient



Calculate $\frac{\partial J}{\partial W^{(2)}}$

Where:

$$\frac{\partial J}{\partial W^{(2)}} = \frac{\partial J}{\partial a^{(3)}} \frac{\partial a^{(3)}}{\partial z^{(3)}} \frac{\partial z^{(3)}}{\partial W^{(2)}}$$

and

$$\frac{\partial J}{\partial a^{(3)}} = \frac{\partial J}{\partial a^{(4)}} \frac{\partial a^{(4)}}{\partial z^{(4)}} \frac{\partial z^{(4)}}{\partial a^{(3)}}$$

How to calculate gradient

Need to calculate three new pieces for $\frac{\partial J}{\partial W^{(2)}}$

1. $z^{(3)} = a^{(2)}W^{(2)} \quad \Longrightarrow \quad \frac{\partial z^{(3)}}{\partial W^{(2)}} = a^{(2)}$

2. $a^{(3)} = \frac{1}{1+e^{-z^{(3)}}}$

$$\Longrightarrow \frac{\partial a^{(3)}}{\partial z^{(3)}} = \sigma'(z^{(3)}) = \sigma(z^{(3)})(1-\sigma(z^{(3)}))$$

3. $z^{(4)} = a^{(3)}W^{(3)} \quad \Longrightarrow. \quad \frac{\partial z^{(4)}}{\partial a^{(3)}} = W^{(3)}$

Calculate $\frac{\partial J}{\partial W^{(2)}}$

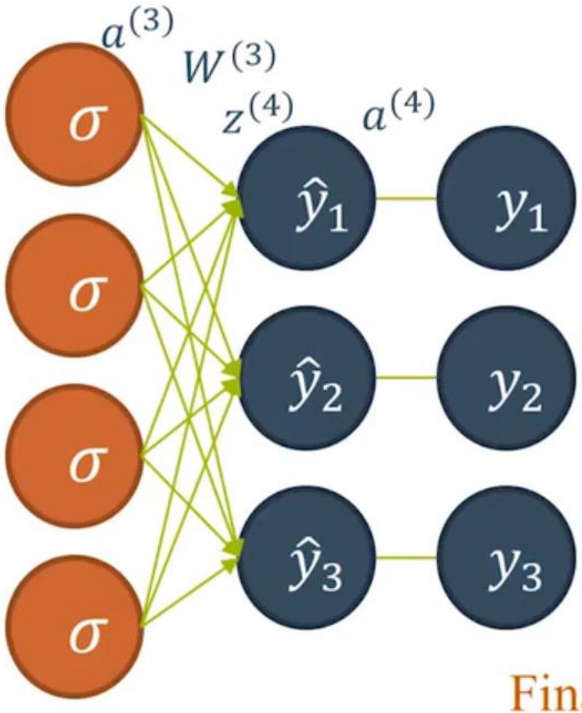
Where:

$$\frac{\partial J}{\partial W^{(2)}} = \frac{\partial J}{\partial a^{(3)}} \frac{\partial a^{(3)}}{\partial z^{(3)}} \frac{\partial z^{(3)}}{\partial W^{(2)}}$$

and

$$\frac{\partial J}{\partial a^{(3)}} = \frac{\partial J}{\partial a^{(4)}} \frac{\partial a^{(4)}}{\partial z^{(4)}} \frac{\partial z^{(4)}}{\partial a^{(3)}}$$

How to calculate gradient



Calculate $\frac{\partial J}{\partial W^{(2)}}$

Where:

$$\frac{\partial J}{\partial W^{(2)}} = \frac{\partial J}{\partial a^{(3)}} \frac{\partial a^{(3)}}{\partial z^{(3)}} \frac{\partial z^{(3)}}{\partial W^{(2)}}$$

and

$$\frac{\partial J}{\partial a^{(3)}} = \frac{\partial J}{\partial a^{(4)}} \frac{\partial a^{(4)}}{\partial z^{(4)}} \frac{\partial z^{(4)}}{\partial a^{(3)}}$$

Finally:

$$\frac{\partial J}{\partial W^{(2)}} = (\hat{y} - y) \cdot W^{(3)} \cdot \sigma\left(z^{(3)}\right)\left(1 - \sigma\left(z^{(3)}\right)\right) \cdot a^{(2)}$$

Punchline

fior improving each weight layer

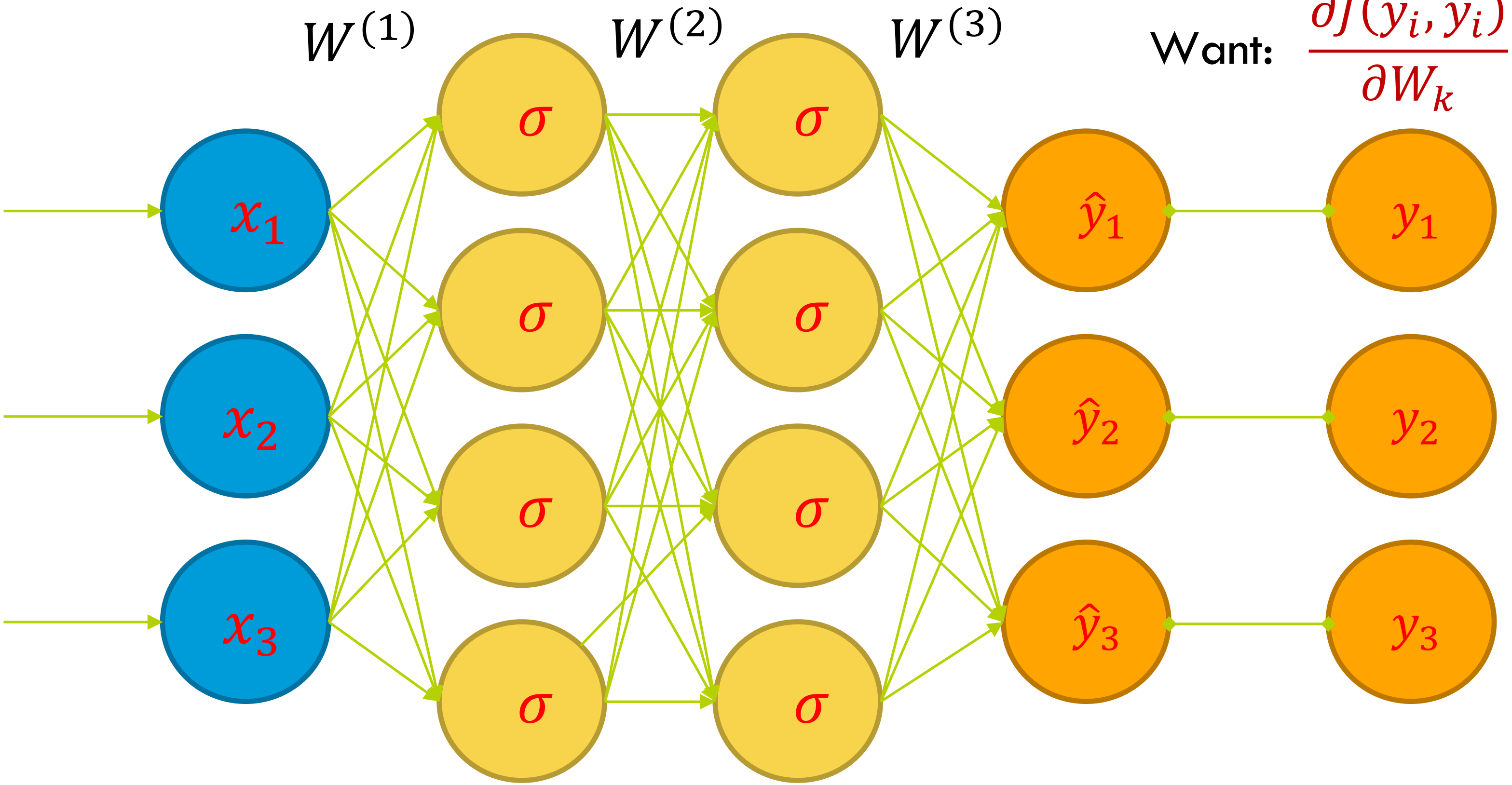
$$\frac{\partial J}{\partial W^{(3)}} = (\hat{y} - y) \cdot a^{(3)}$$

$$\frac{\partial J}{\partial W^{(2)}} = (\hat{y} - y) \cdot W^{(3)} \cdot \sigma'\left(z^{(3)}\right) \cdot a^{(2)}$$

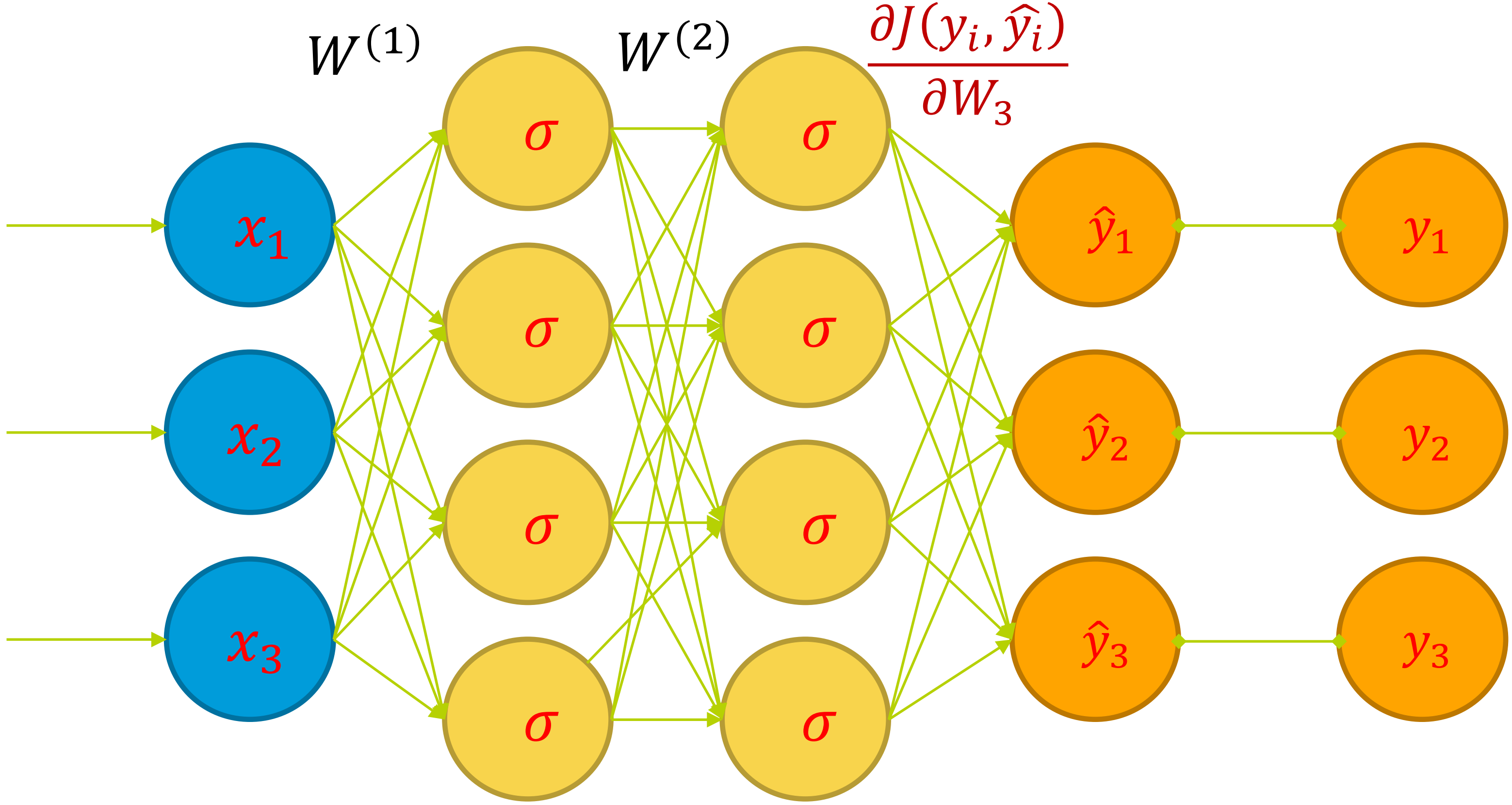
$$\frac{\partial J}{\partial W^{(1)}} = (\hat{y} - y) \cdot W^{(3)} \cdot \sigma'\left(z^{(3)}\right) \cdot W^{(2)} \cdot \sigma'\left(z^{(2)}\right) \cdot X$$

- Recall that: $\sigma'(z) = \sigma(z)(1 - \sigma(z))$
- Though they appear complex, above are easy to compute!

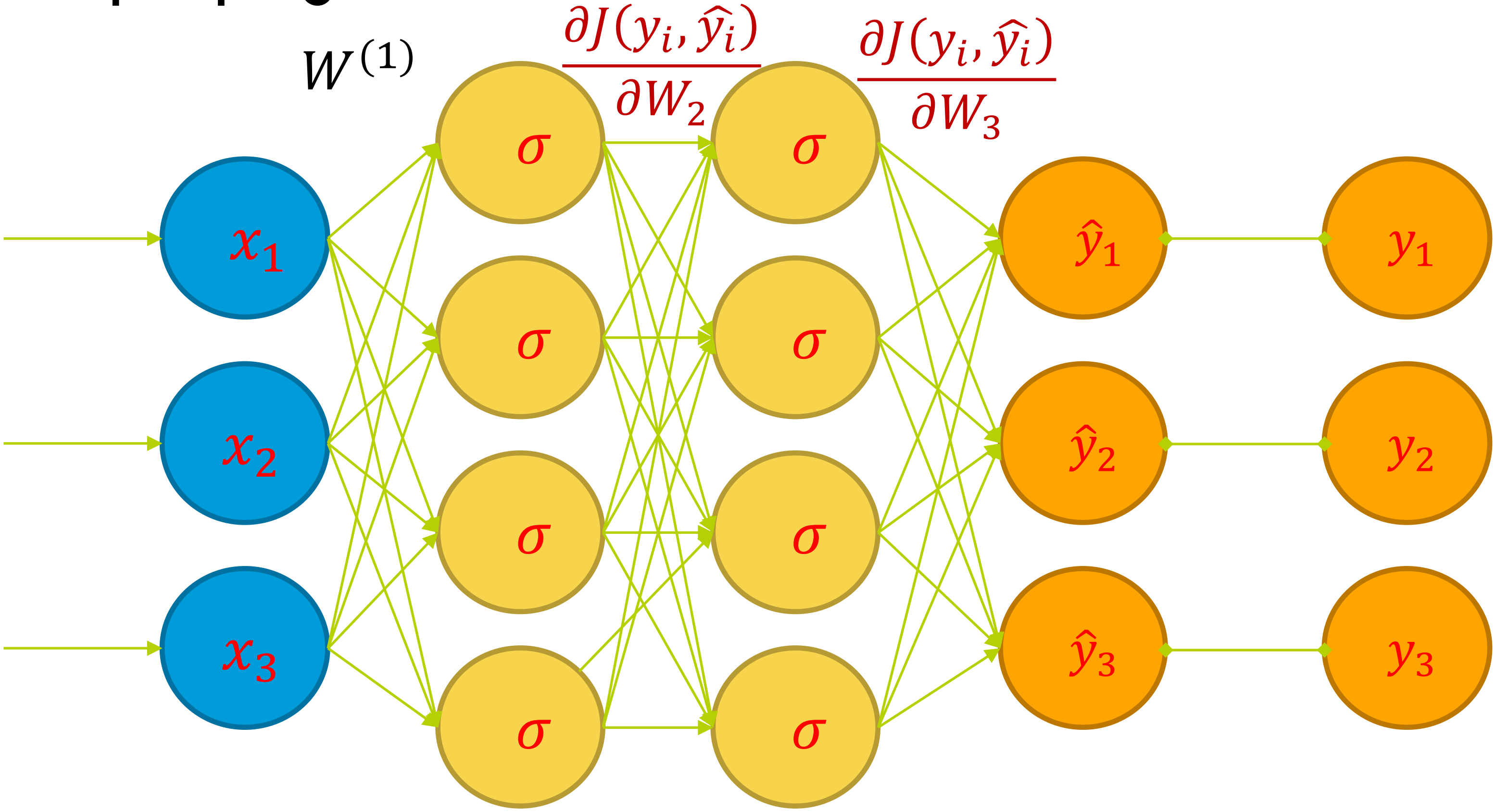
Backpropagation



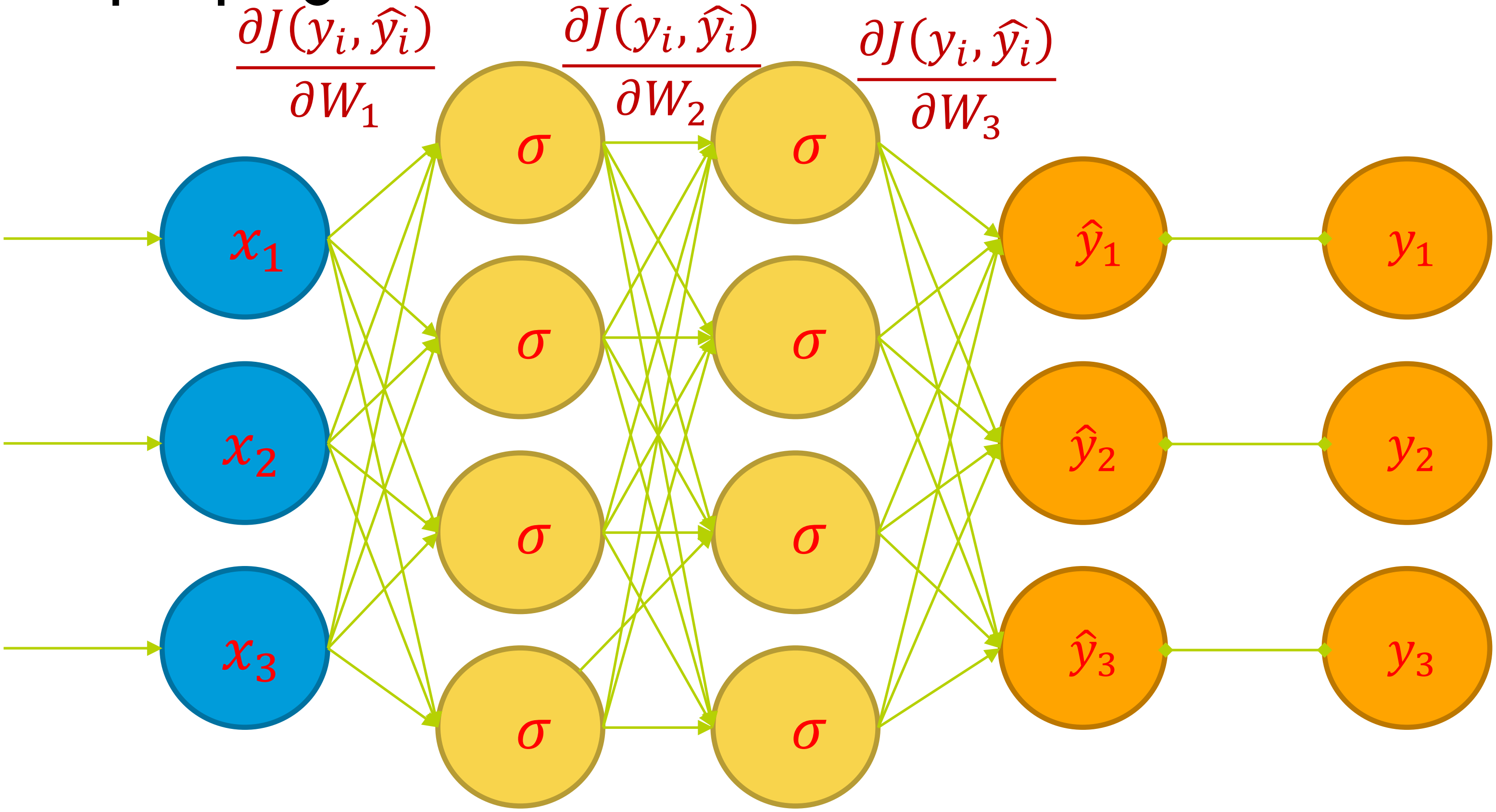
Backpropagation



Backpropagation



Backpropagation



How have we trained before?

- Gradient Descent!

1. Make prediction
2. Calculate Loss
3. Calculate gradient of the loss function w.r.t. parameters
4. Update parameters by taking a step in the opposite direction
5. Iterate

Vanishing Gradients

Recall that:

$$\frac{\partial J}{\partial W^{(1)}} = (\hat{y} - y) \cdot W^{(3)} \cdot \sigma'\left(z^{(3)}\right) \cdot W^{(2)} \cdot \sigma'\left(z^{(2)}\right) \cdot X$$

gradient : for updating the weight
the value of gradient is too low due to this sigmoid

(0.5)(1 - 0.5) = 0.25
(0.25)(0.25) = …

assume if the layer is more than 100 then …

- Remember: $\sigma'(z) = \sigma(z)\big(1 - \sigma(z)\big) \leq .25$
- As we have more layers, the gradient gets very small at the early layers.
- This is known as the "vanishing gradient" problem.
- For this reason, other activations (such as ReLU) have become more common.

Other Activation Functions

Hyperbolic Tangent Function

- Hyperbolic tangent function
- Pronounced “tanch”

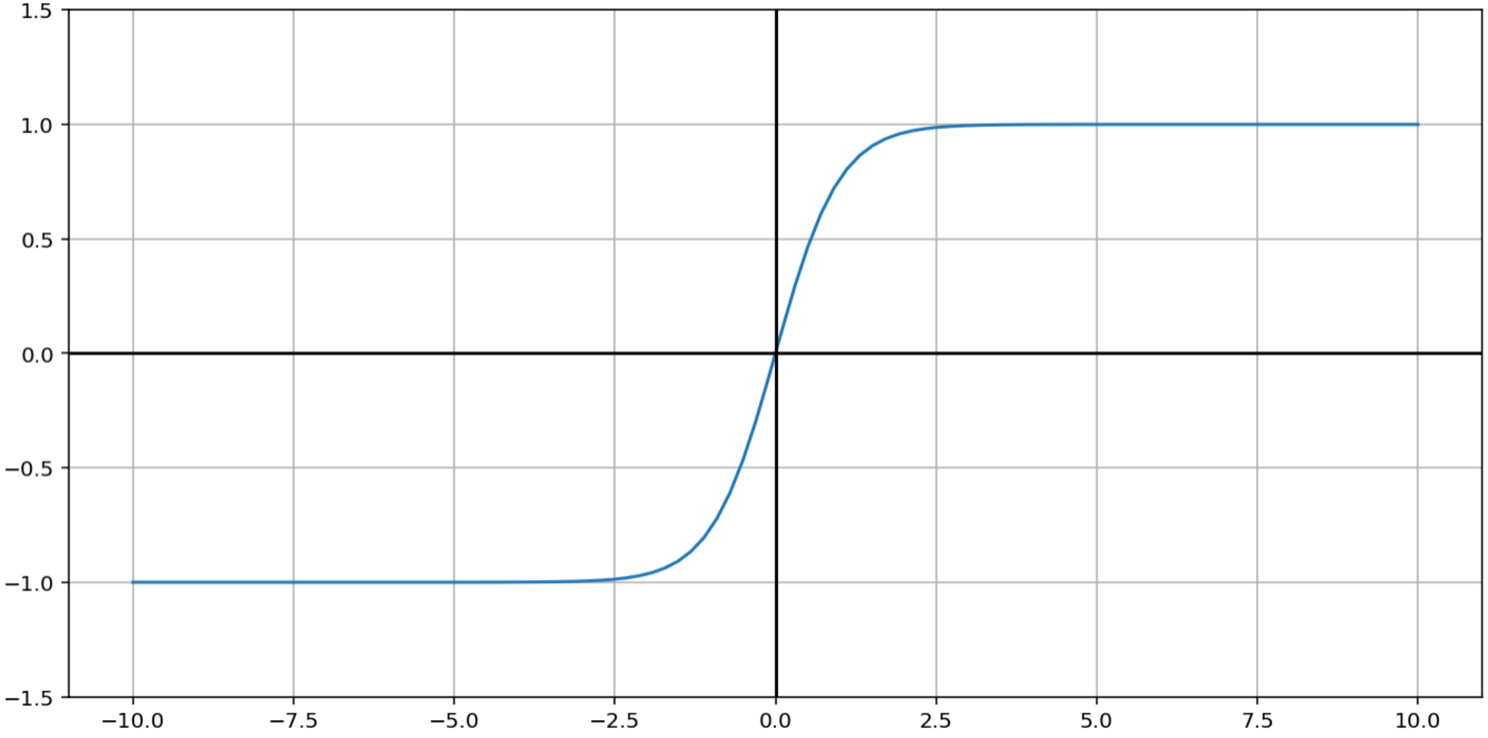
$$tanh(z) = \frac{\sinh(z)}{\cosh(z)} = \frac{e^{2x} - 1}{e^{2x} + 1}$$

$$tanh(0) = 0$$

$$tanh(\infty) = 1$$

$$tanh(-\infty) = -1$$

Hyperbolic Tangent Function



Rectified Linear Unit (ReLU)

$$ReLU(z) = \begin{cases} 0, & z < 0 \\ z, & z \geq 0 \end{cases}$$

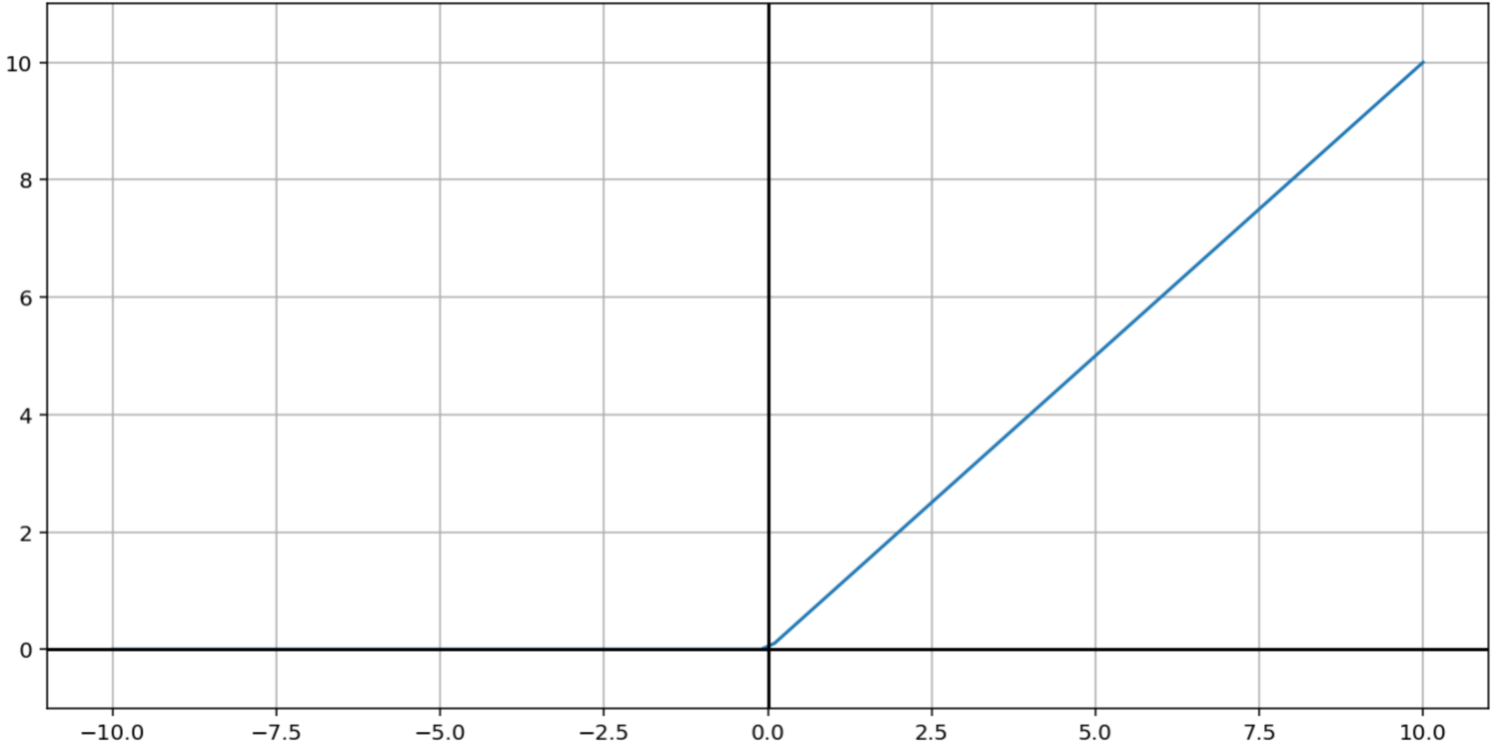
$$= \max(0, z)$$

$$ReLU(0) = 0$$

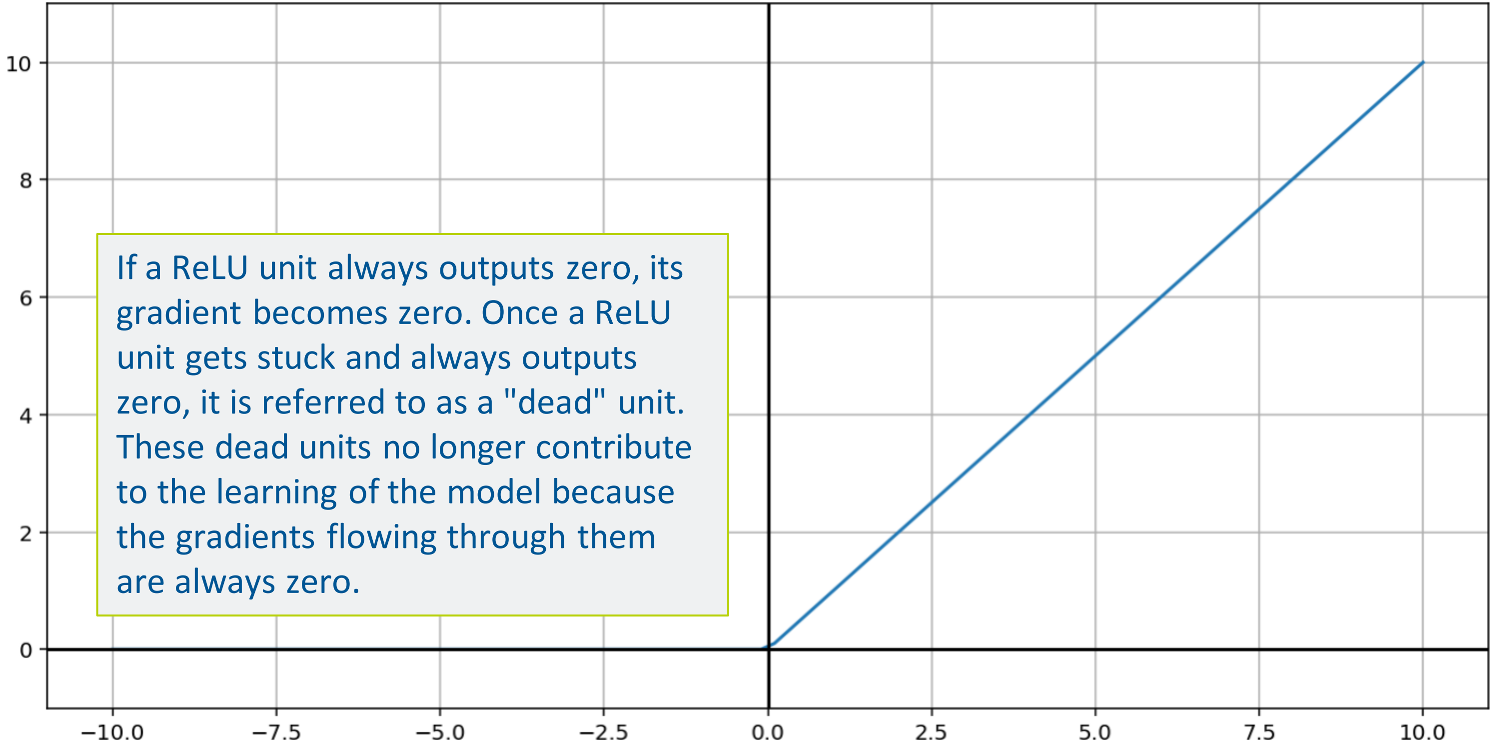
$$ReLU(z) = z \qquad \text{for } (z \gg 0)$$

$$ReLU(-z) = 0$$

Rectified Linear Unit (ReLU)



Dying ReLU Problem



If a ReLU unit always outputs zero, its gradient becomes zero. Once a ReLU unit gets stuck and always outputs zero, it is referred to as a "dead" unit. These dead units no longer contribute to the learning of the model because the gradients flowing through them are always zero.

"Leaky" Rectified Linear Unit (ReLU)

$$LReLU(z) = \begin{cases} \alpha z, & z < 0 \\ z, & z \geq 0 \end{cases}$$

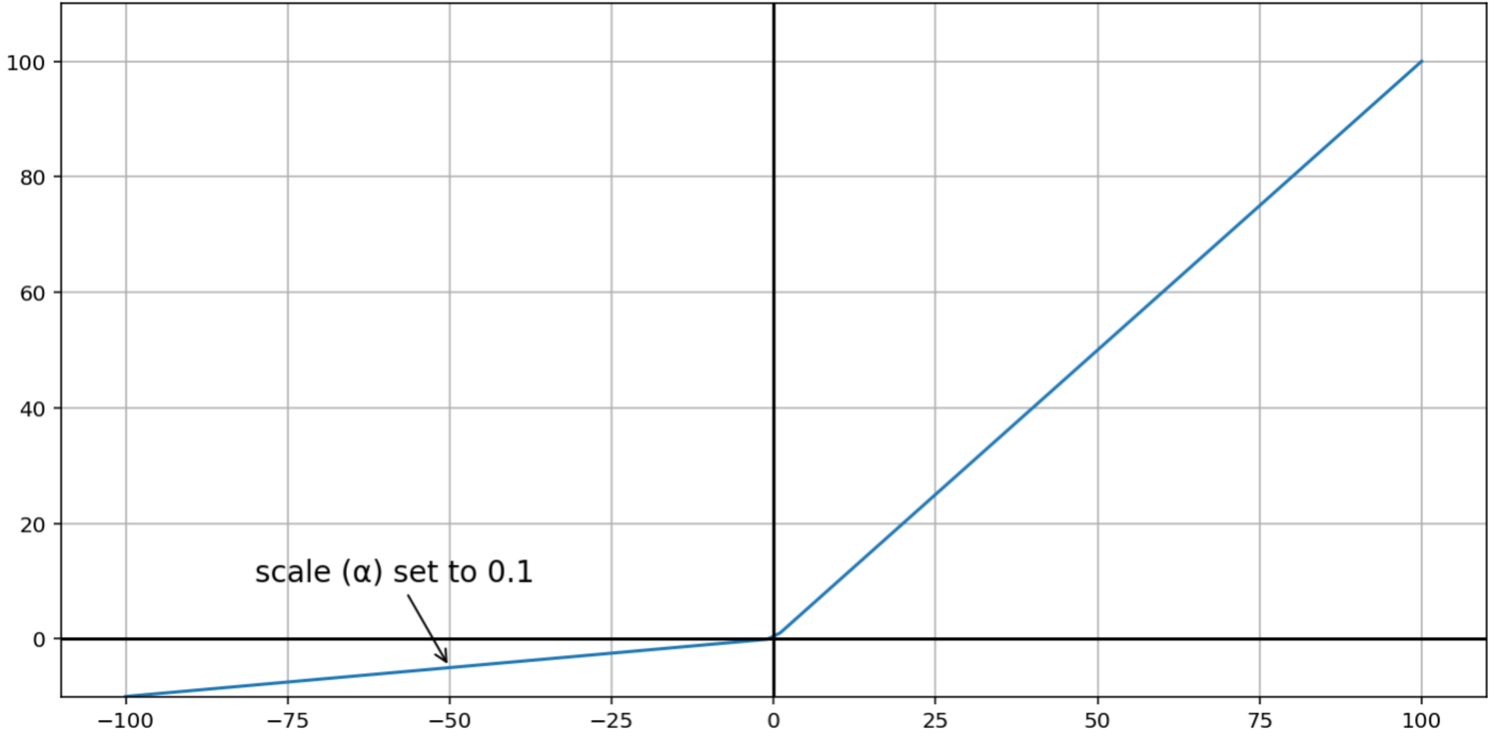
$$= \max(\alpha z, z) \quad \text{for } (\alpha < 1)$$

$$LReLU(0) = 0$$

$$LReLU(z) = z \qquad \text{for } (z \gg 0)$$

$$LReLU(-z) = -\alpha z$$

“Leaky” Rectified Linear Unit (ReLU)



What next?

- We now know how to make a single update to a model given some data.
- But how do we do the full training?
- We will dive into these details in the next lecture.

Software