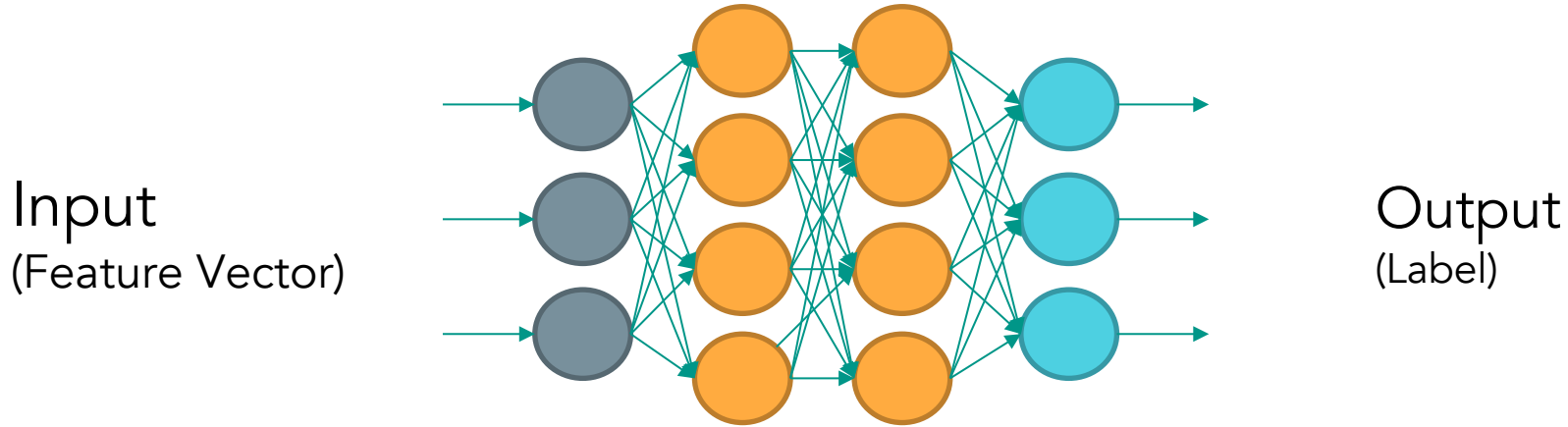


Backpropagation in Neural Nets



METIS

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?

- Gradient Descent
- Make prediction
- Calculate Loss
- Calculate Gradient w.r.t. Weights
- Update Weights by taking a step in that direction
- Iterate

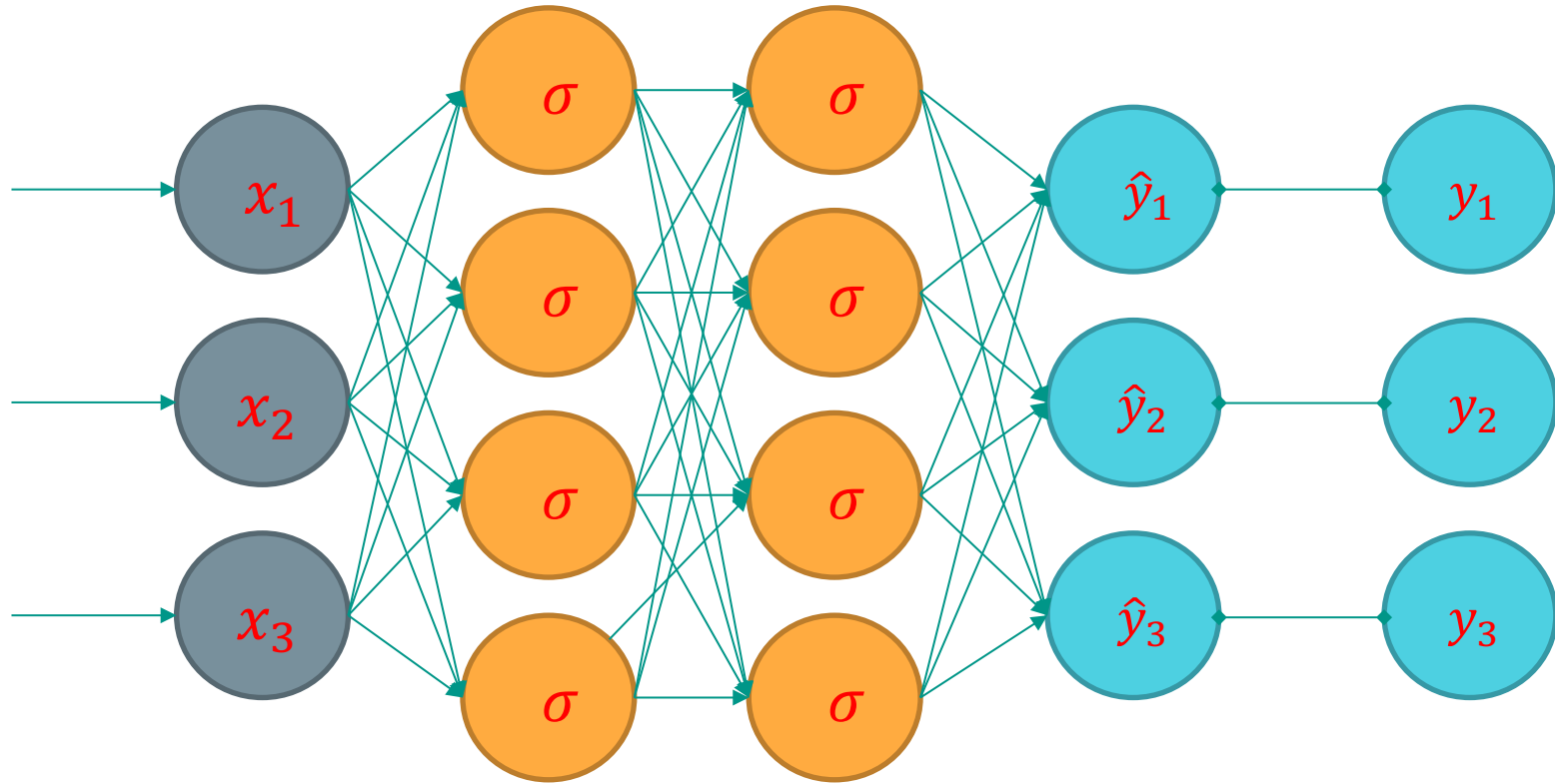


Step 1

- Make prediction
- forward propagate
- get predictions
- calculate loss

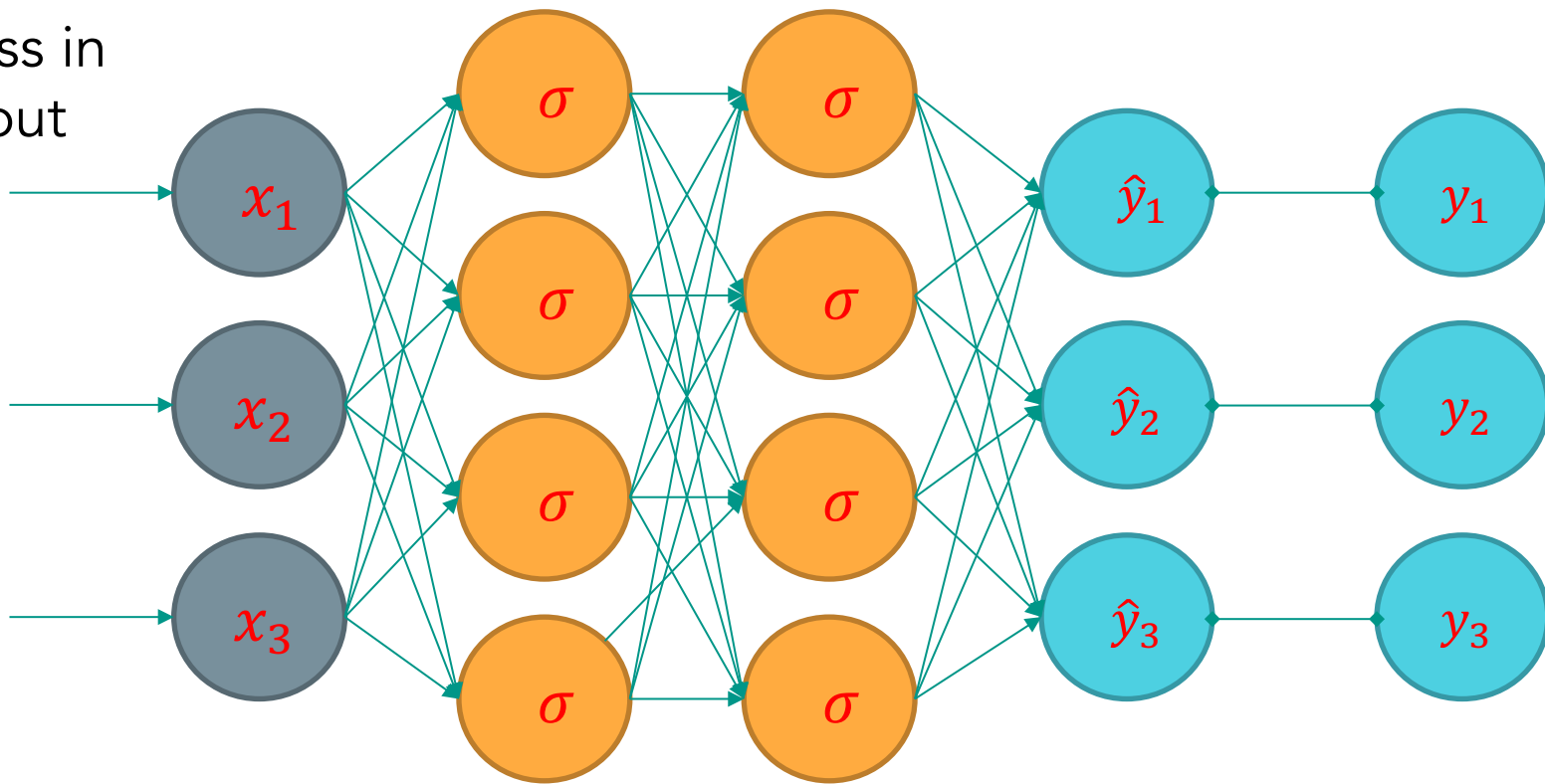


Feedforward Neural Network



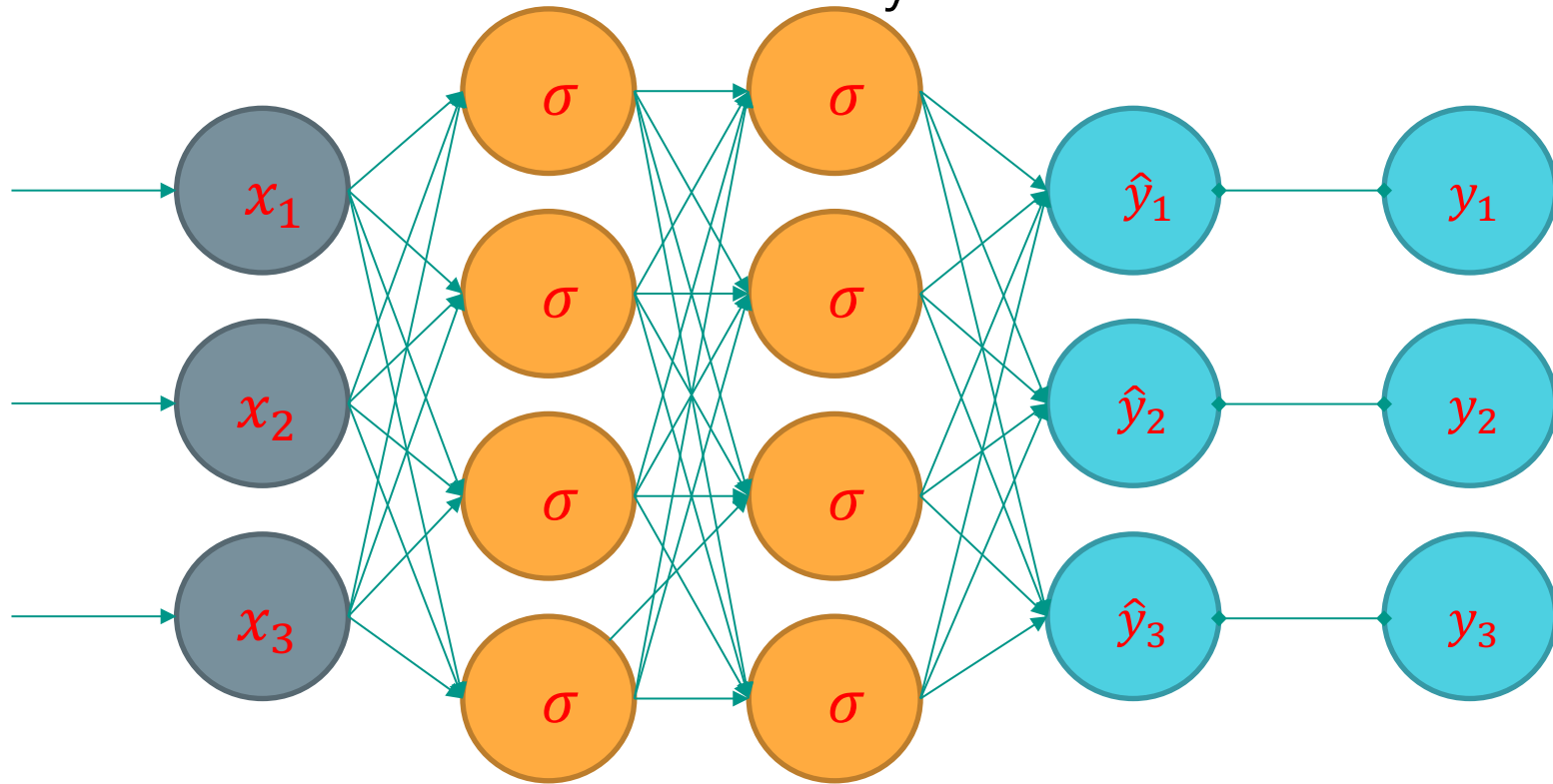
Forward Propagation

Pass in
Input



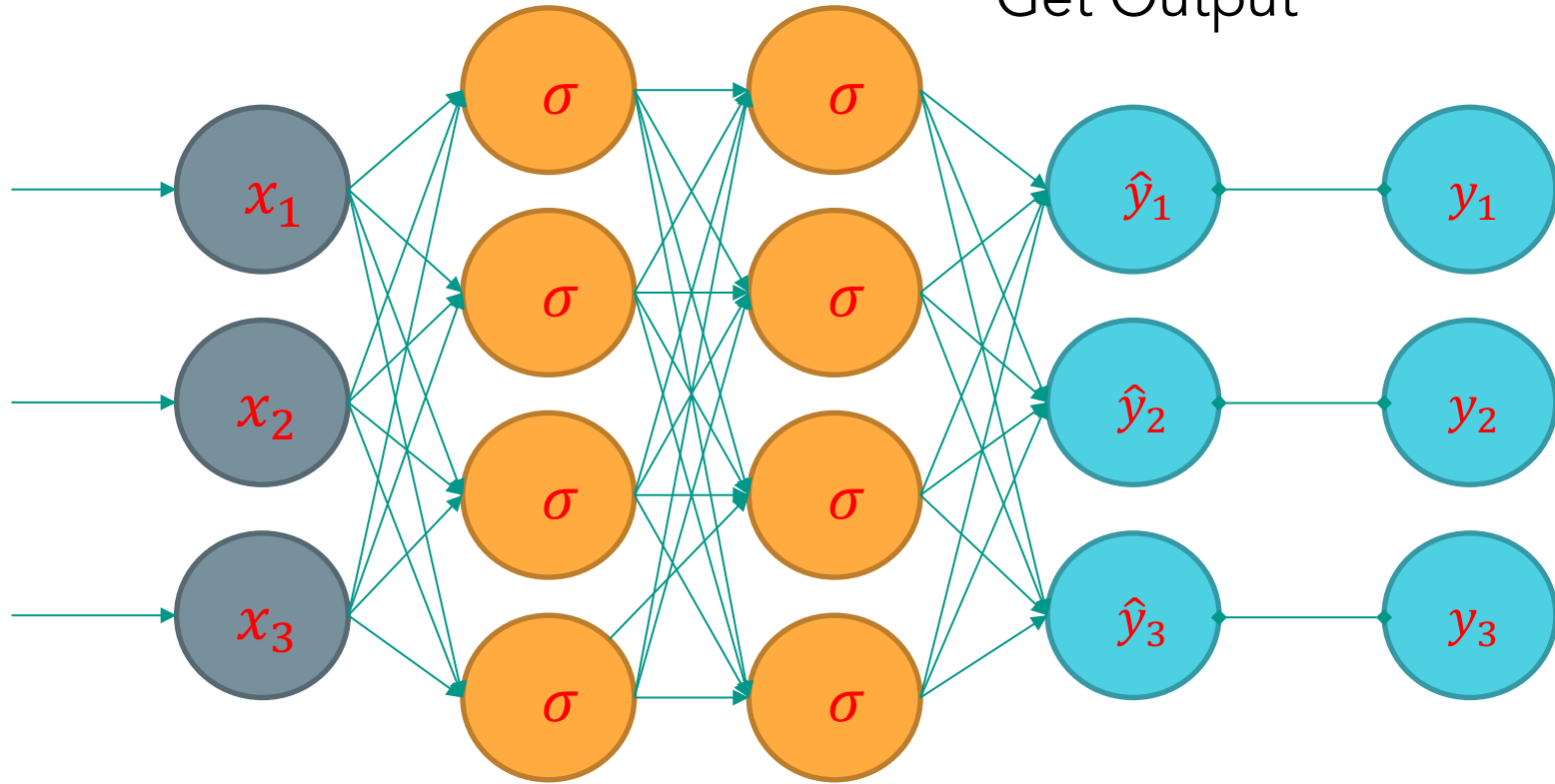
Forward Propagation

Calculate each Layer



Forward Propagation

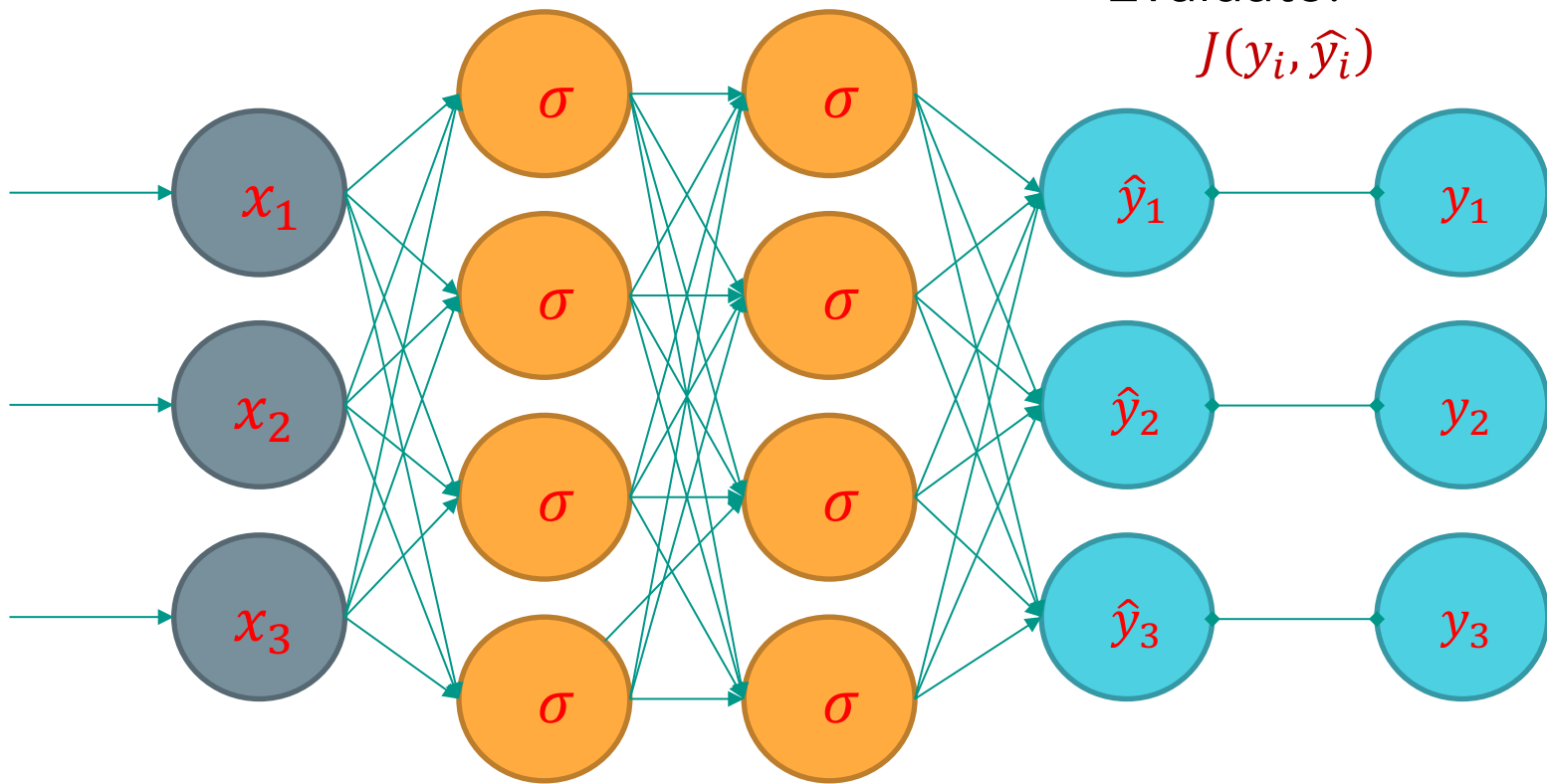
Get Output



Forward Propagation

Evaluate:

$$J(y_i, \hat{y}_i)$$



How have we trained before?

- Gradient Descent
- Make prediction
- Calculate Loss
- Calculate Gradient w.r.t. Weights
- Update Weights by taking a step in that direction
- 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 $F: X \rightarrow Y$
- 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 $J(y, F(x))$

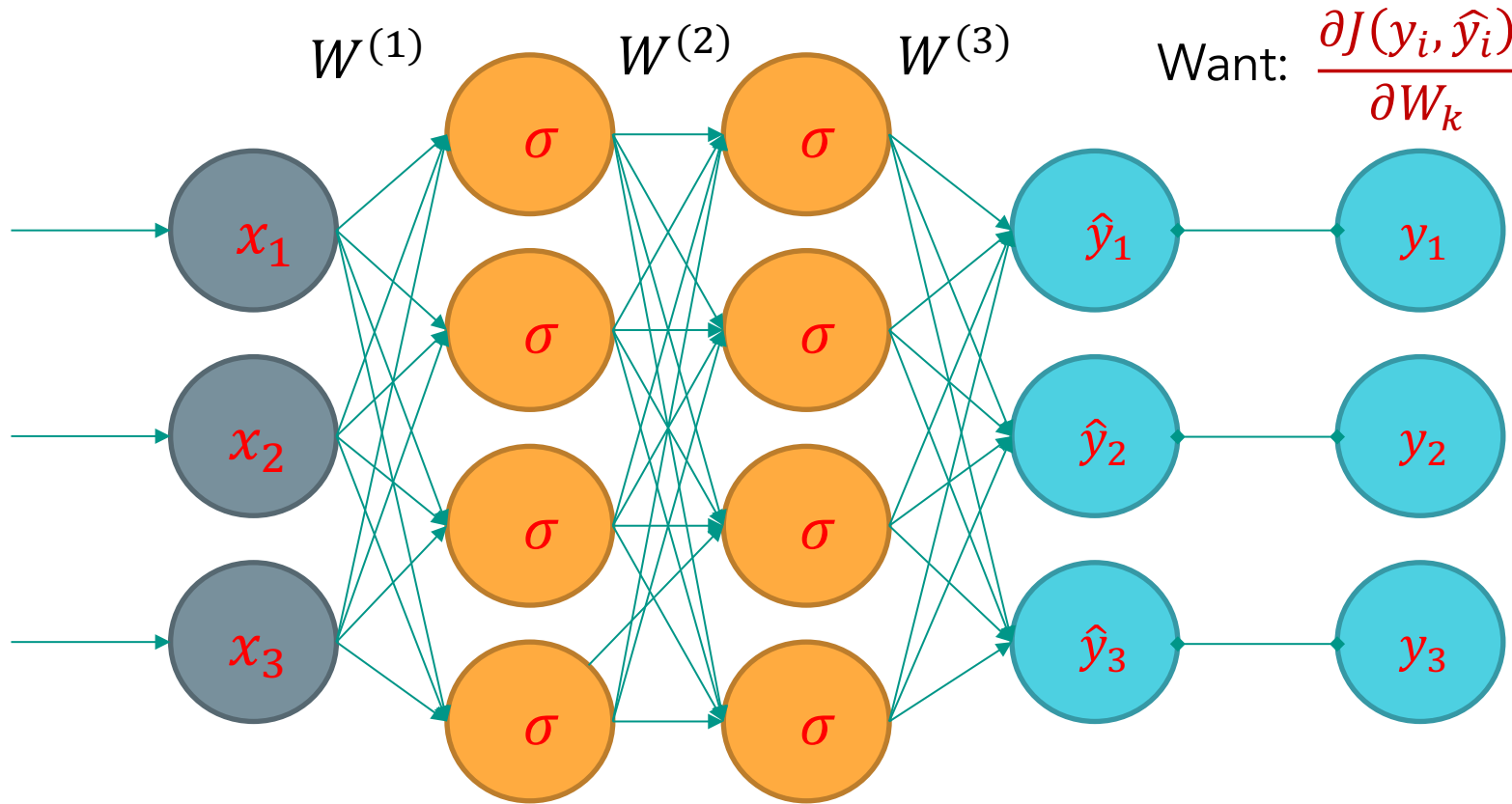


How to Train a Neural Net?

- 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



Punchline

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

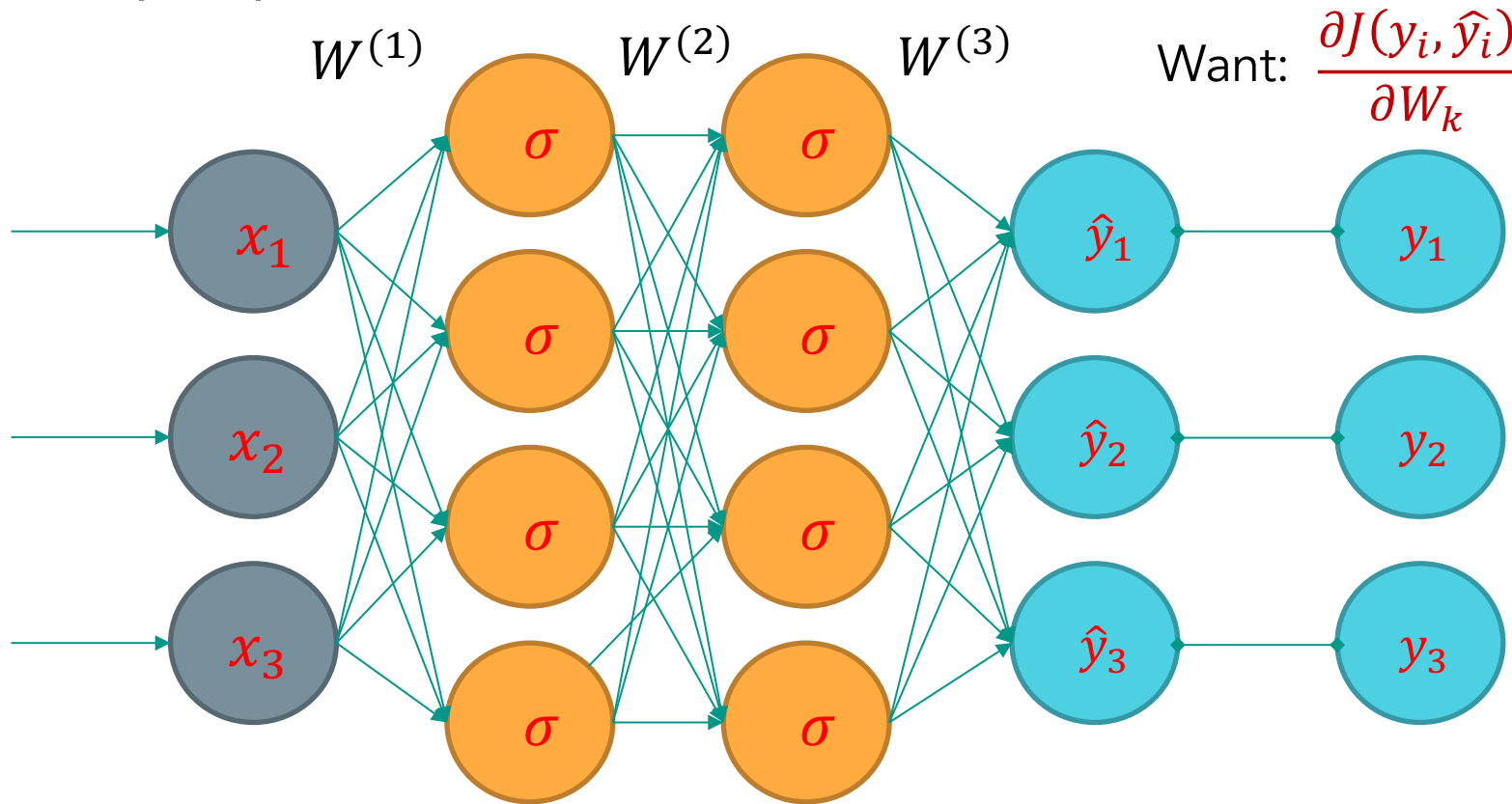
$$\frac{\partial J}{\partial W^{(2)}} = (\hat{y} - y) \cdot W^{(3)} \cdot \sigma'(z^{(3)}) \cdot a^{(2)}$$

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

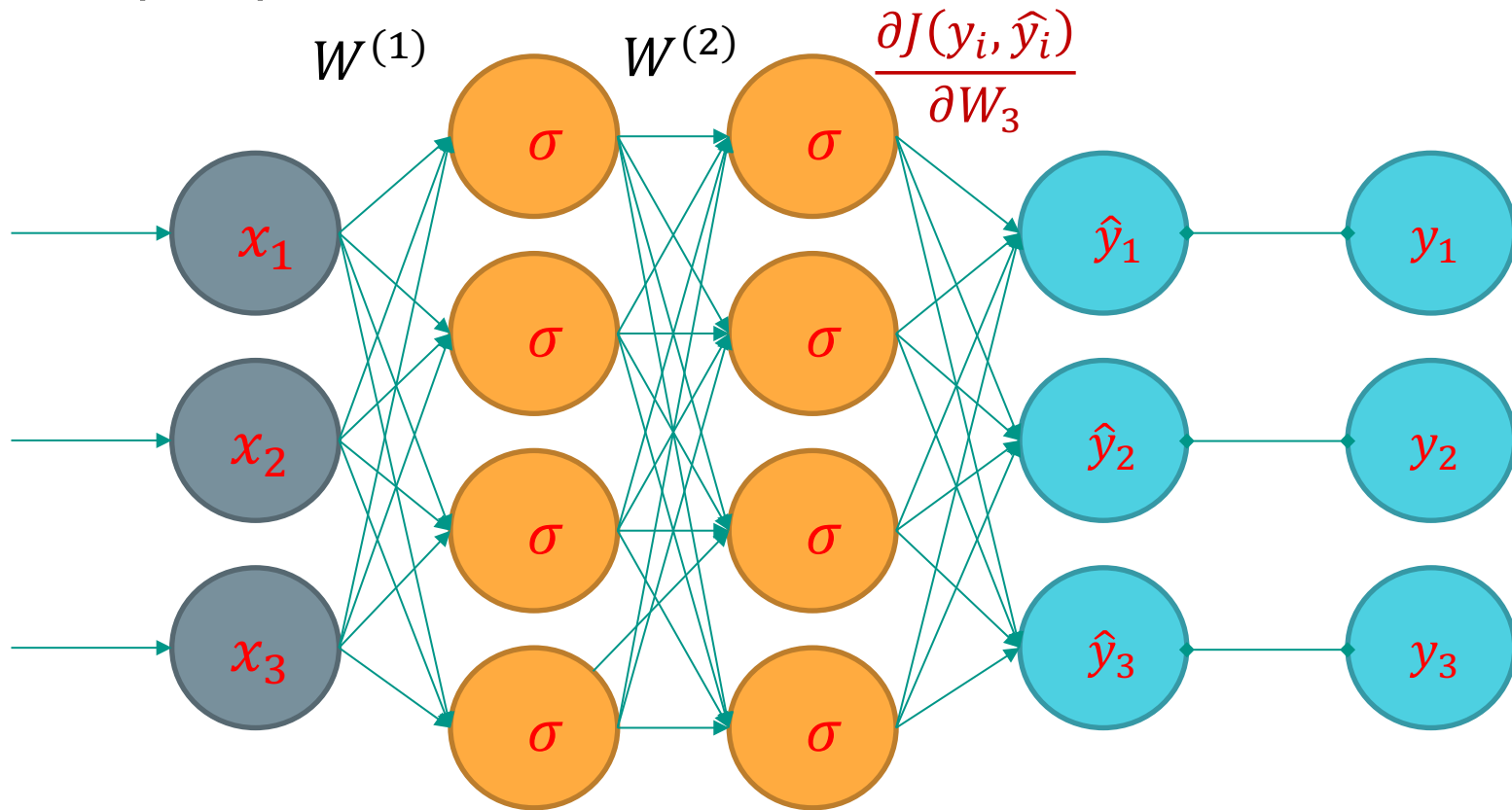
- Recall that: $\sigma'(z) = \sigma(z)(1 - \sigma(z))$
- Though they appear complex, above are easy to compute!



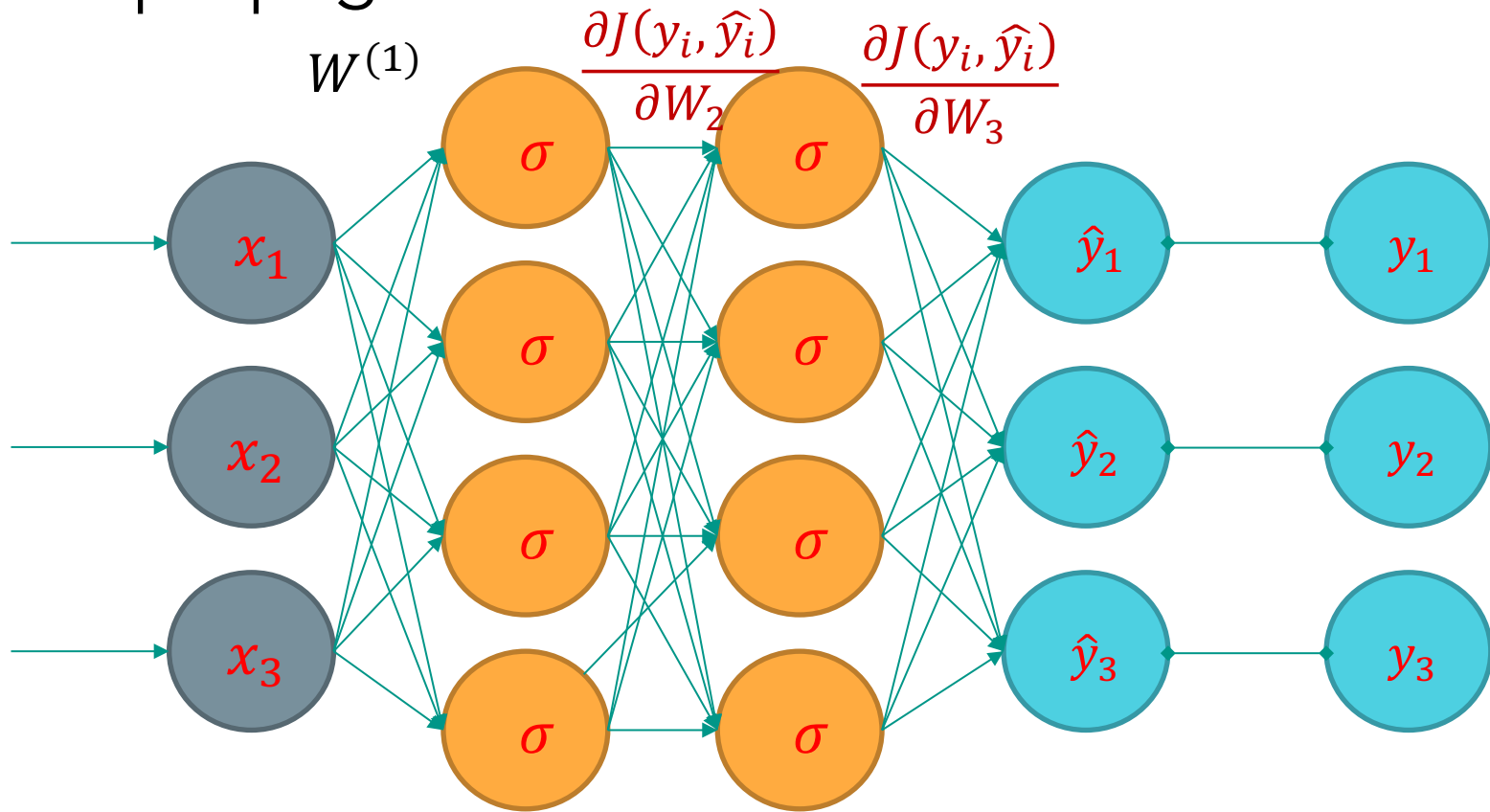
Backpropagation



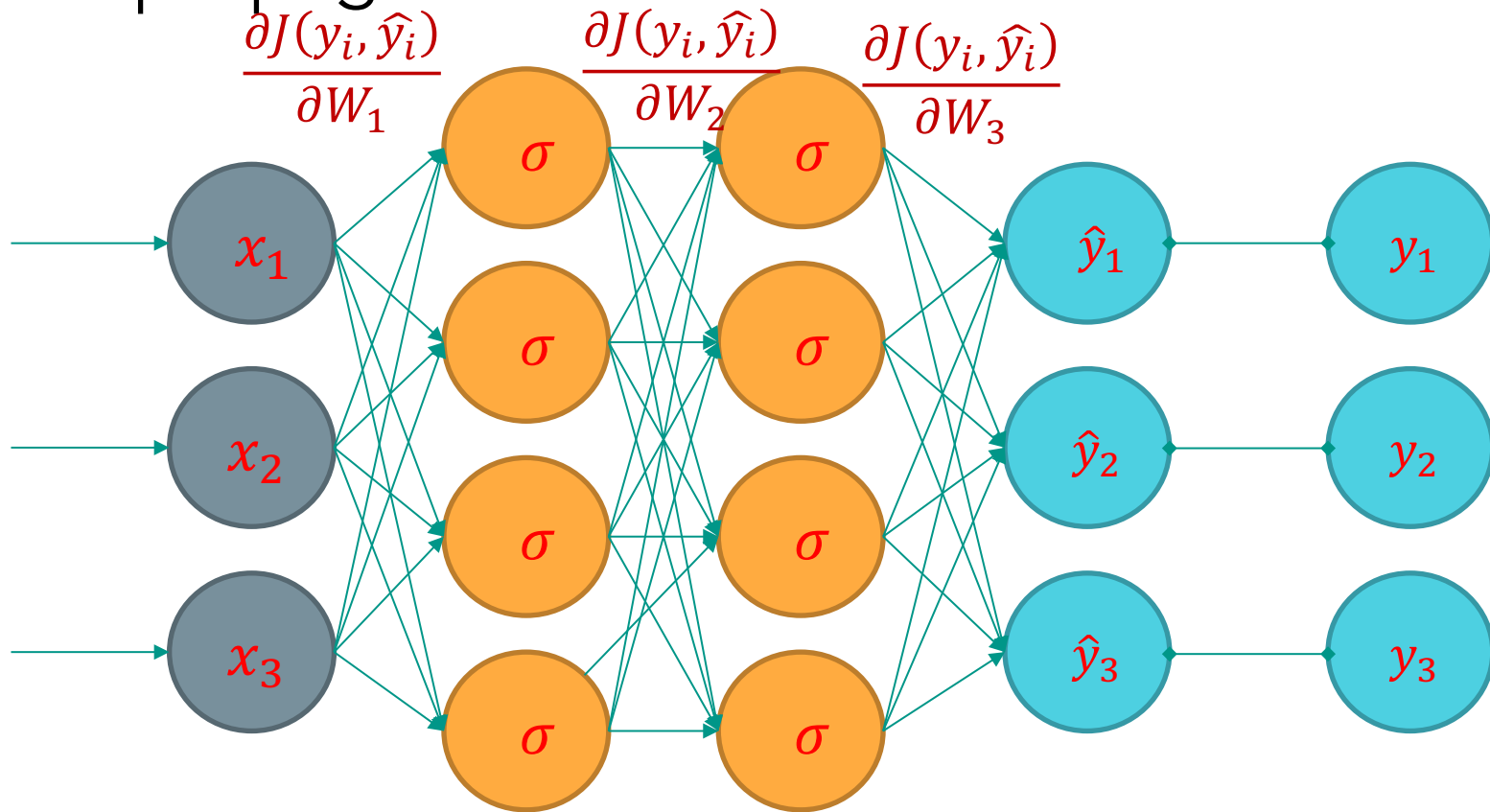
Backpropagation



Backpropagation



Backpropagation



How have we trained before?

- Gradient Descent
- Make prediction
- Calculate Loss
- Calculate Gradient w.r.t. Weights
- Update Weights by taking a step in that direction
- Iterate



How have we trained before?

- Gradient Descent
- Make prediction
- Calculate Loss
- Calculate Gradient w.r.t. Weights
- Update Weights by taking a step in that direction
- Iterate



Vanishing Gradients

Recall that:

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

- Remember: $\sigma'(z) = \sigma(z)(1 - \sigma(z)) \leq .25$
- 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"

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

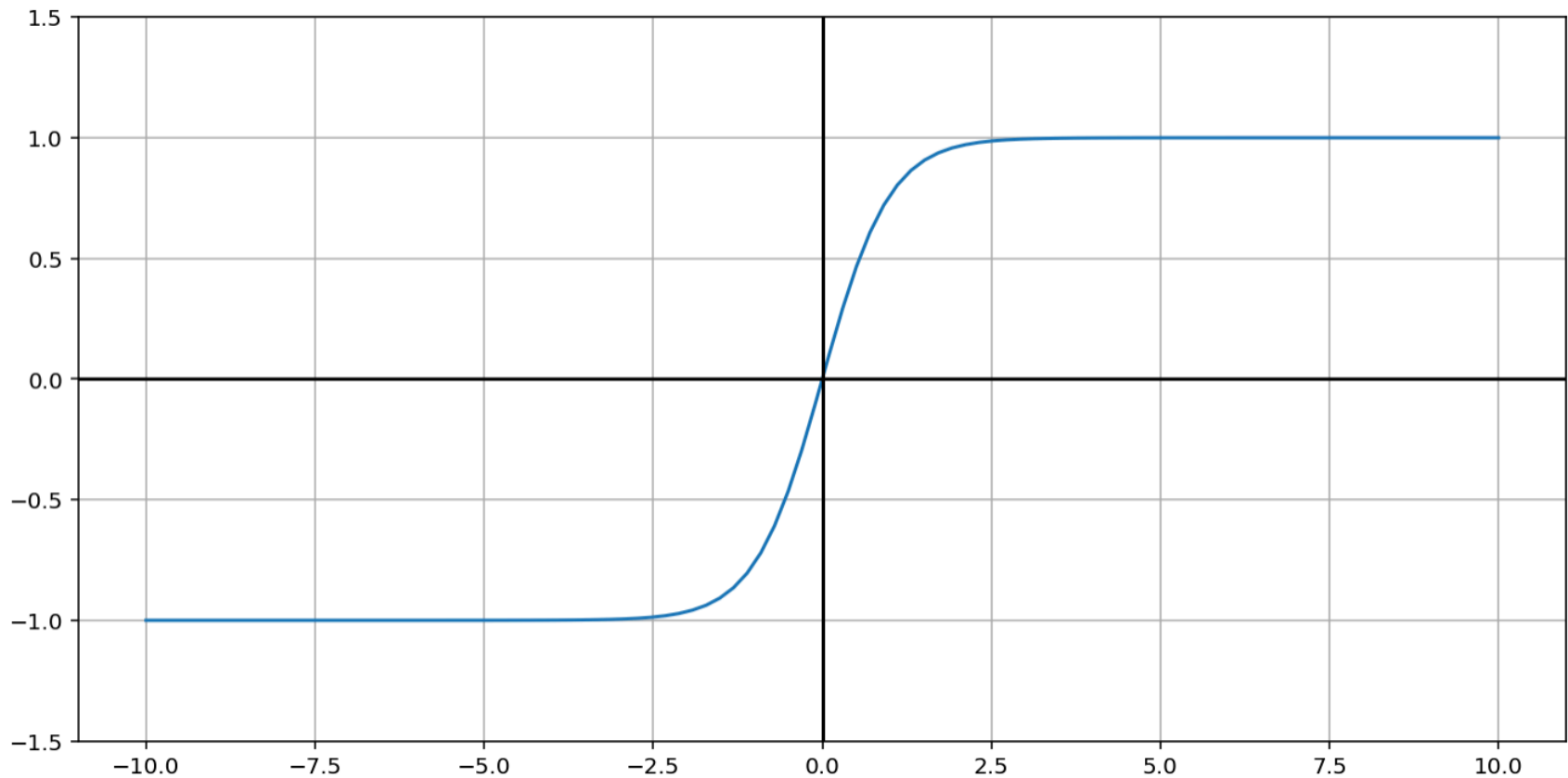
$$\tanh(0) = 0$$

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

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



Hyperbolic Tangent Function



Rectified Linear Unit (ReLU)

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

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

$$ReLU(0) = 0$$

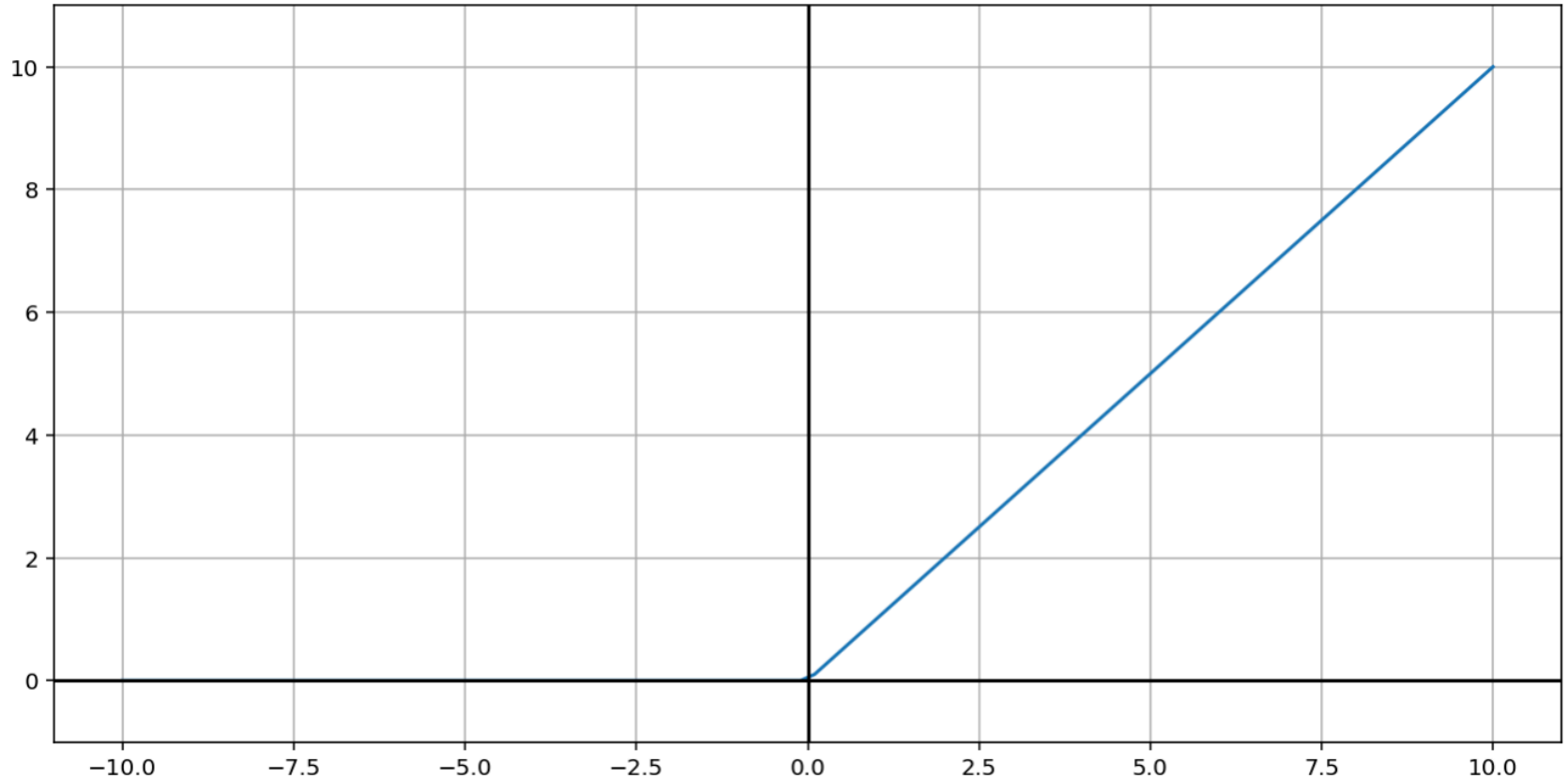
$$ReLU(z) = z$$

$$ReLU(-z) = 0$$

for ($z \gg 0$)



Rectified Linear Unit (ReLU)



“Leaky” Rectified Linear Unit (ReLU)

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

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

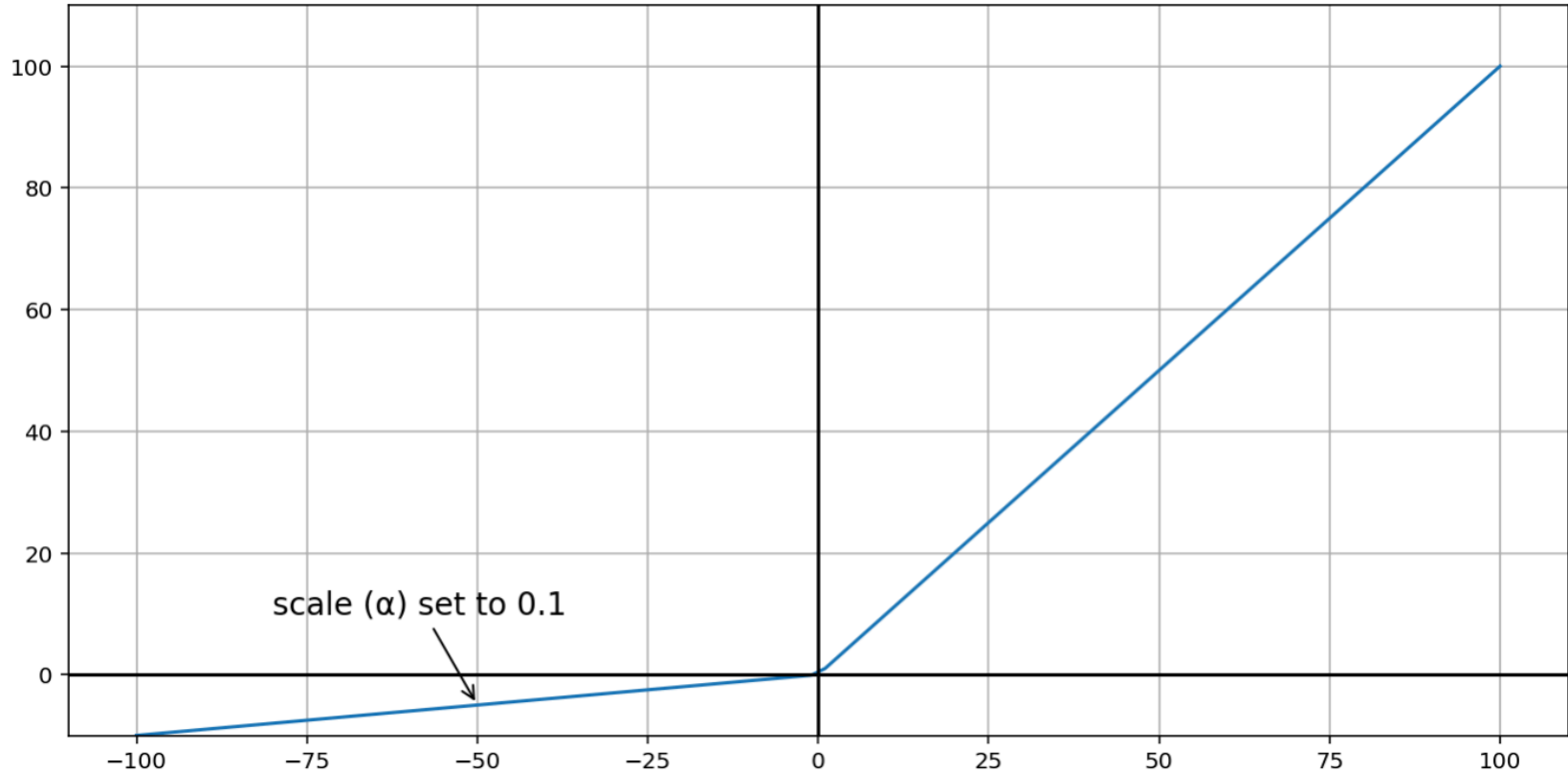
$$LReLU(0) = 0$$

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

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



"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.





METIS