Backpropagation in Neural Nets

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.

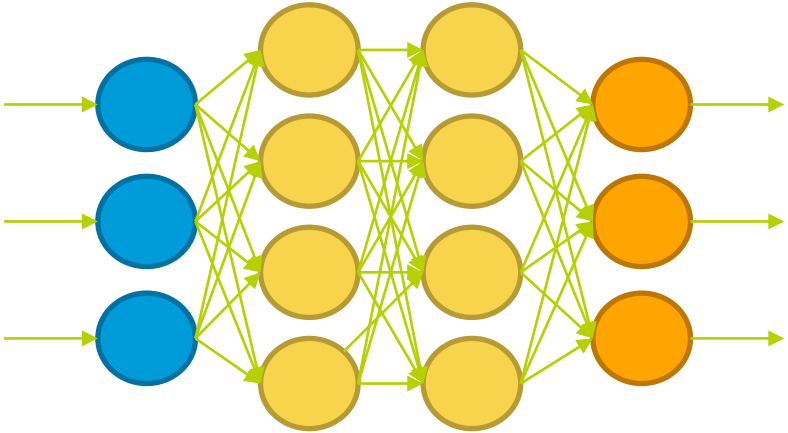
Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others.

Copyright © 2017, Intel Corporation. All rights reserved.

How to Train a Neural Net?

Input
(Feature Vector)



Output
(Label)

- 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!

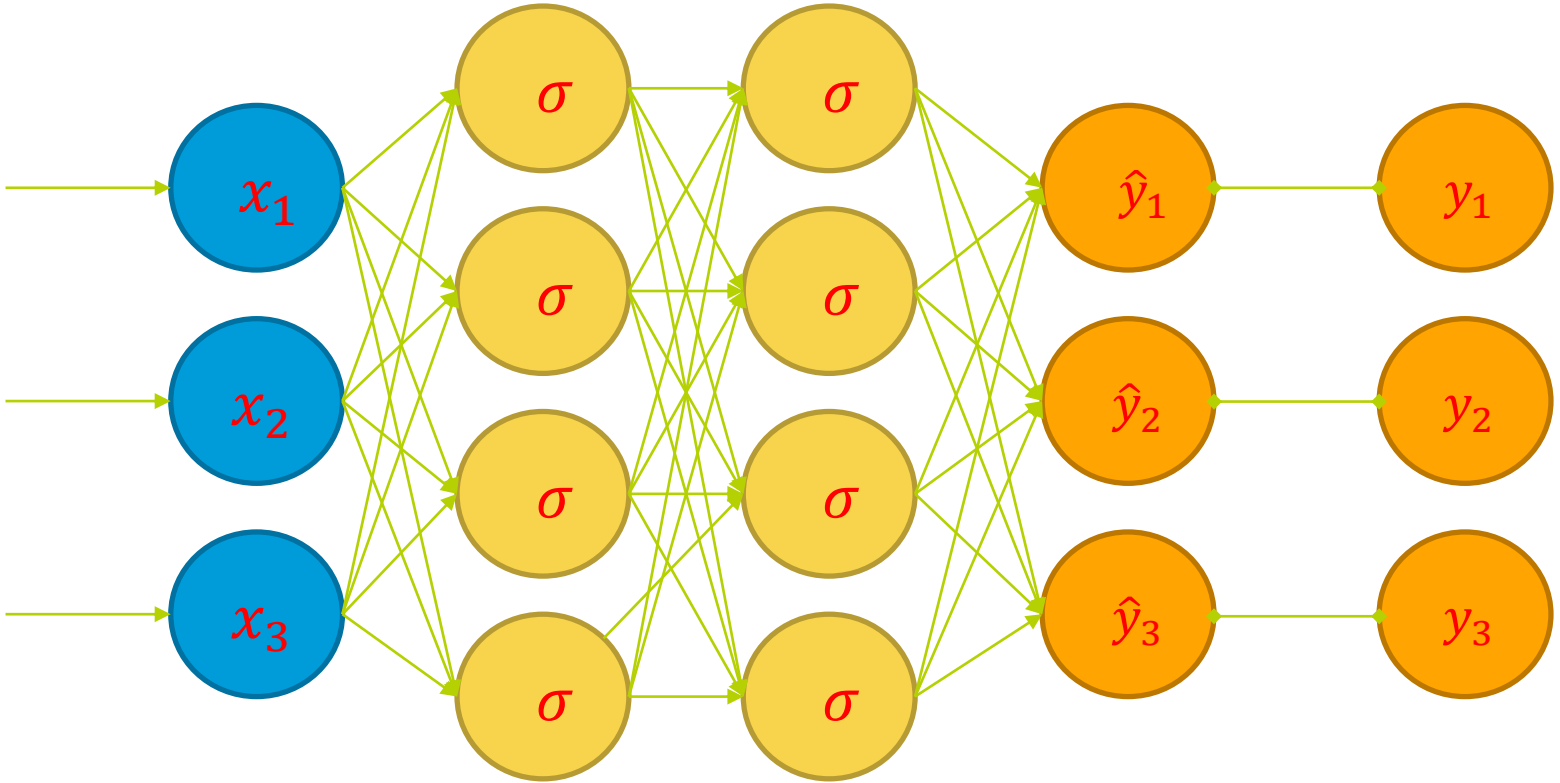
1. Make prediction
2. Calculate Loss
3. Calculate gradient of the loss function w.r.t. parameters
4. Update parameters by taking a step in the opposite direction
5. Iterate

How have we trained before?

- 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

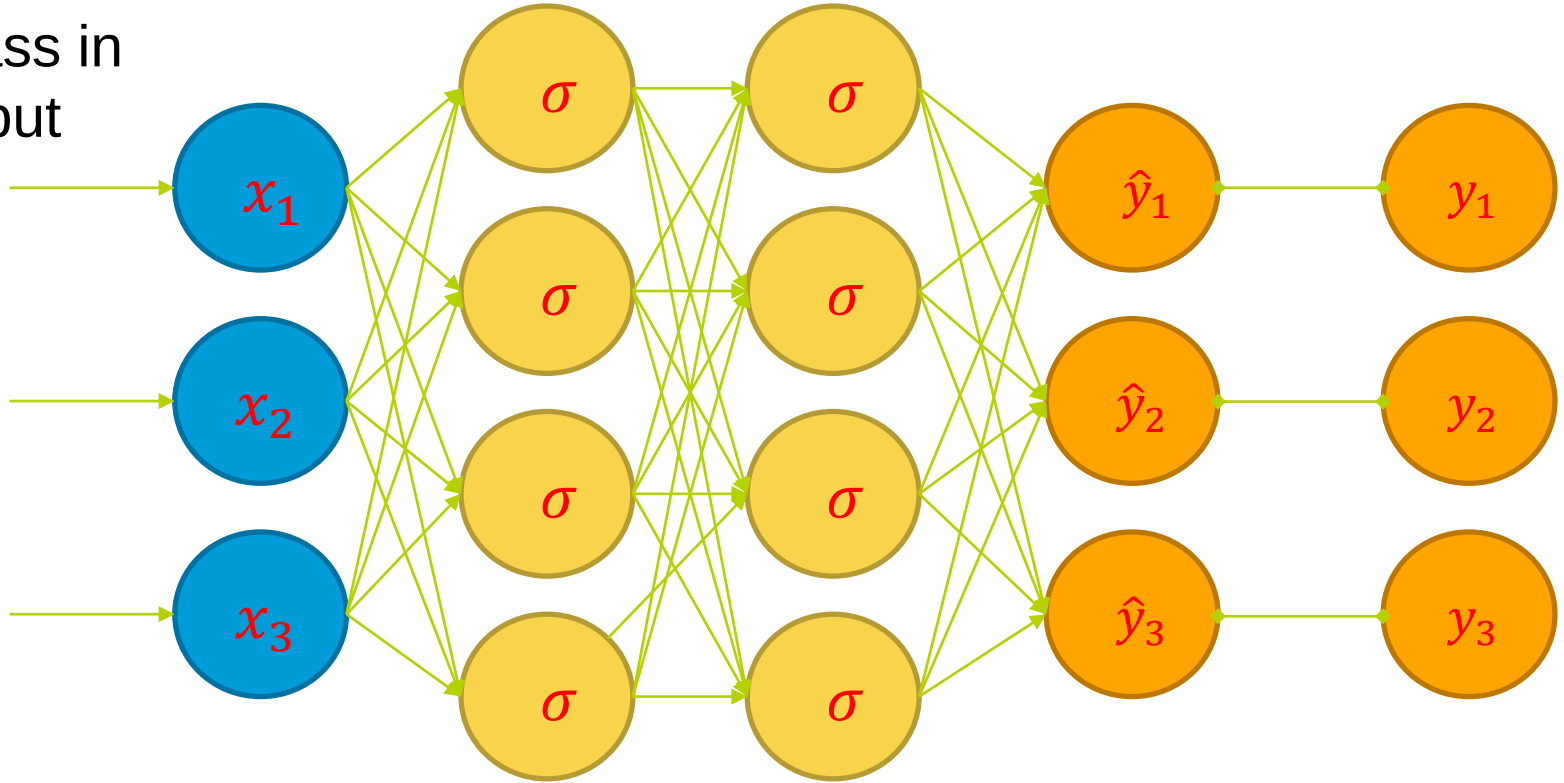
Feedforward Neural Network



intel Software

Forward Propagation

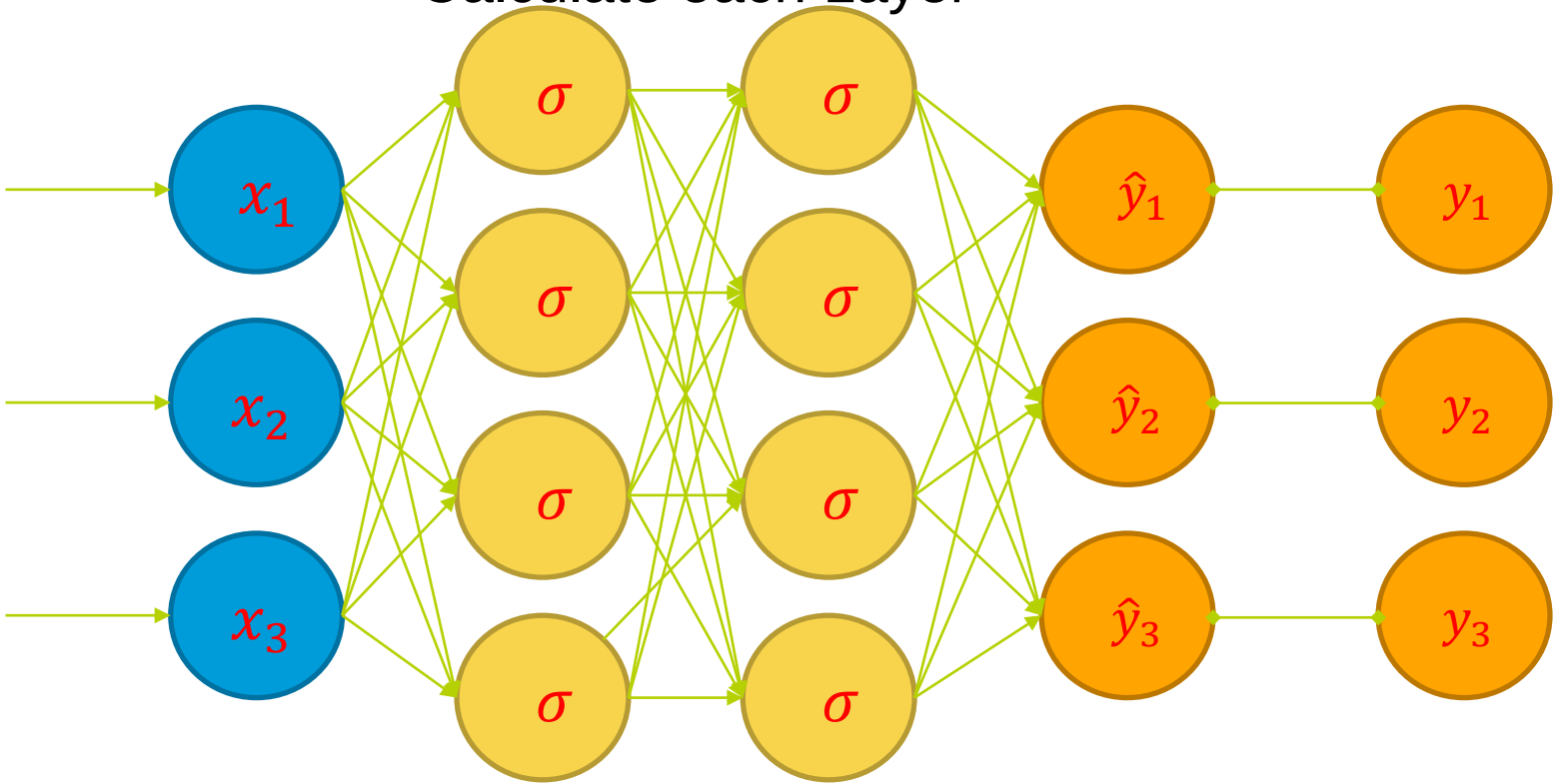
Pass in Input



intel Software

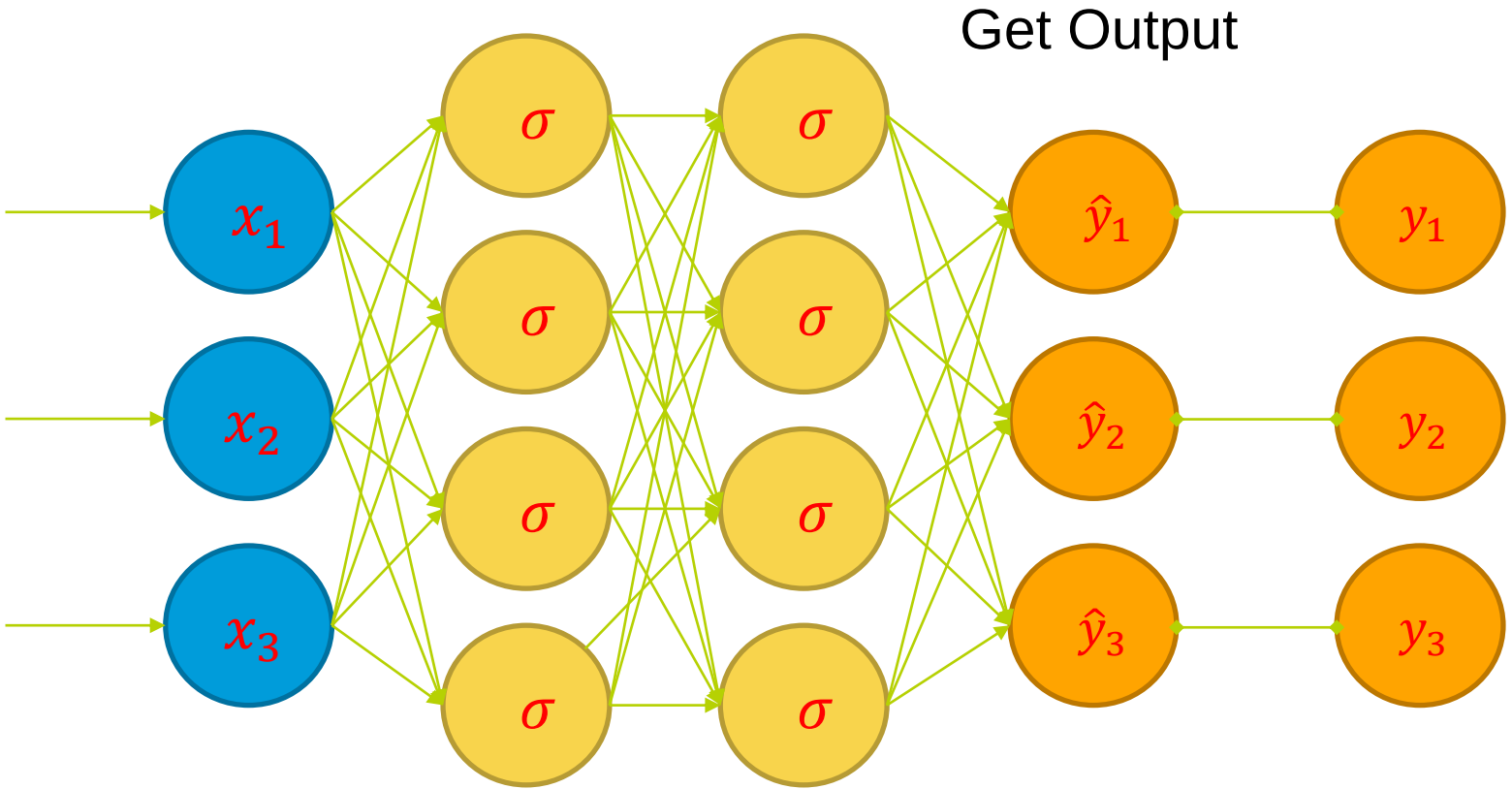
Forward Propagation

Calculate each Layer



intel Software

Forward Propagation

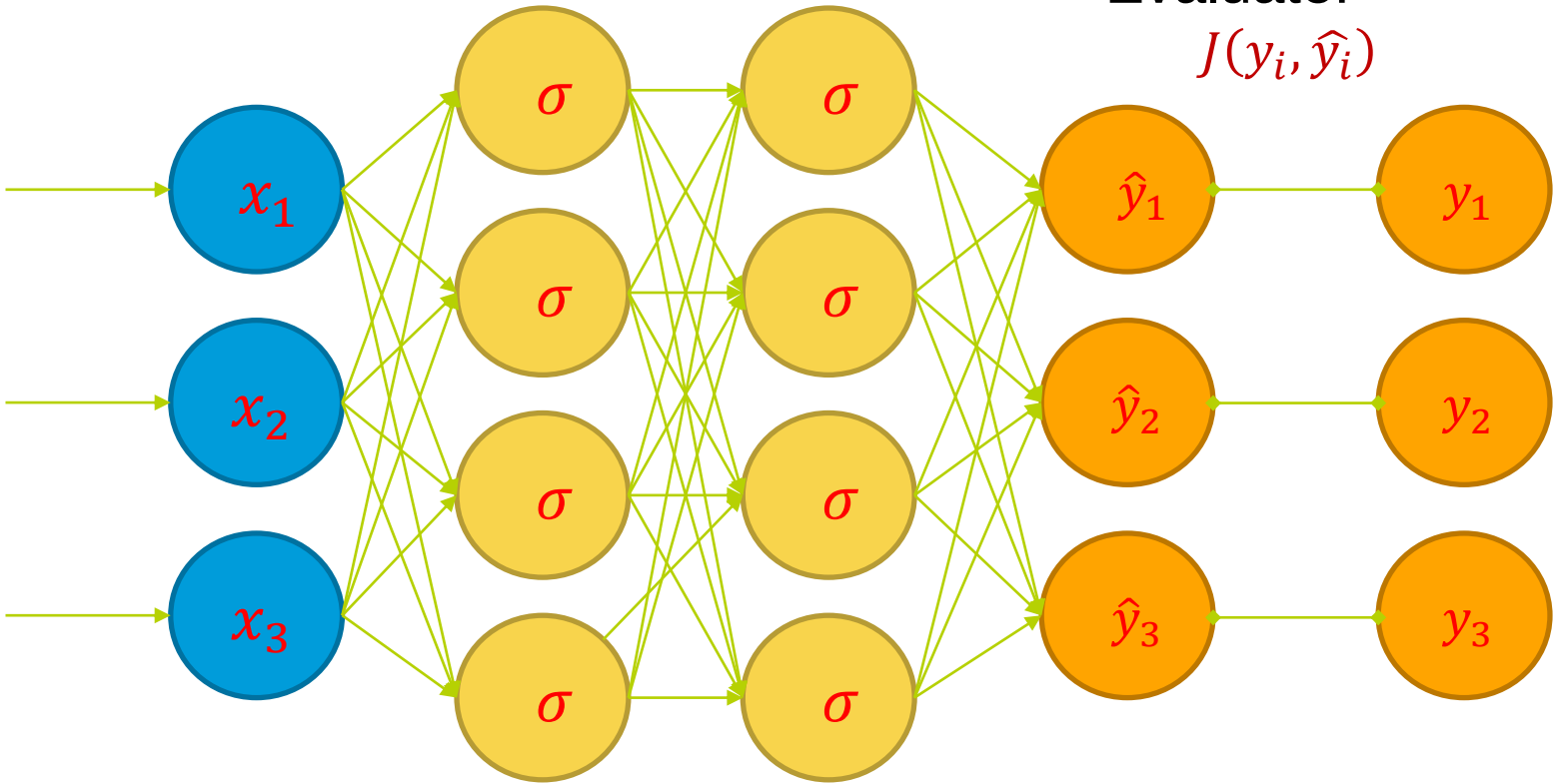


intel Software

Forward Propagation

Evaluate:

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



intel Software

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

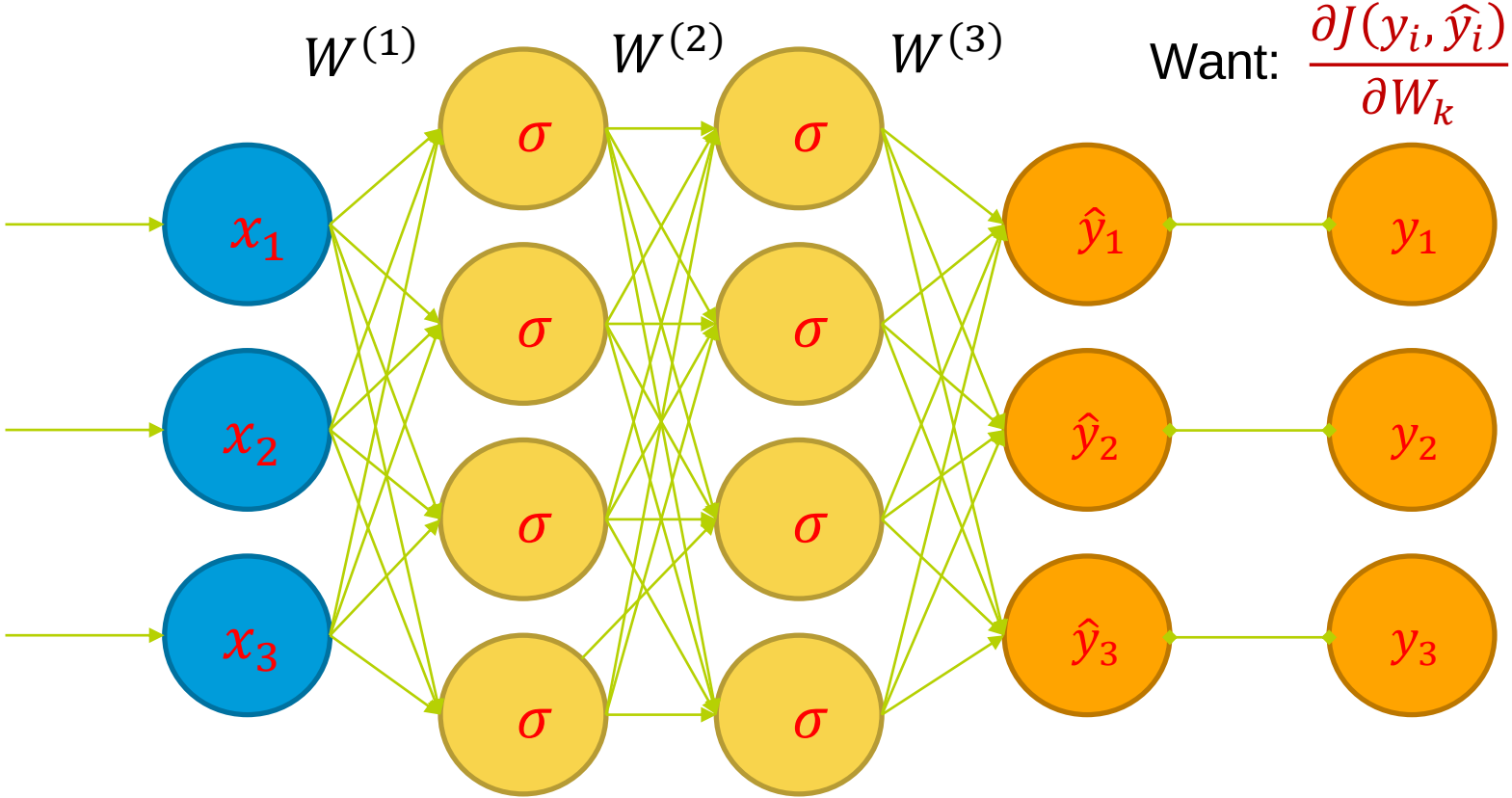
How to Train a Neural Net?

- How could we change the weights to make our Loss Function lower?
- Think of neural net as a function F: X -> 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



intel Software

Calculus to the Rescue

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

Punchline

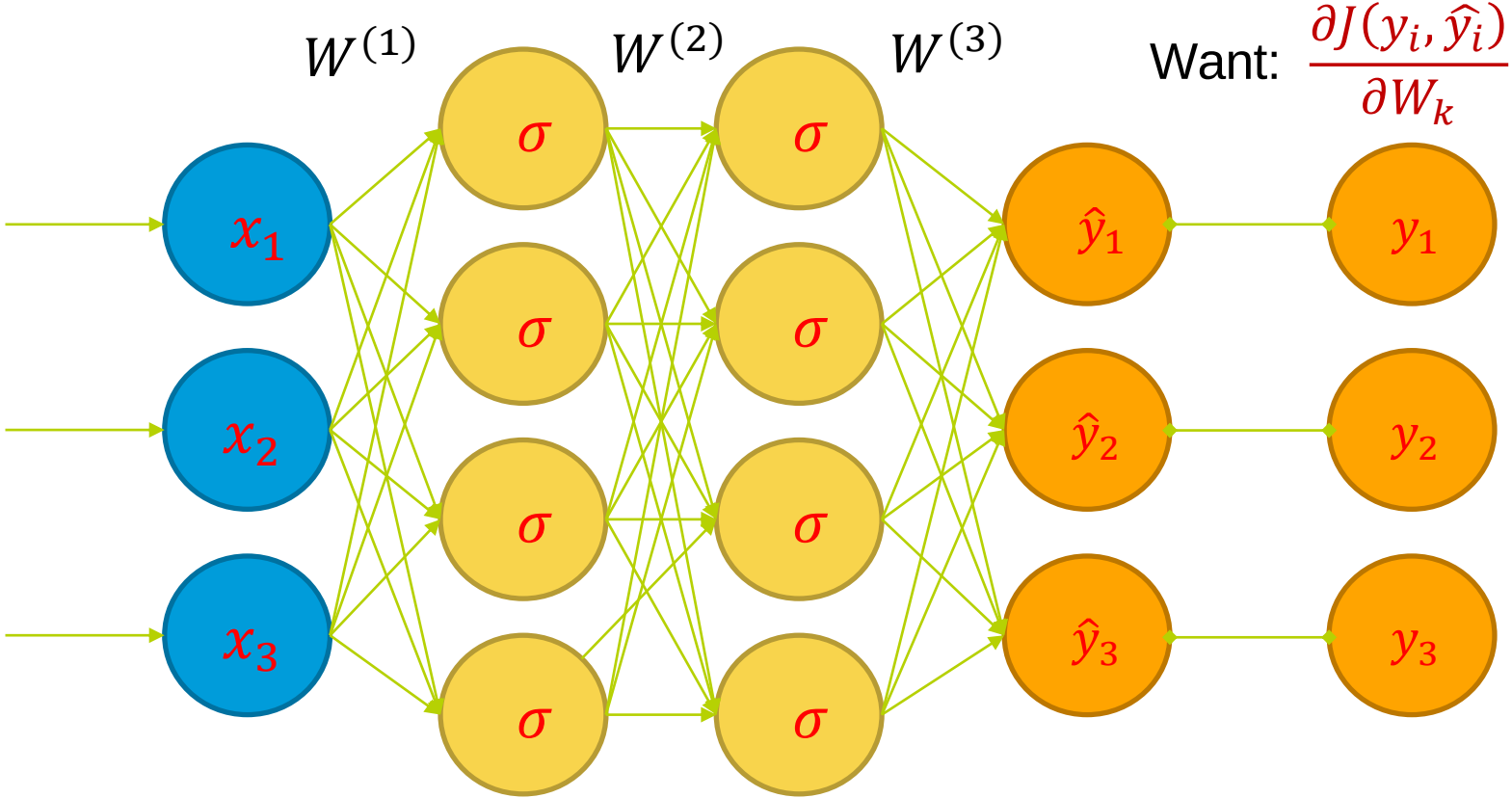
$$\frac{\partial J}{\partial W^{(3)}} = (\hat{y} - y) \cdot a^{(3)}$$

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

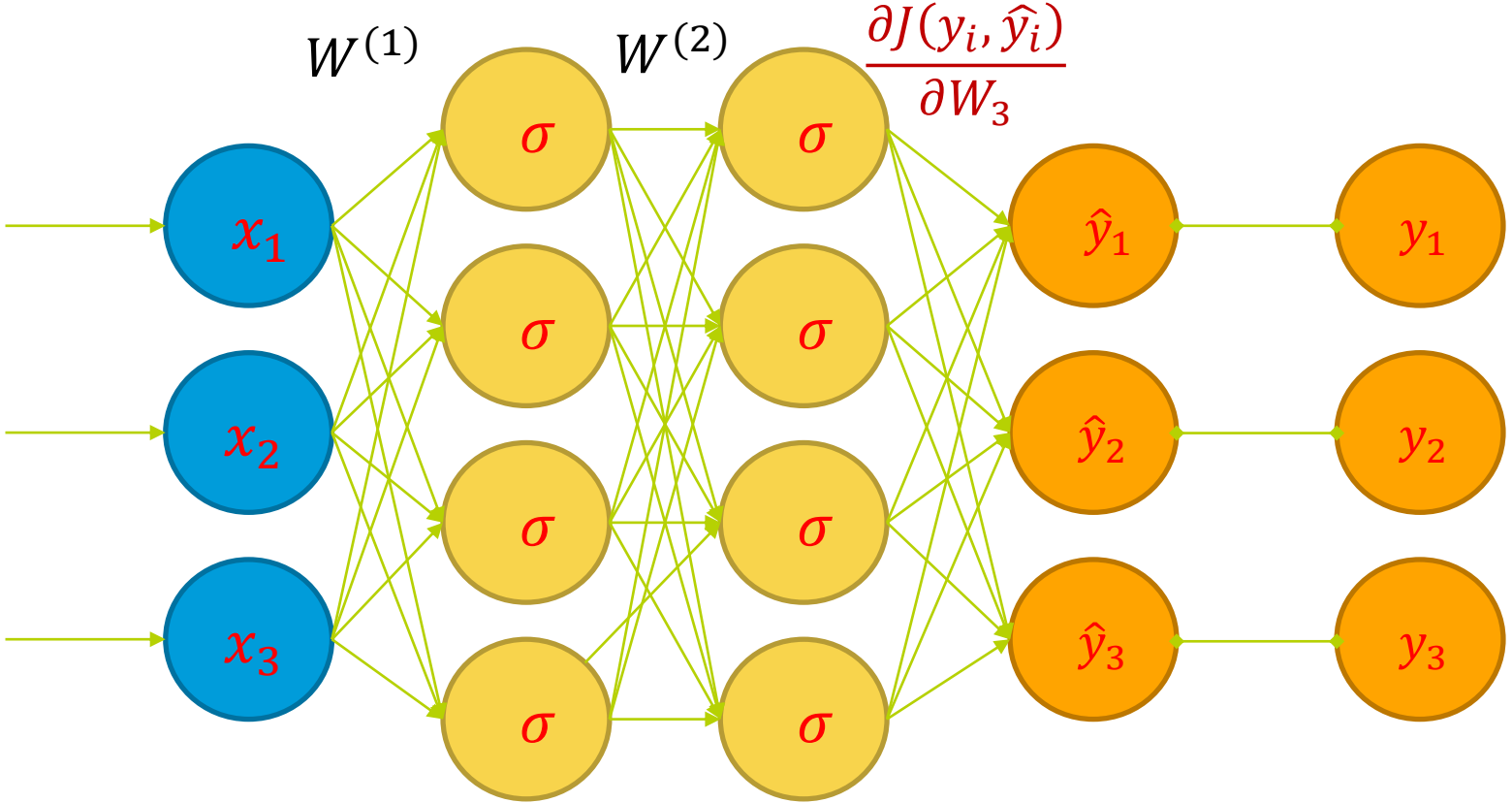
$$\frac{\partial J}{\partial W^{(1)}} = (\hat{y} - y) \cdot W^{(3)} \cdot \sigma'\left(z^{(3)}\right) \cdot W^{(2)} \cdot \sigma'\left(z^{(2)}\right) \cdot X$$

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

Backpropagation

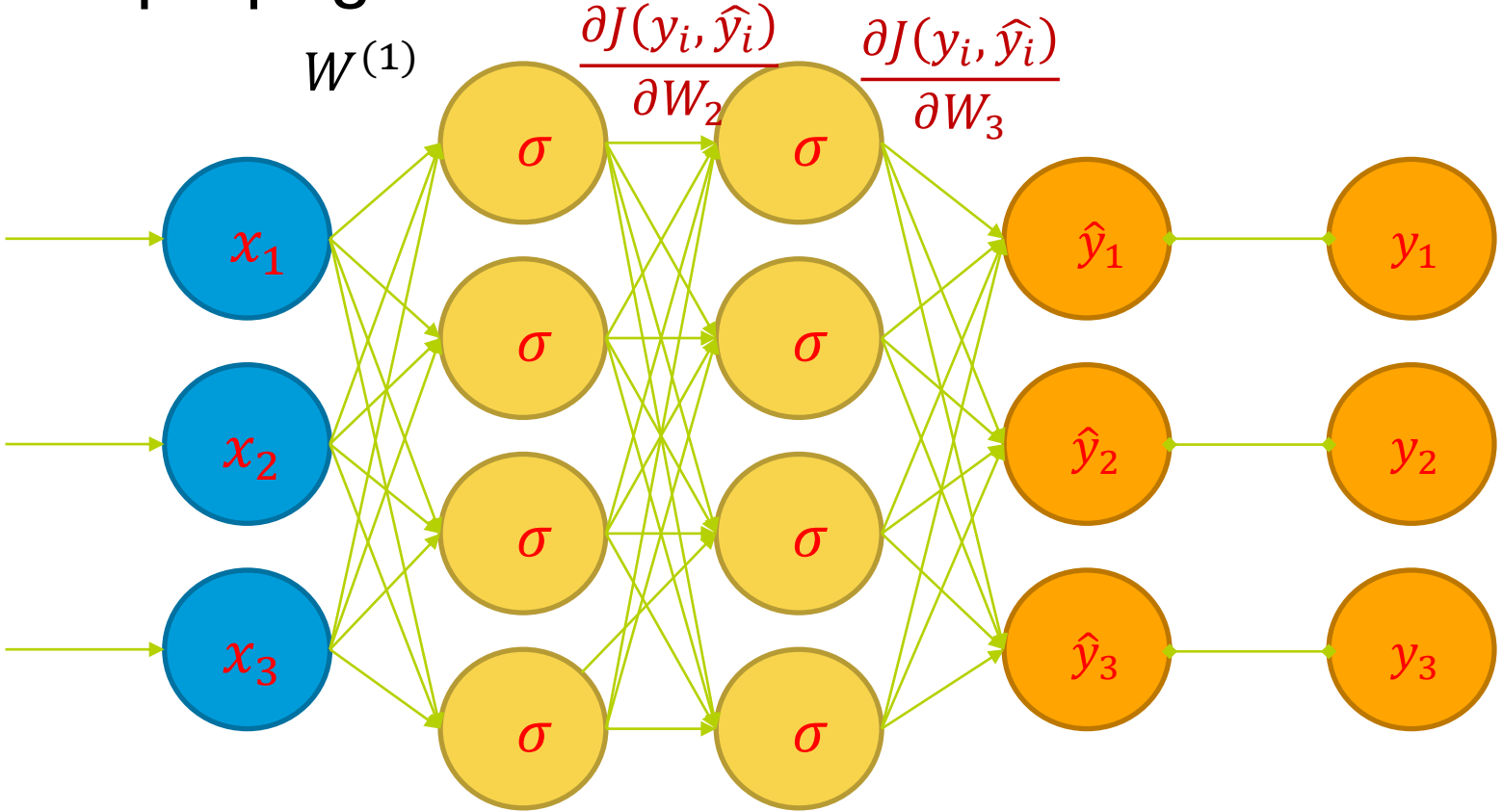


Backpropagation



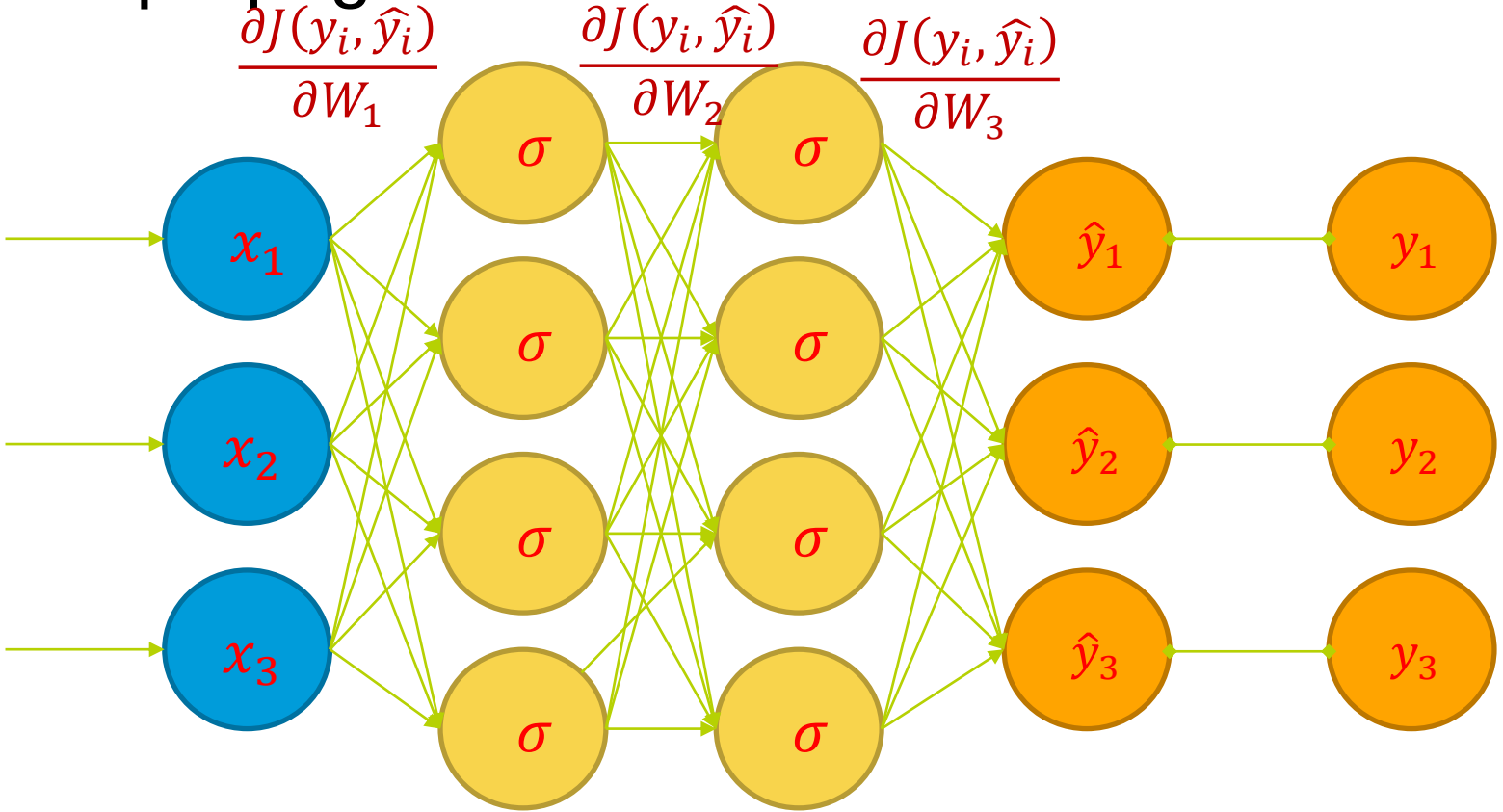
intel Software

Backpropagation



intel Software

Backpropagation



intel Software

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:

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

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

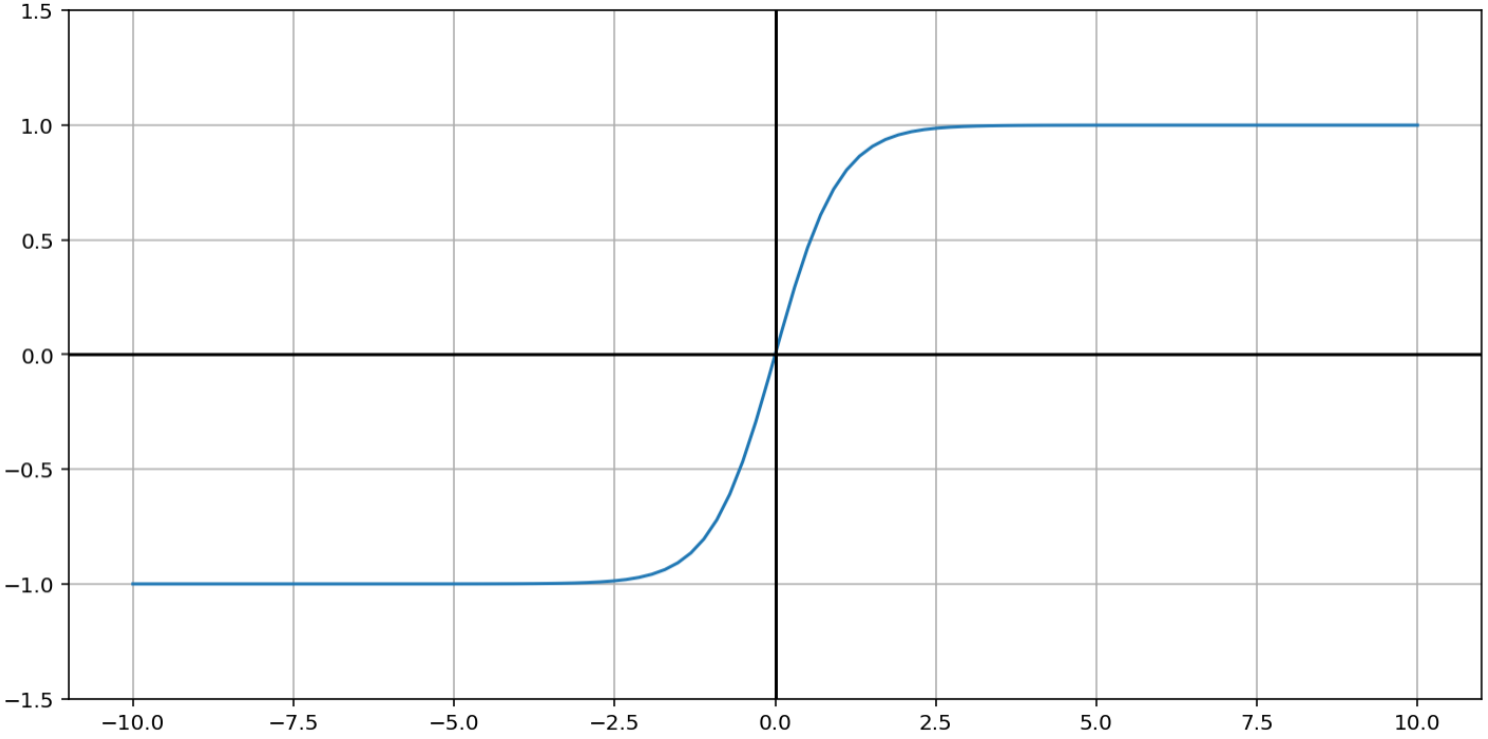
Hyperbolic Tangent Function

- 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



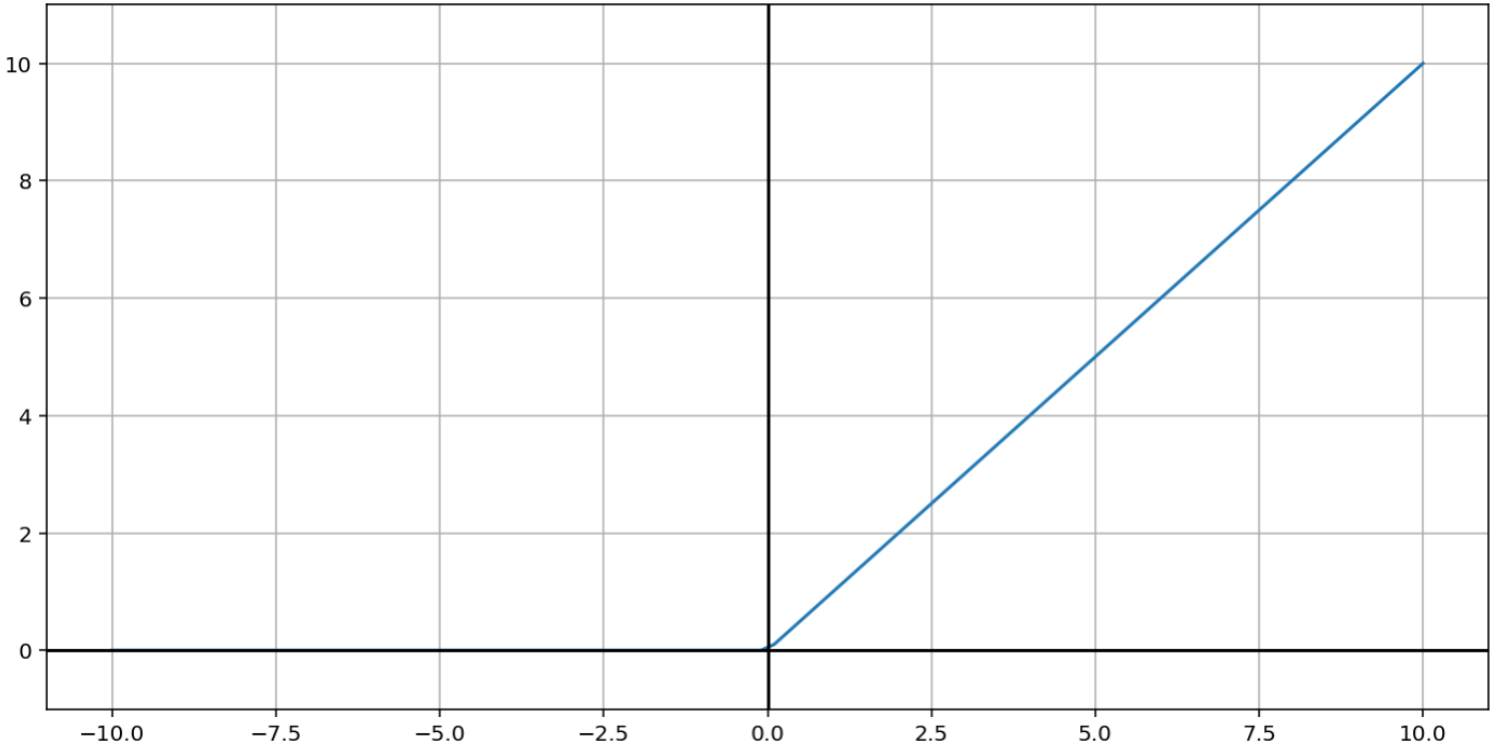
Rectified Linear Unit (ReLU)

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

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

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

Rectified Linear Unit (ReLU)



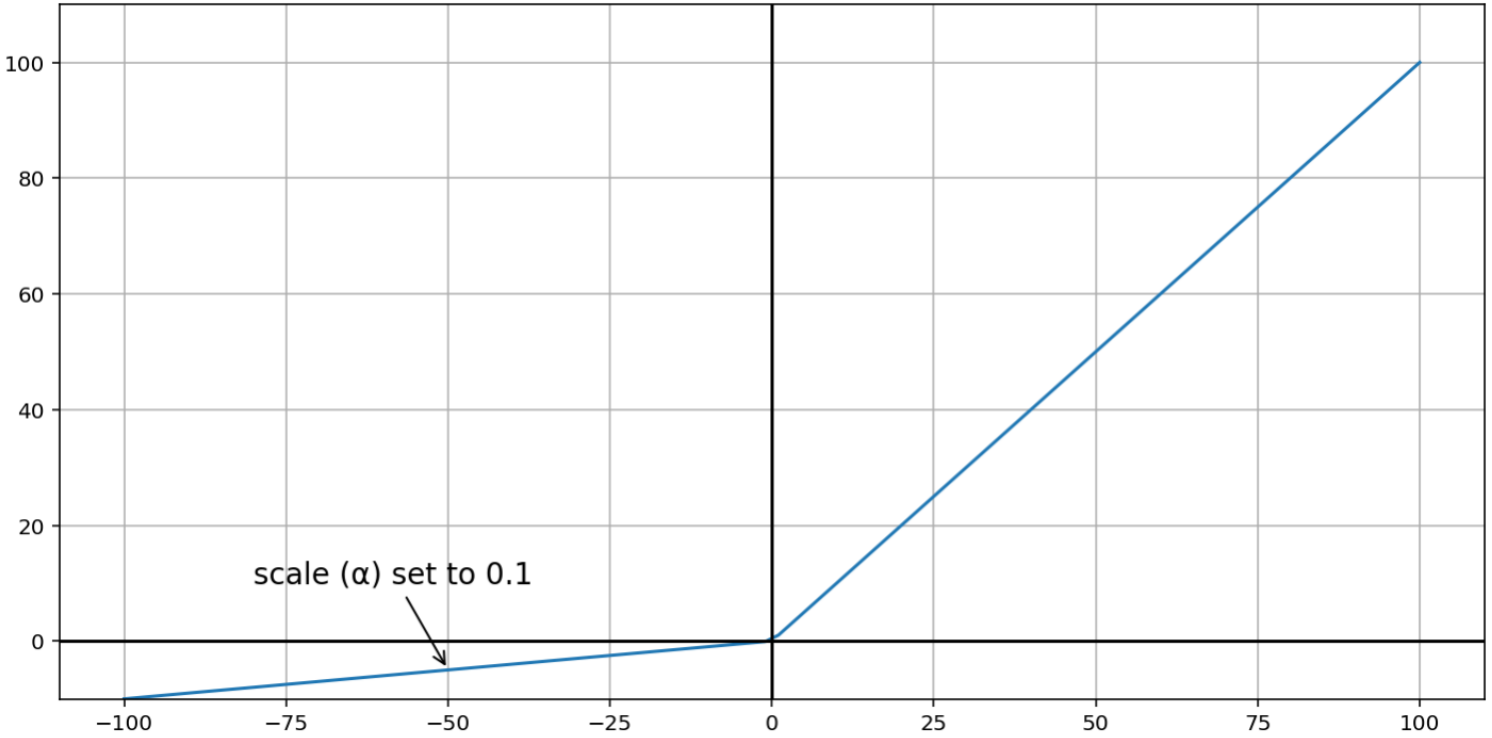
“Leaky” Rectified Linear Unit (ReLU)

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

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

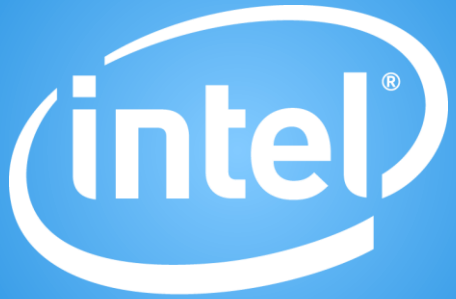
$$LReLU(0) = 0$$
$$LReLU(z) = z \qquad \text{for } (z \gg 0)$$
$$LReLU(-z) = -\alpha z$$

“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.


intel
Software