Scalable Data Mining

Sourangshu Bhattacharya
Dept. of Computer Science and Engineering,
IIT Kharagpur.

Introduction to Machine Learning

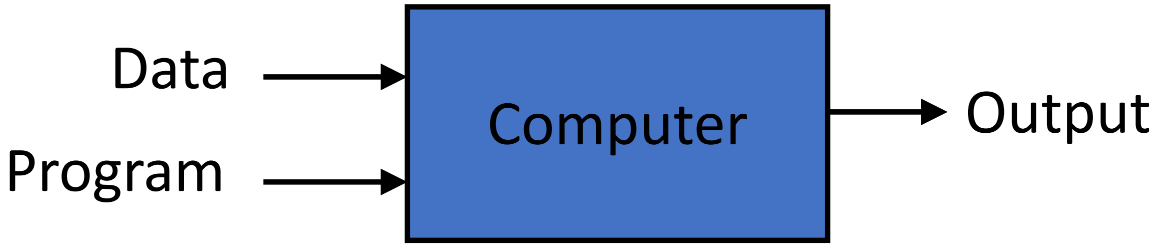
Algorithms

- An algorithm is an unambiguous specification of how to solve a class of problems.
- Example: Euclid’s algorithm for finding the greatest common divisor.
- Important Aspects:
  - Analysis
  - Design

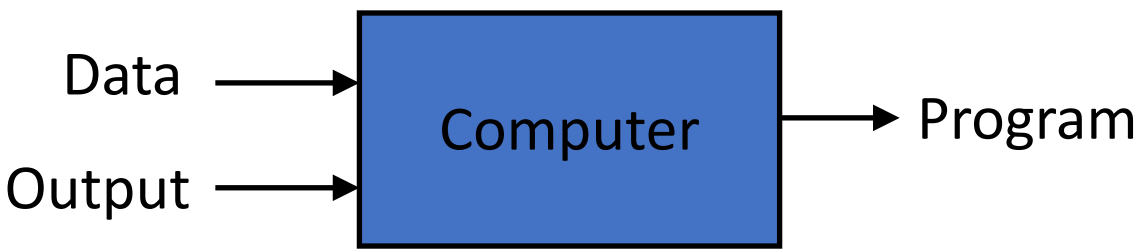
Machine Learning

- Machine learning is a field of computer science that gives computers the ability to learn [from data] without being explicitly programmed.
- Example: Bayesian classifier for automatically filtering email spams.
- Aspects:
  - Modeling
  - Inference and learning

Traditional Programming



Machine Learning



Magic?

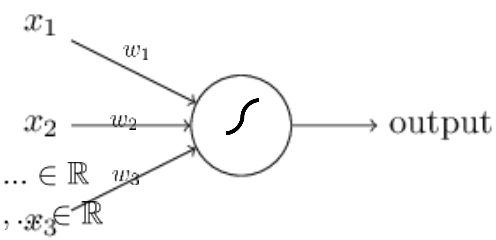
No, more like gardening

- Seeds = Algorithms
- Nutrients = Data
- Gardener = You
- Plants = Programs

Neural Network Basics

- Given several **inputs**:
  and several **weights**:
  and a **bias** value:

$x_1, x_2, x_3, \ldots \in \mathbb{R}$
$w_1, w_2, w_3, \ldots \in \mathbb{R}$
$b \in \mathbb{R}$



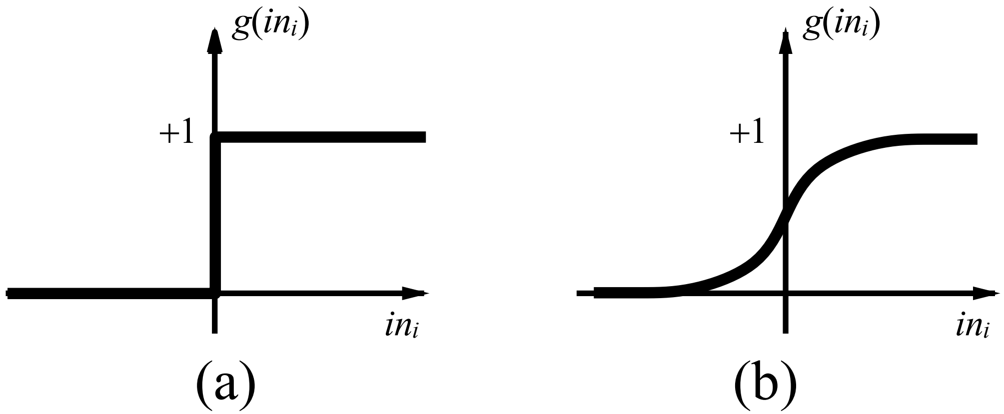
- A neuron produces a single output:

$$o_1 = s(\sum_i w_i x_i + b)$$

$$\sum_i w_i x_i + b$$

- This sum is called the **activation** of the neuron
- The function *s* is called the **activation function** for the neuron
- The weights and bias values are typically initialized randomly and learned during training

Activation functions

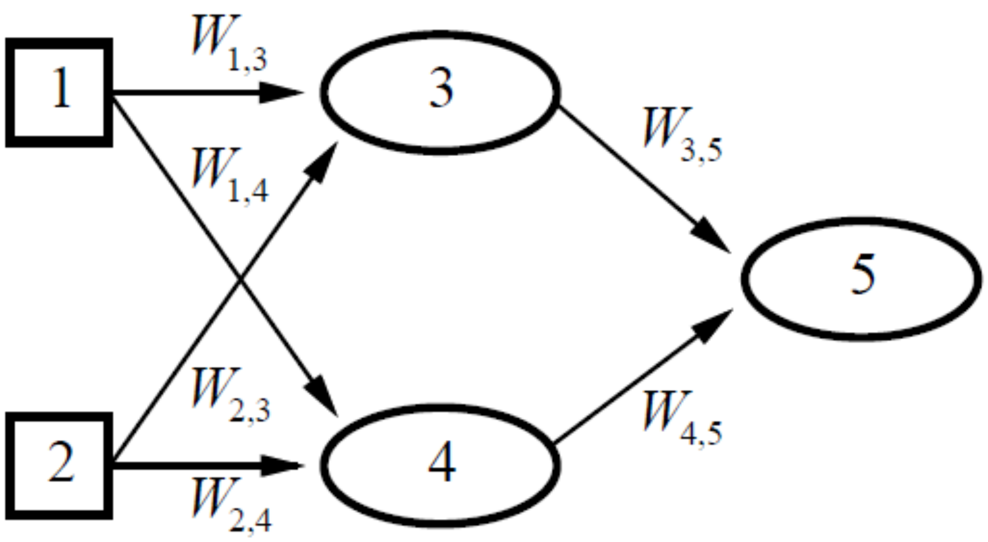


(a) is a step function or threshold function

(b) is a sigmoid function $1/(1+e^{-x})$

Changing the bias weight $W_{0,i}$ moves the threshold location

Feed forward example



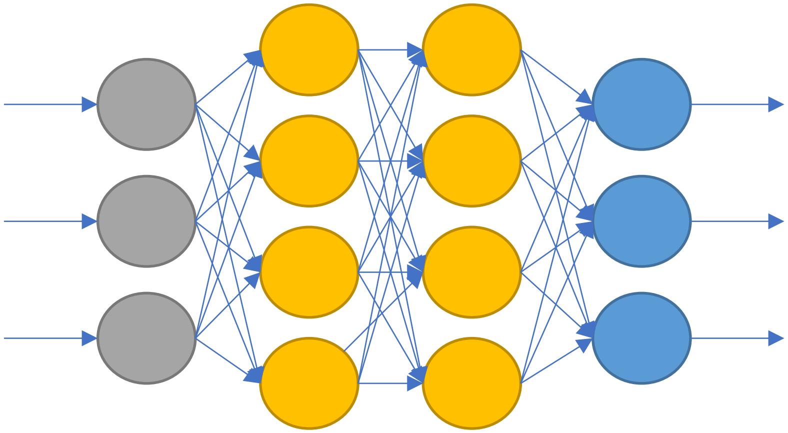
Feed-forward network = a parameterized family of nonlinear functions:

$$\begin{aligned} a_5 &= g(W_{3,5} \cdot a_3 + W_{4,5} \cdot a_4) \\ &= g(W_{3,5} \cdot g(W_{1,3} \cdot a_1 + W_{2,3} \cdot a_2) + W_{4,5} \cdot g(W_{1,4} \cdot a_1 + W_{2,4} \cdot a_2)) \end{aligned}$$

Adjusting weights changes the function: do learning this way!

How to Train a Neural Net?

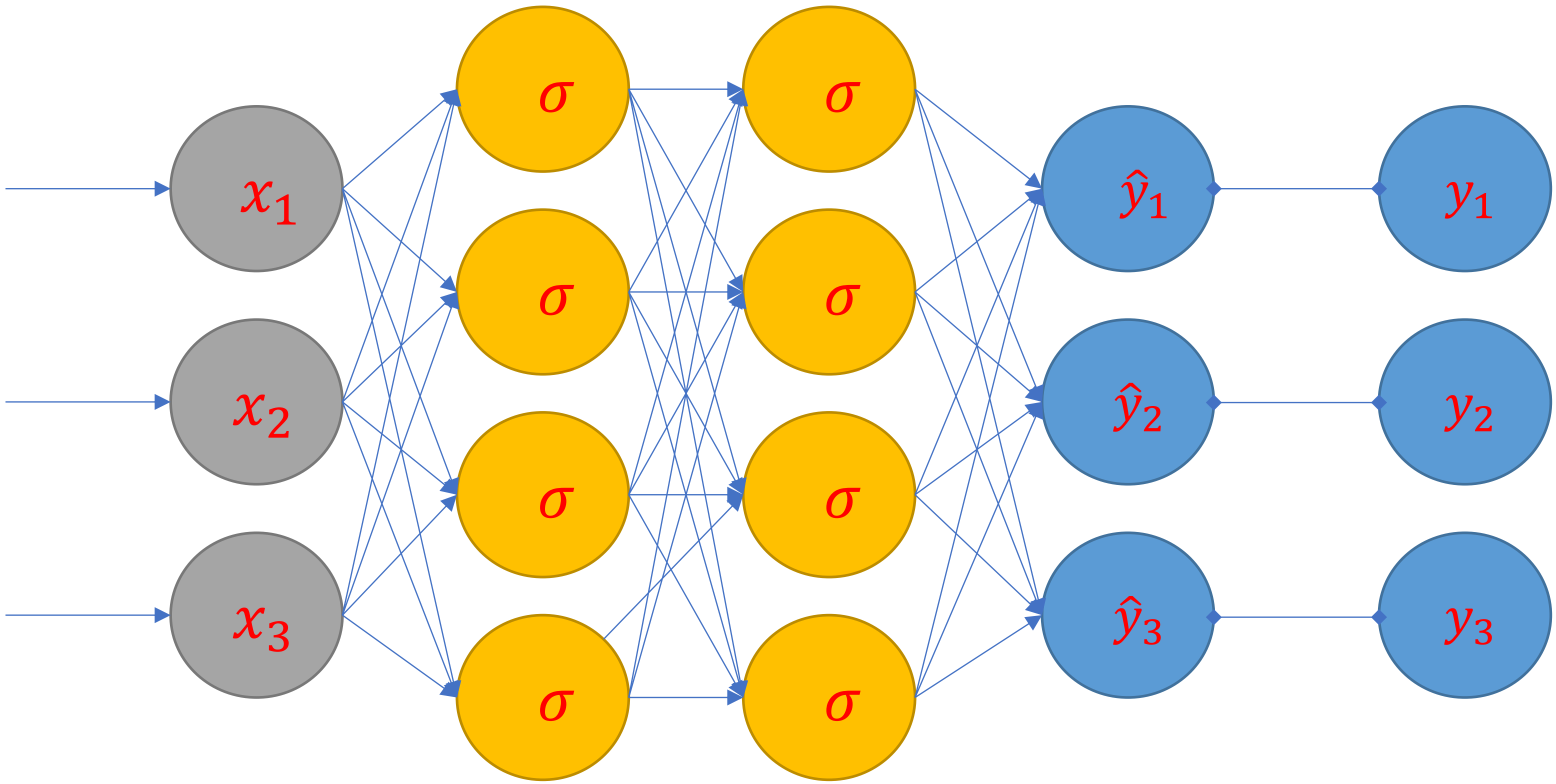
Input
(Feature Vector)



Output
(Label)

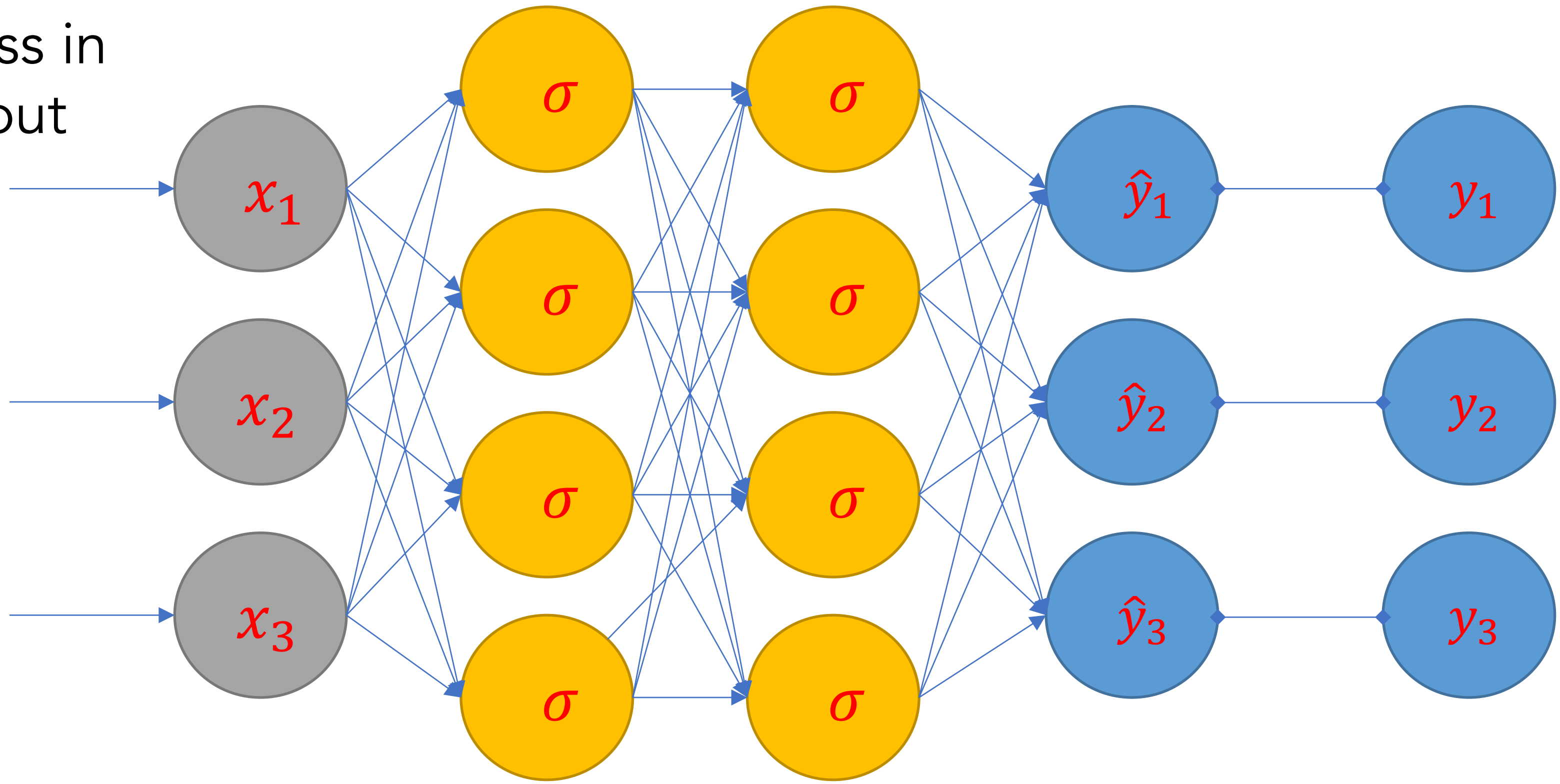
- Put in Training inputs, get the output
- Compare output to correct answers: Look at loss function J
- Adjust and repeat!
- Backpropagation tells us how to make a single adjustment using calculus.

Feedforward Neural Network

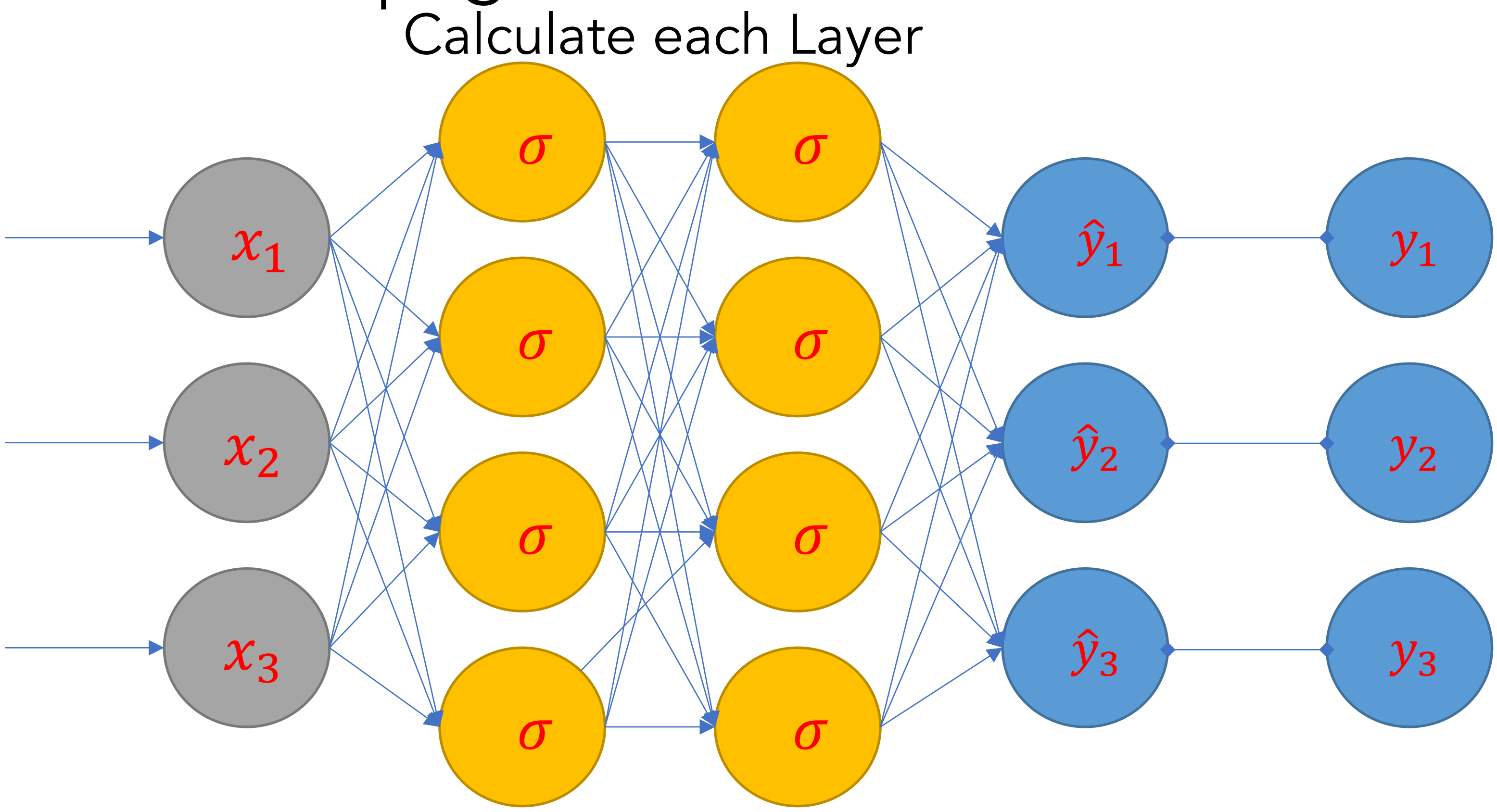


Forward Propagation

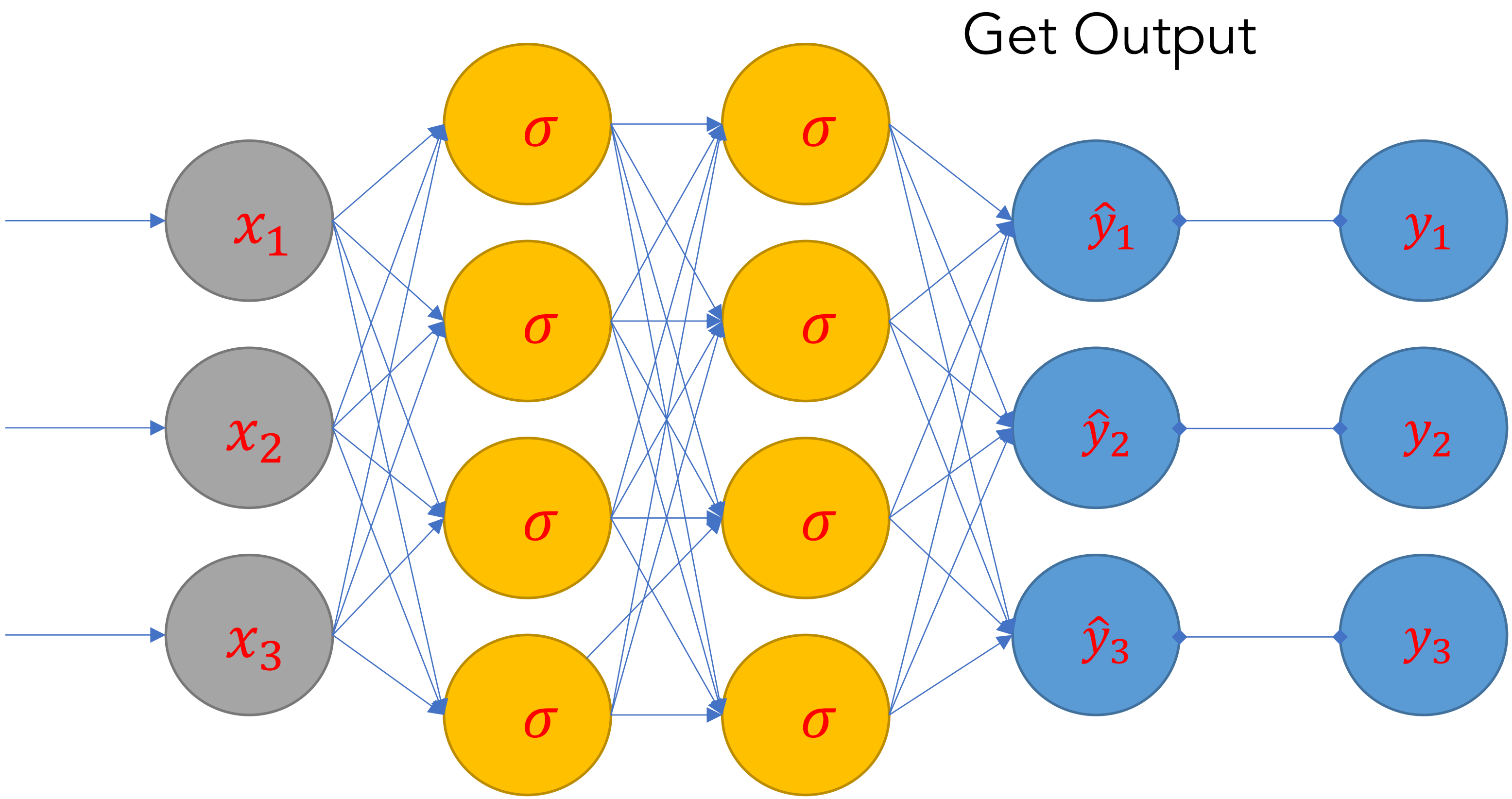
Pass in
Input



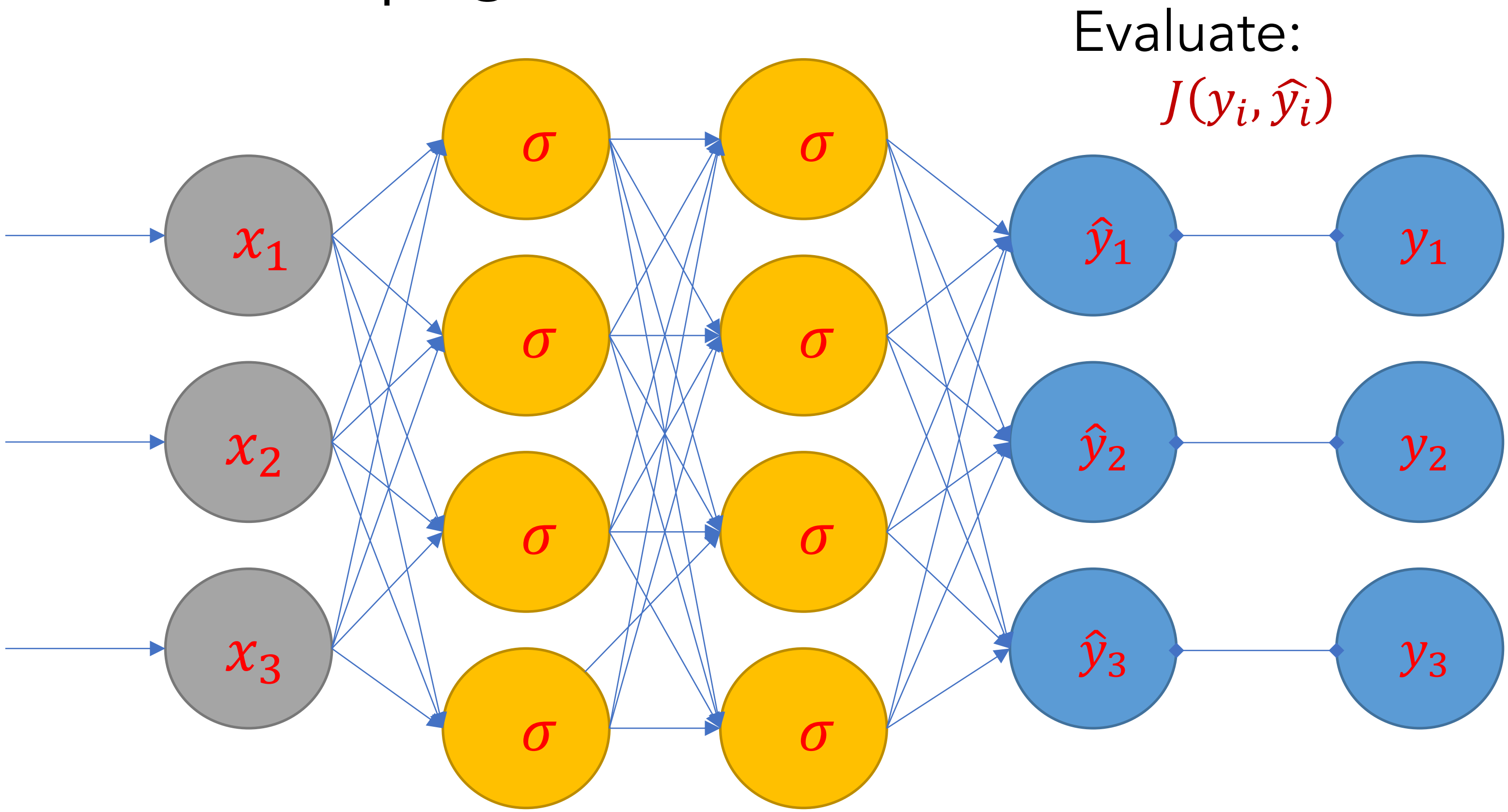
Forward Propagation



Forward Propagation



Forward Propagation



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

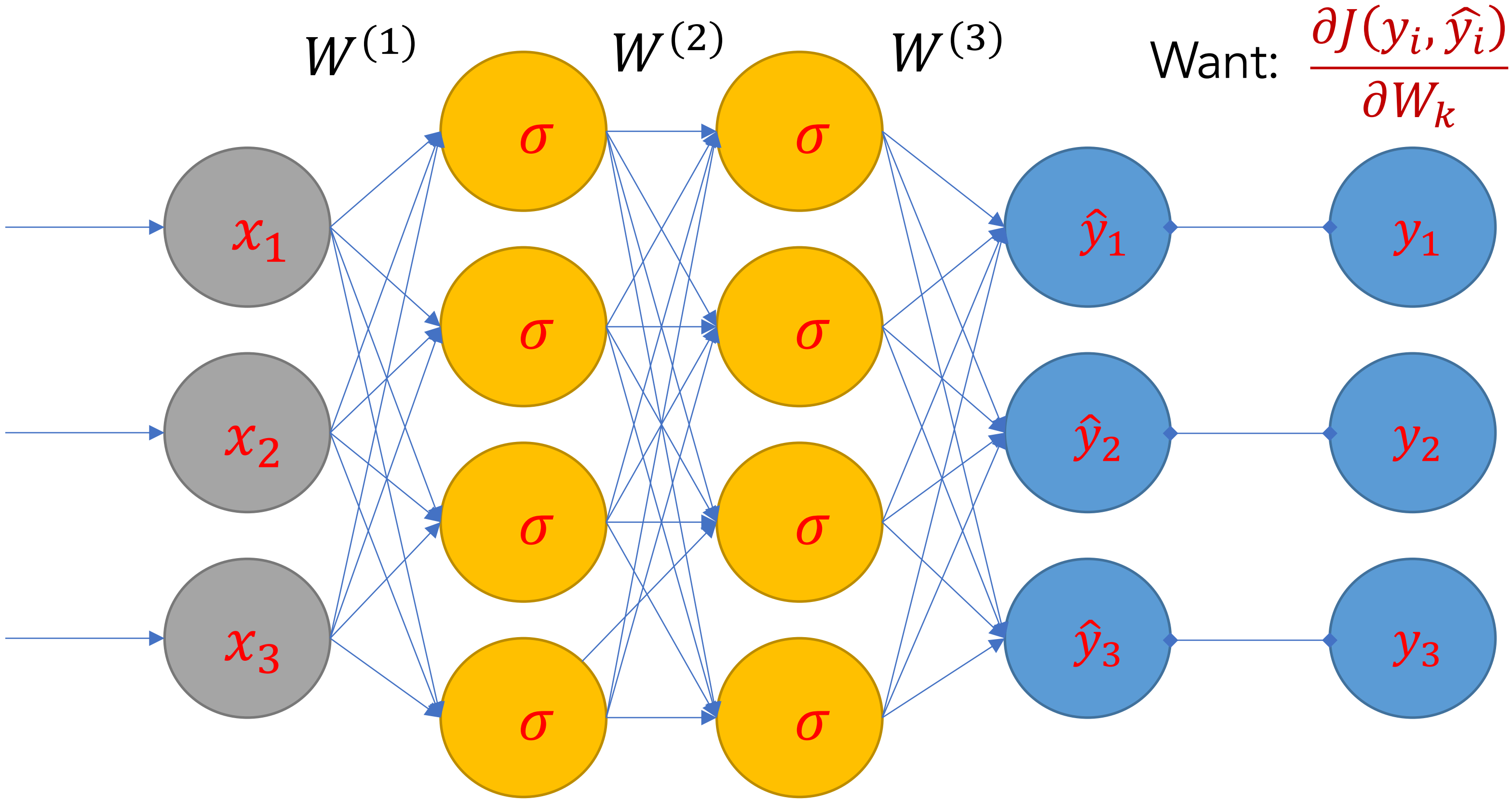
How to Train a Neural Net?

- 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

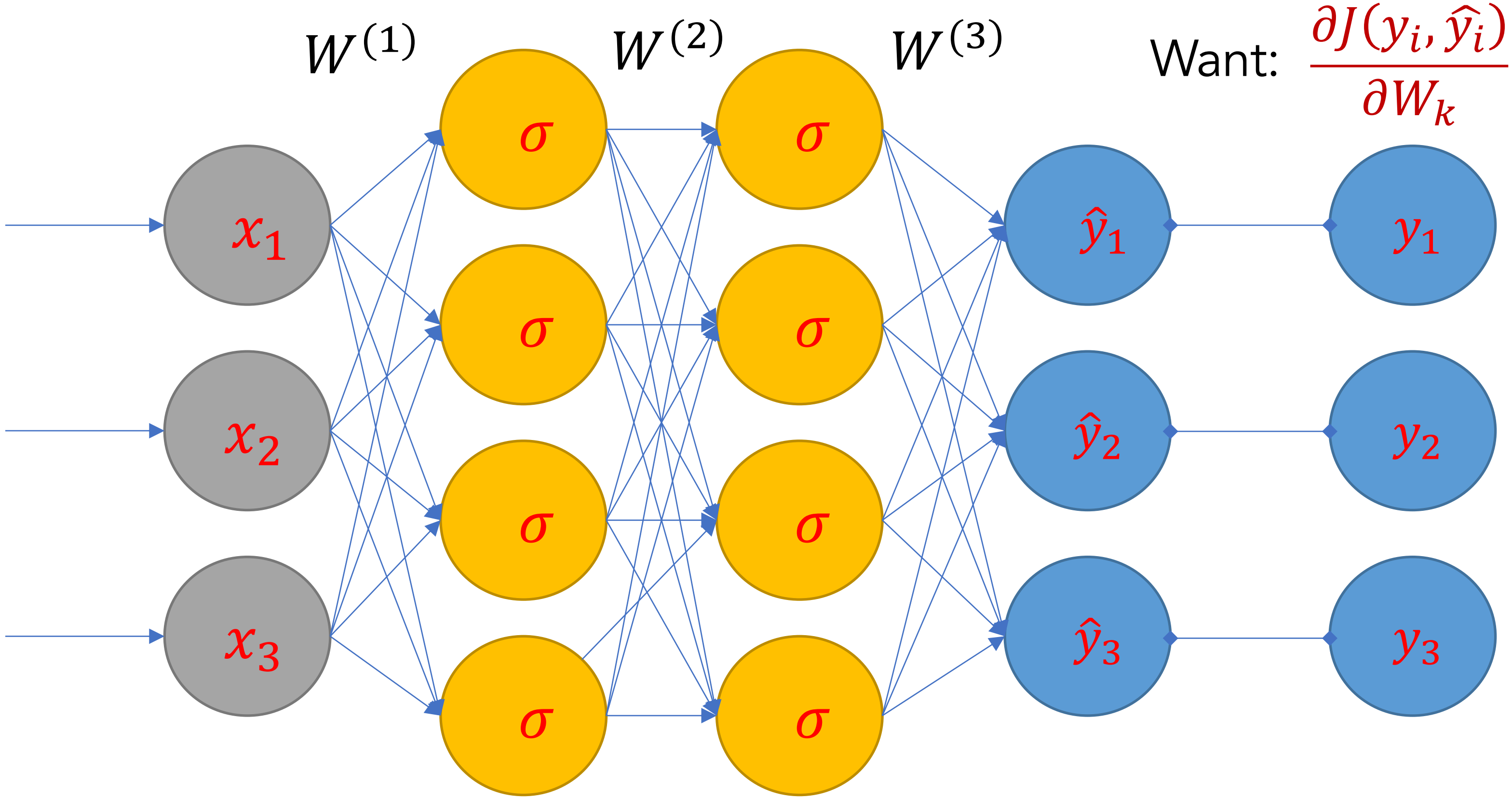


Calculus to the Rescue

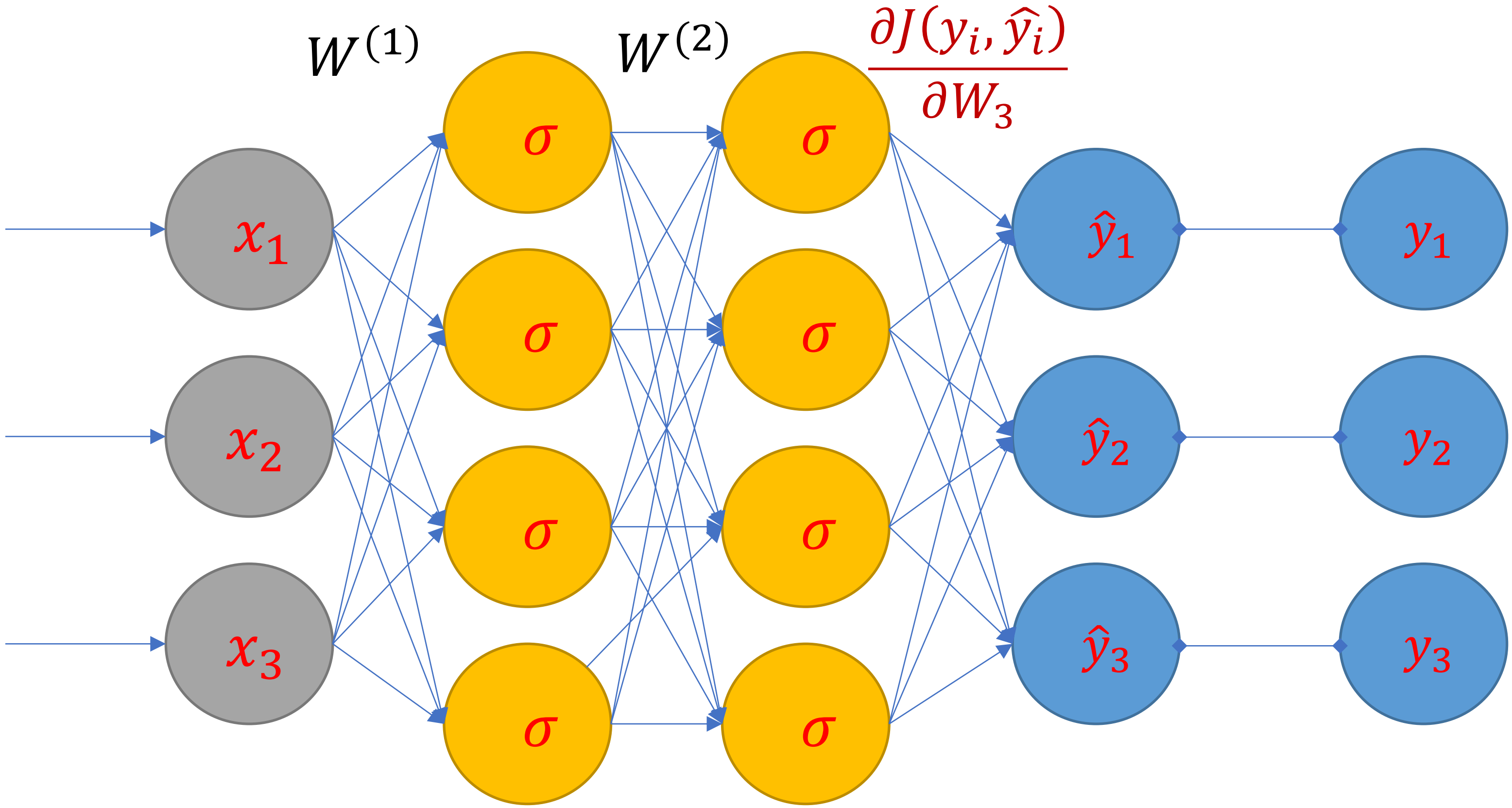
- Use calculus, chain rule, etc. etc.
- Functions are chosen to have “nice” derivatives
- Numerical issues to be considered

Backpropagation

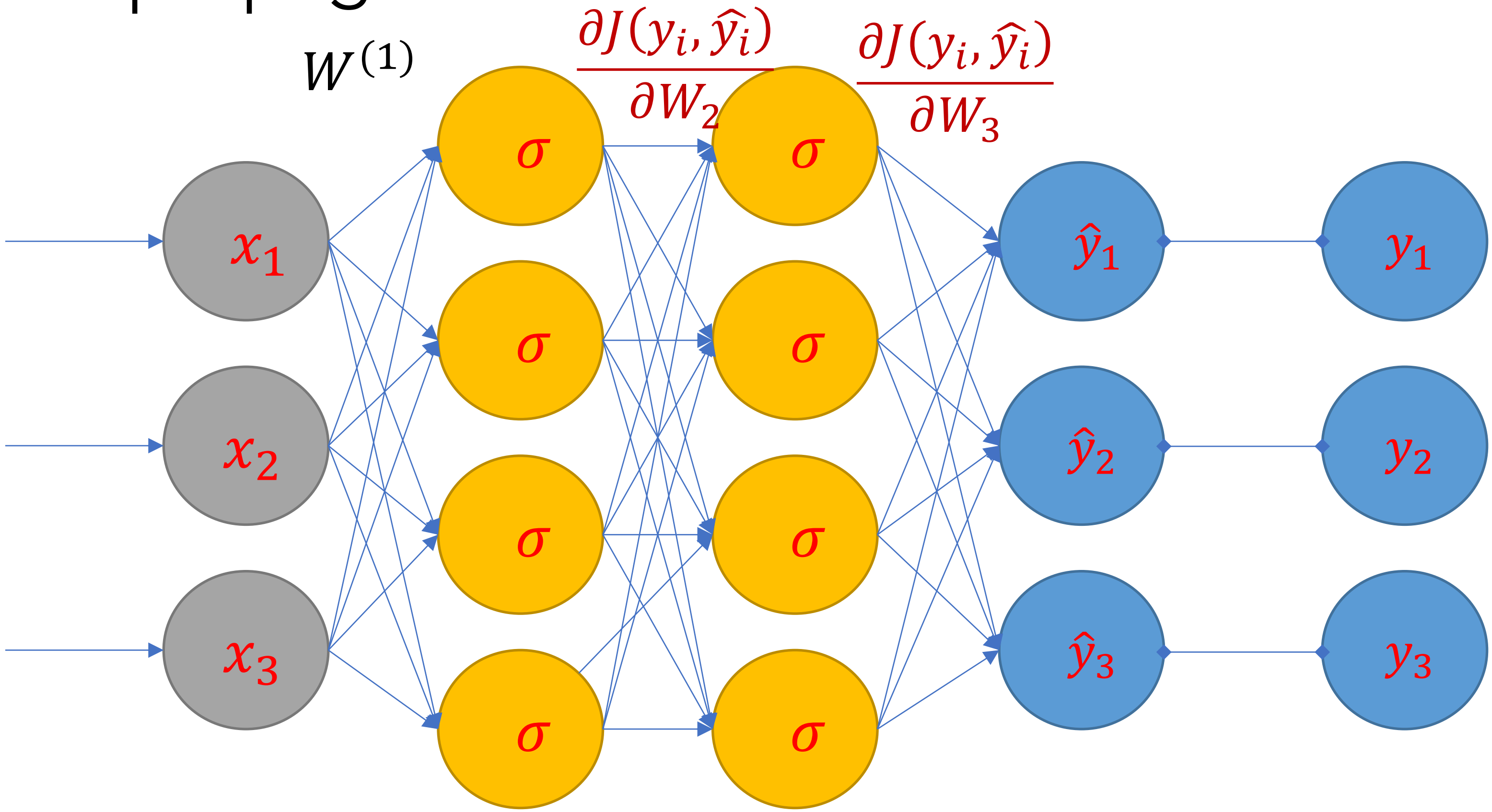
Want: $\frac{\partial J(y_i, \hat{y_i})}{\partial W_k}$



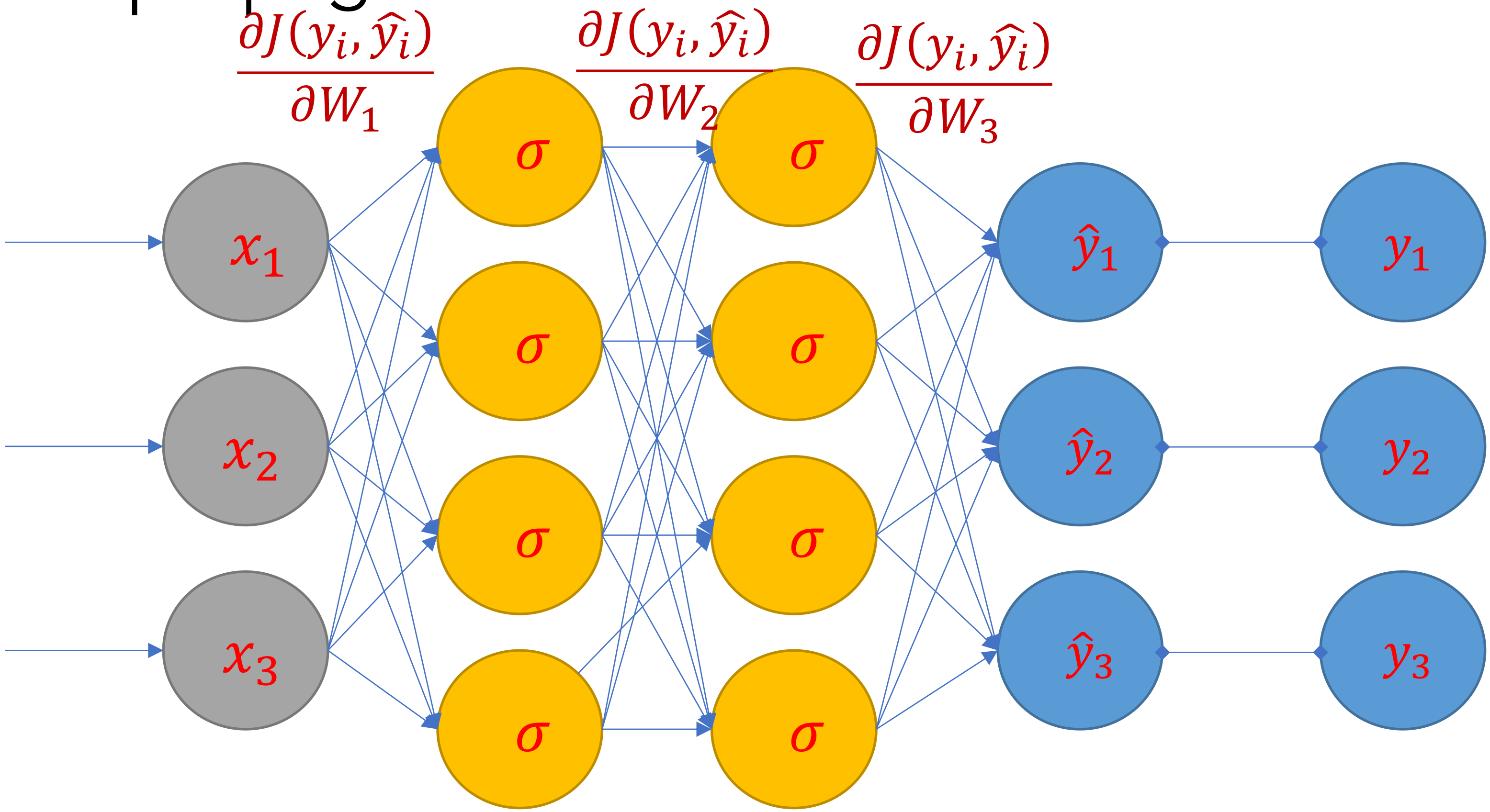
Backpropagation



Backpropagation



Backpropagation



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!

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