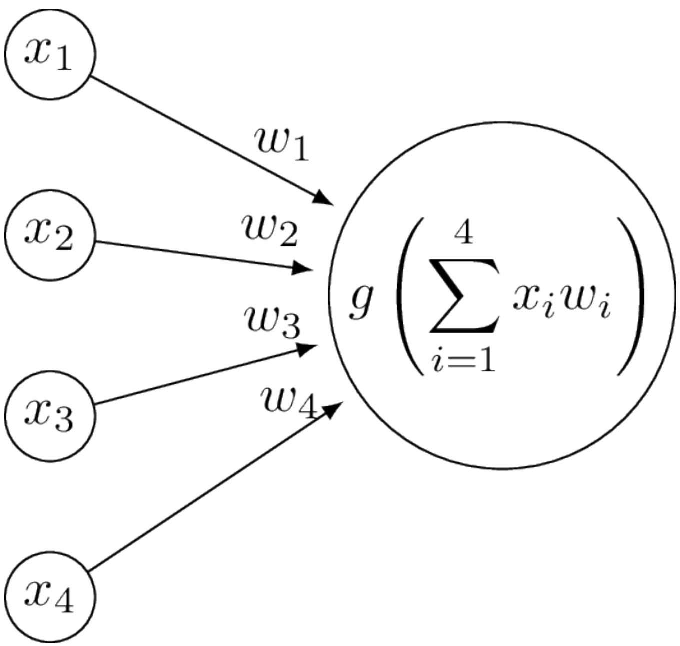
# Multi-Layer Neural Networks

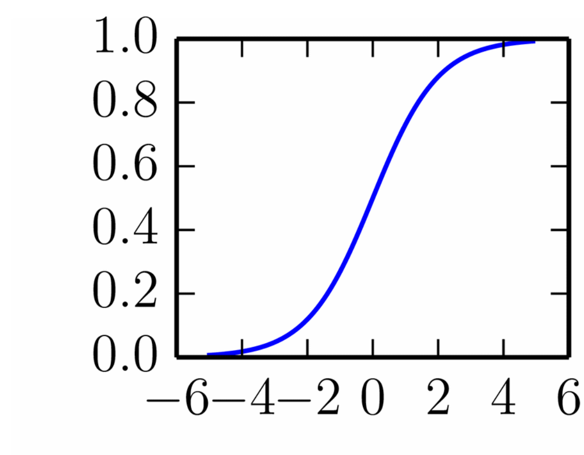
# Review

## Neuron

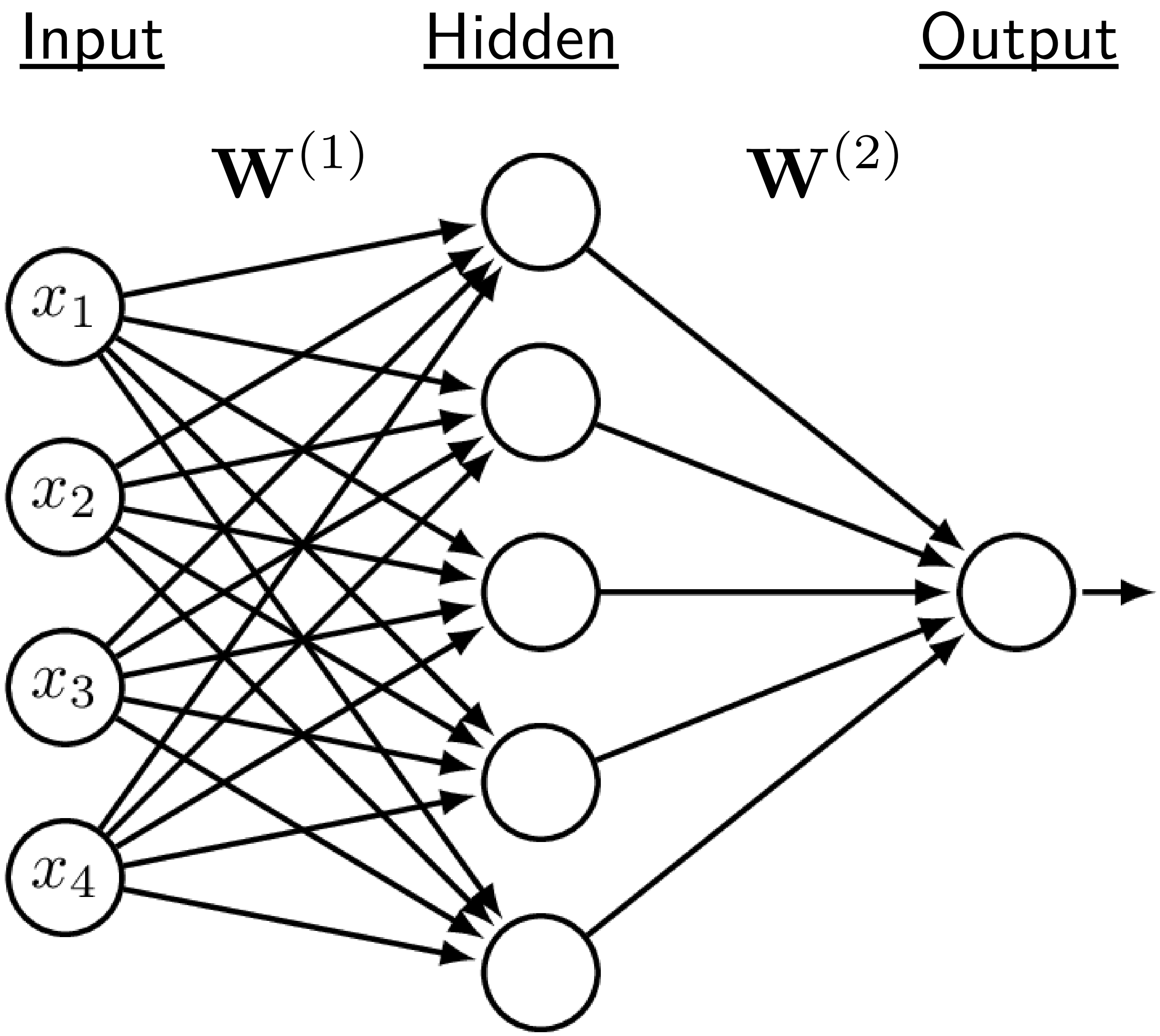



## Non-linearity

$$g(a) = \frac{1}{1+e^{-a}}$$



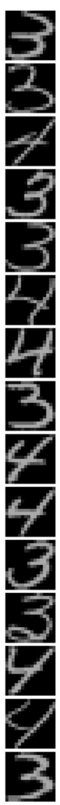
# Multi-Layer Networks



# Neural Network Example

Training Data

$\mathbf{x}$ $y$

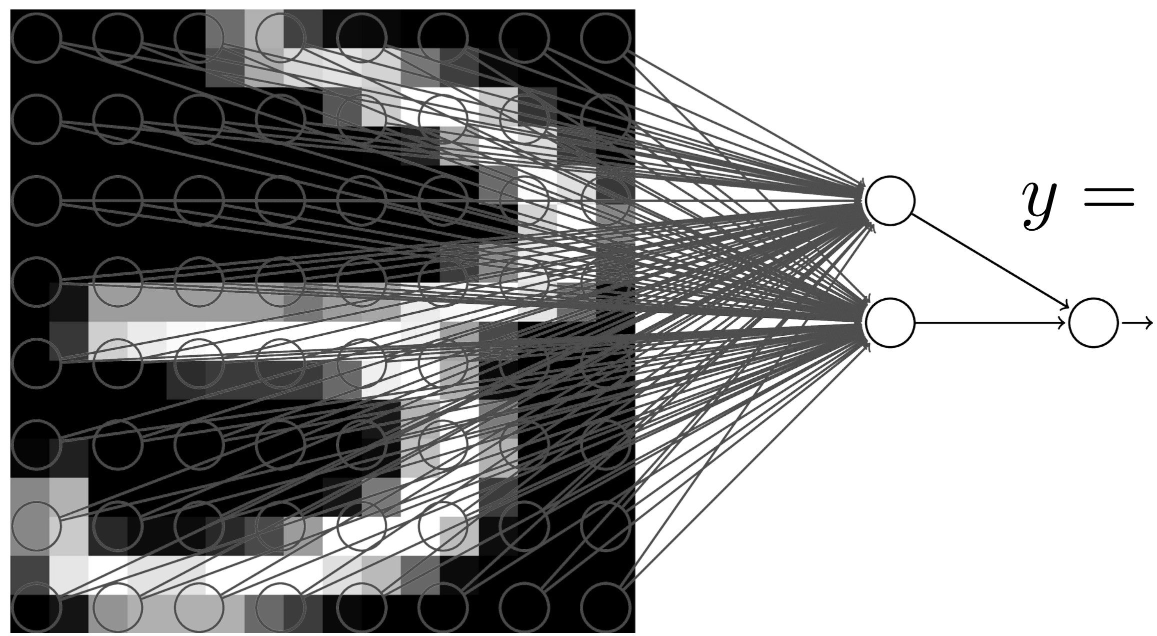


$\rightarrow 1$
$\rightarrow 1$
$\rightarrow 0$
$\rightarrow 1$
$\rightarrow 1$
$\rightarrow 0$
$\rightarrow 0$
$\rightarrow 1$
$\rightarrow 0$
$\rightarrow 0$
$\rightarrow 1$
$\rightarrow 1$
$\rightarrow 0$
$\rightarrow 0$
$\rightarrow 1$

$\vdots$

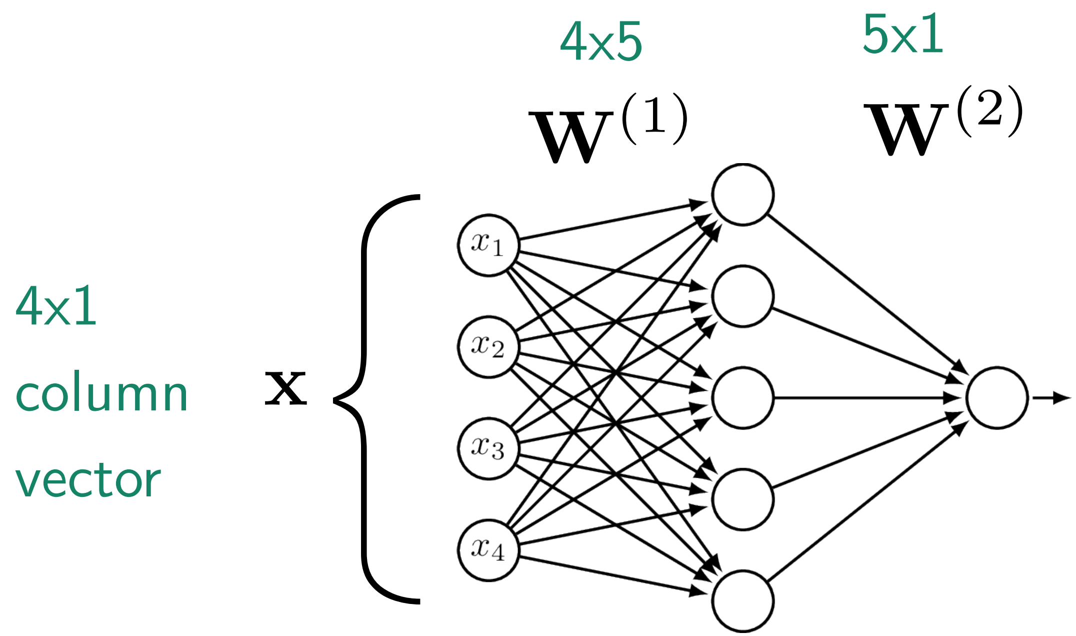
Network

$\mathbf{x}$



$y = 1$

# Computation Example

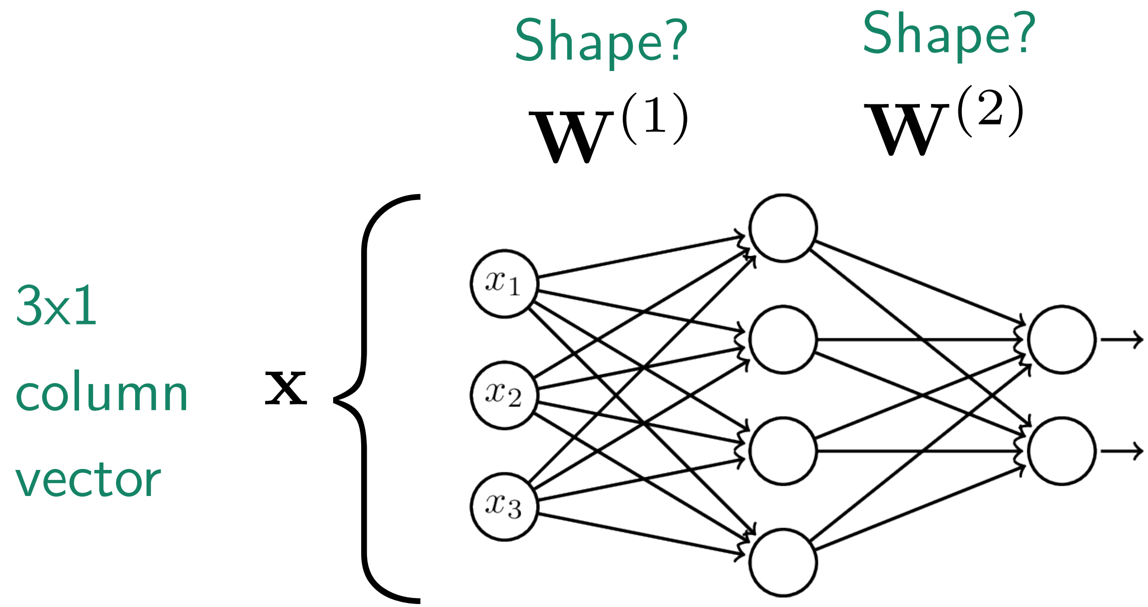



Hidden activation: $h(\mathbf{x}^\mathsf{T} W^{(1)})$

Output activation: $\sigma\left(h(\mathbf{x}^\mathsf{T} W^{(1)}) W^{(2)}\right)$

*(h* is the non-linearity at the hidden layer. *σ* is the non-linearity at the output. Applied element-wise.)

# QUIZ




Hidden activation: $h(\mathbf{x}^\mathsf{T} W^{(1)})$

Output activation: $\sigma\left(h(\mathbf{x}^\mathsf{T} W^{(1)}) W^{(2)}\right)$

*(h* is the non-linearity at the hidden layer. $\sigma$ is the non-linearity at the output. Applied element-wise.)

# Backpropagation

- Activation at the output layer:

$$a_k = \sigma\left(\sum_j w_{j,k}^{(2)}\, h\left(\sum_i w_{i,j}^{(1)} x_i\right)\right)$$

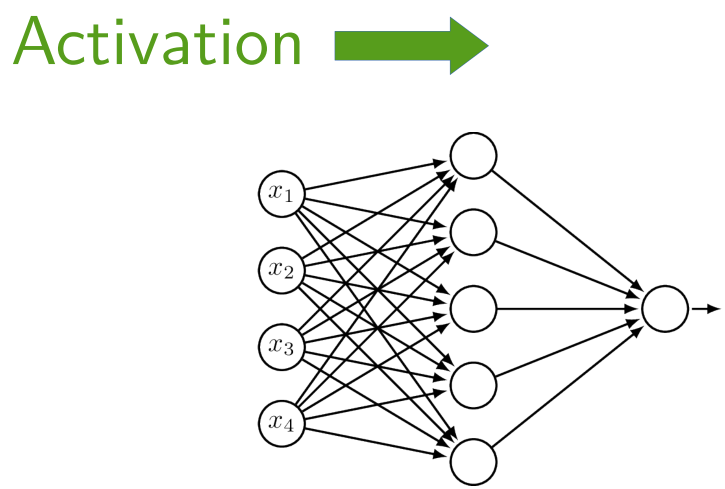
- Here σ is the activation function at the output layer. Units at the input layer are indexed with *i*, hidden with *j* and output with *k.*
- Error metric, assuming multiple output units:

$$Error = \frac{1}{k}\sum_k (y_k - a_k)^2$$

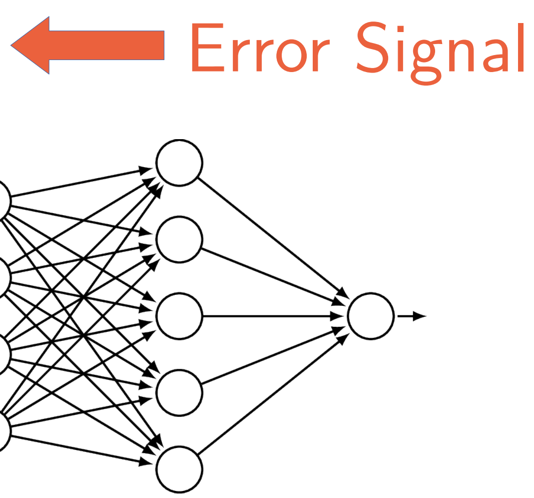
- Now just compute $\frac{\partial Error}{\partial w_{j,k}^{(2)}}$ and $\frac{\partial Error}{\partial w_{i,j}^{(1)}}$ .

# Backpropagation Algorithm

- Forward Pass:




- Backward Pass:



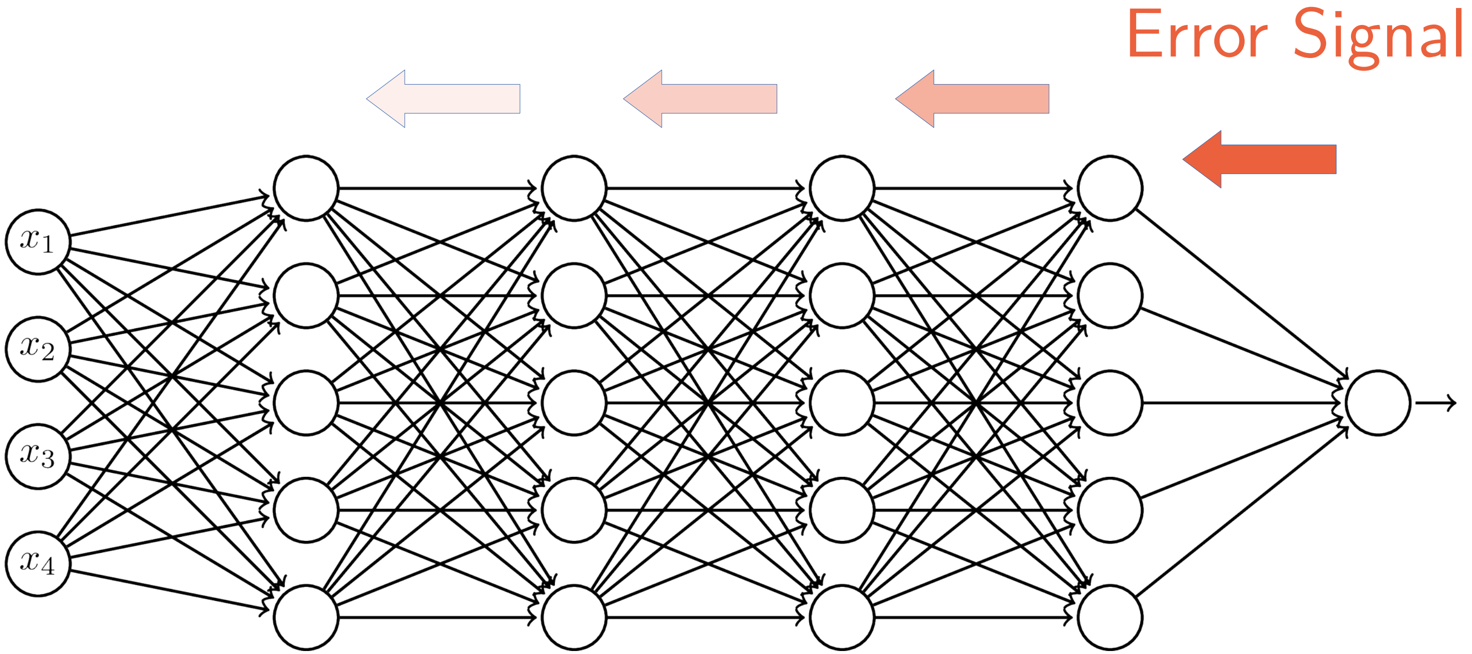
# Backpropagation: Some Good News

- Calculating partial derivatives is tedious, but mechanical
- Modern neural network libraries perform automatic differentiation
  - Tensorflow
  - PyTorch
  - Etc.
- The programmer just needs to specify the network structure and the loss function – No need to explicitly write code for performing weight updates
- The computational cost for the backward pass is not much more than the cost for the forward pass

# Deep vs. Shallow Networks

- How best to add capacity?
  - More units in a single hidden layer?
    - Three layer networks are universal approximators: with enough units any continuous function can be approximated
    - Adding layers makes the learning problem harder...

# Vanishing Gradients



# Advantages of Deep Architectures

- There are tasks that require exponentially many hidden units for a three-layer architecture, but only polynomially many with more hidden layers
- The best hand-coded image processing algorithms have deep structure
- The brain has a deep architecture
- MORE SOON.