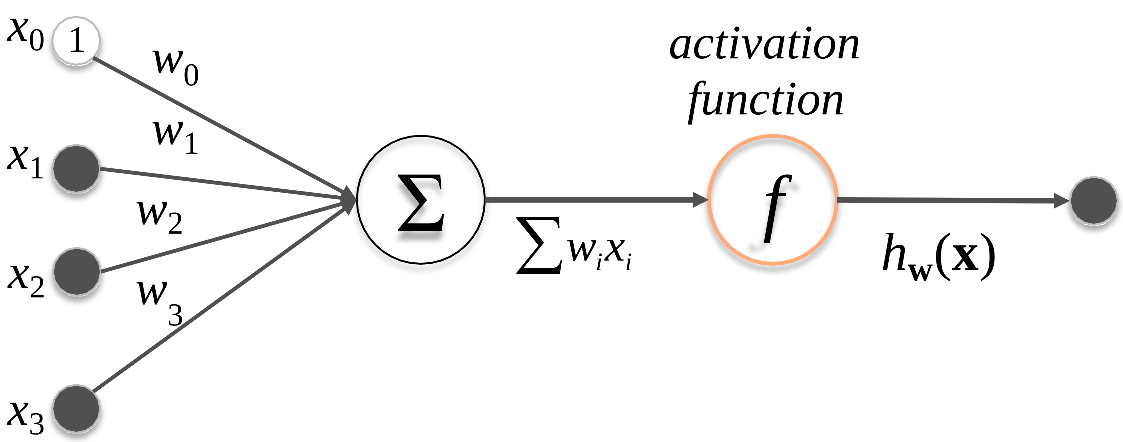
Deep Learning: Backpropagation

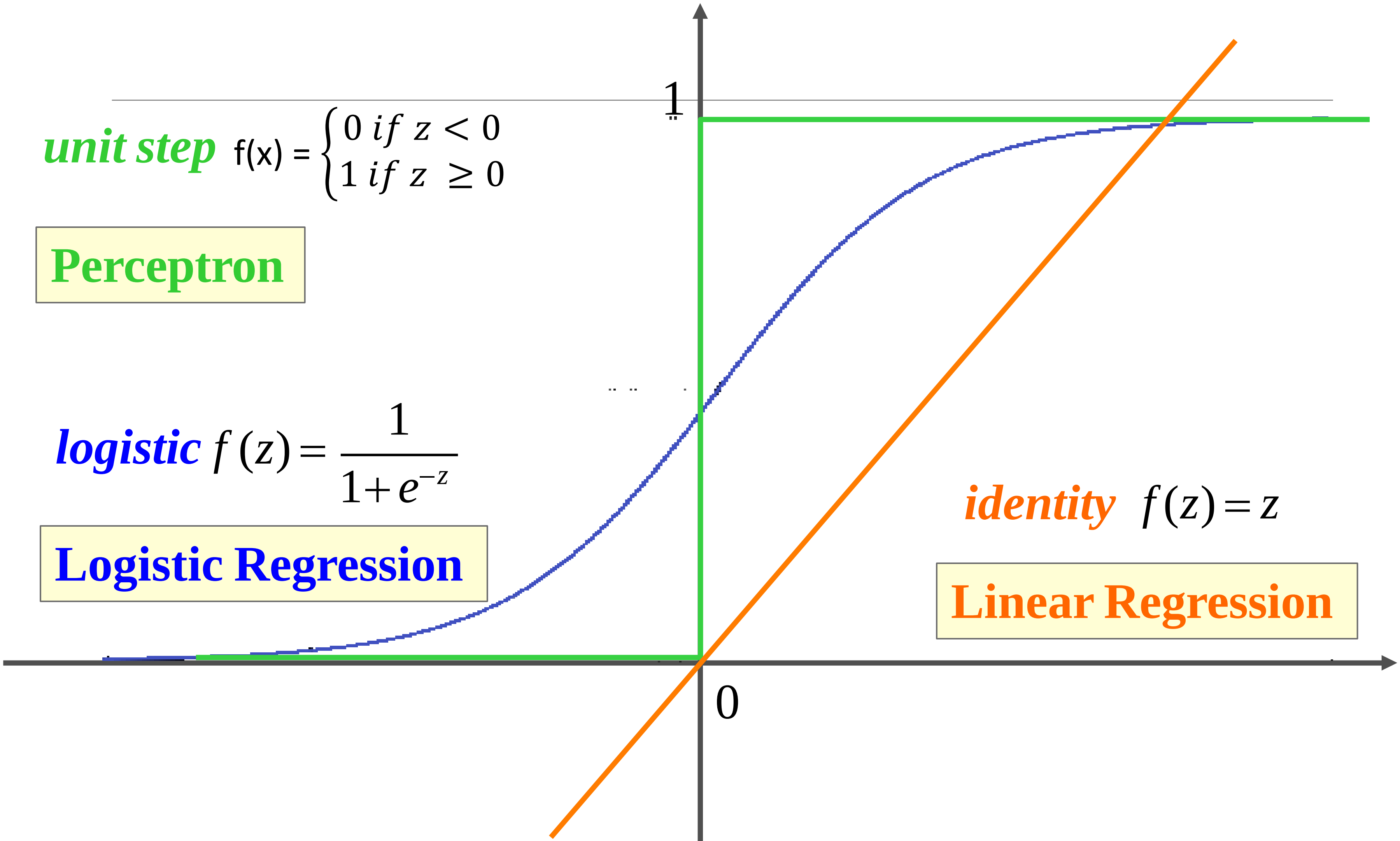
Lecture 03

Neuron Function



- Algebraic interpretation:
  - The output of the neuron is a **linear combination** of inputs from other neurons, **rescaled by** the synaptic **weights**.
    - weights w_i correspond to the synaptic weights (activating or inhibiting).
    - summation corresponds to combination of signals in the soma.
  - It is often transformed through a monotonic **activation function**.

Activation Functions



Perceptron vs. Logistic Neuron

- We will use the **logistic neuron** for this lecture:
  - At inference time, same decision function as **perceptron**, for binary classification with equal misclassification costs (prove it):

$$\hat{t}(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{w}^T\mathbf{x} > 0 \\ 0 & \text{otherwise} \end{cases}$$

  - **Perceptron** cannot represent the XOR function
  - **Logistic neuron** has the same limitation.

- How can we use (**logistic**) **neurons** to achieve better representational power?

Universal Approximation Theorem

Hornik (1991), Cybenko (1989)

- Let σ be a nonconstant, bounded, and monotonically-increasing continuous function;
- Let I_m denote the m-dimensional unit hypercube $[0,1]^m$;
- Let $C(I_m)$ denote the space of continuous functions on I_m;

➢ **Theorem**: Given any function $f \in C(I_m)$ and $\varepsilon > 0$, there exist an integer N and real constants α_i, $b_i \in R$, $\mathbf{w}_i \in R^m$, where $i = 1, ..., N$, such that:

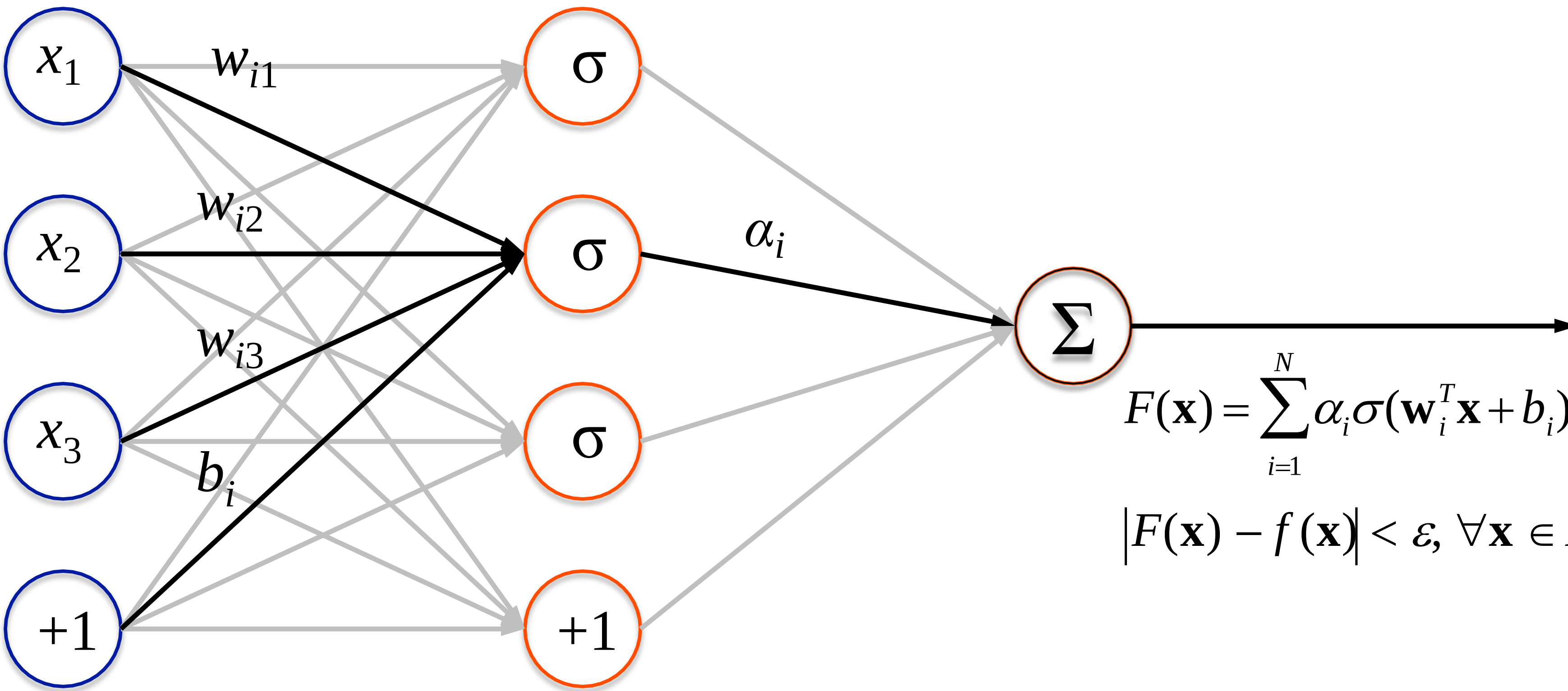
$$\left| F(\mathbf{x}) - f(\mathbf{x}) \right| < \varepsilon, \quad \forall \mathbf{x} \in I_m$$

where

$$F(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i \sigma(\mathbf{w}_i^T \mathbf{x} + b_i)$$

Universal Approximation Theorem

Hornik (1991), Cybenko (1989)

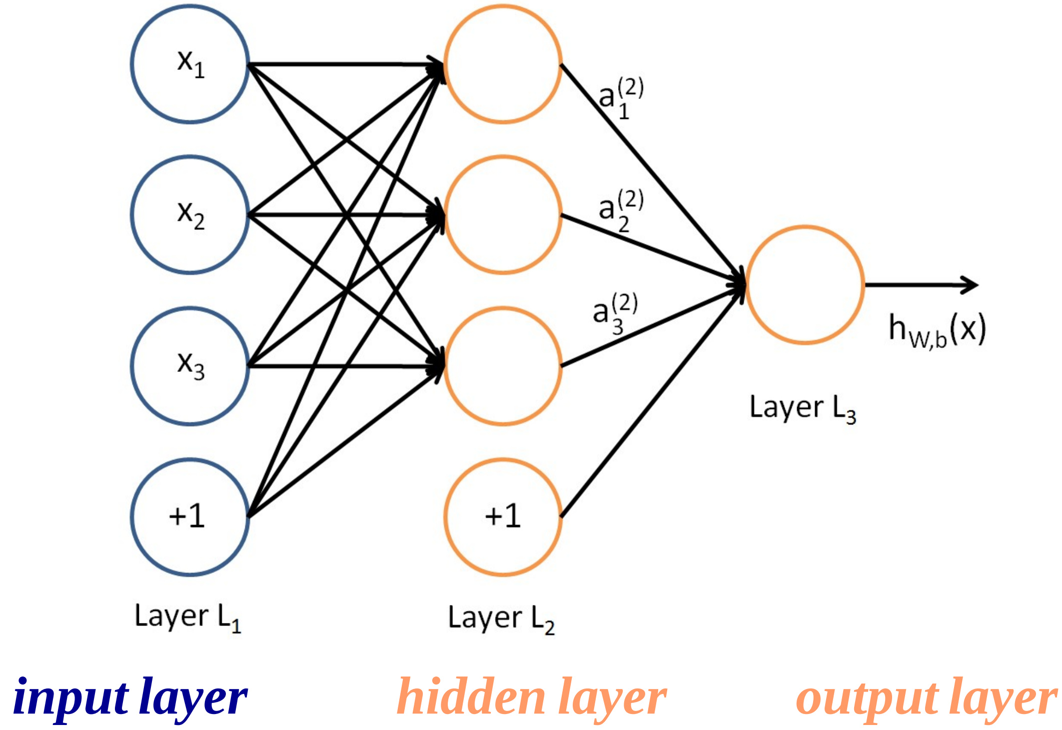


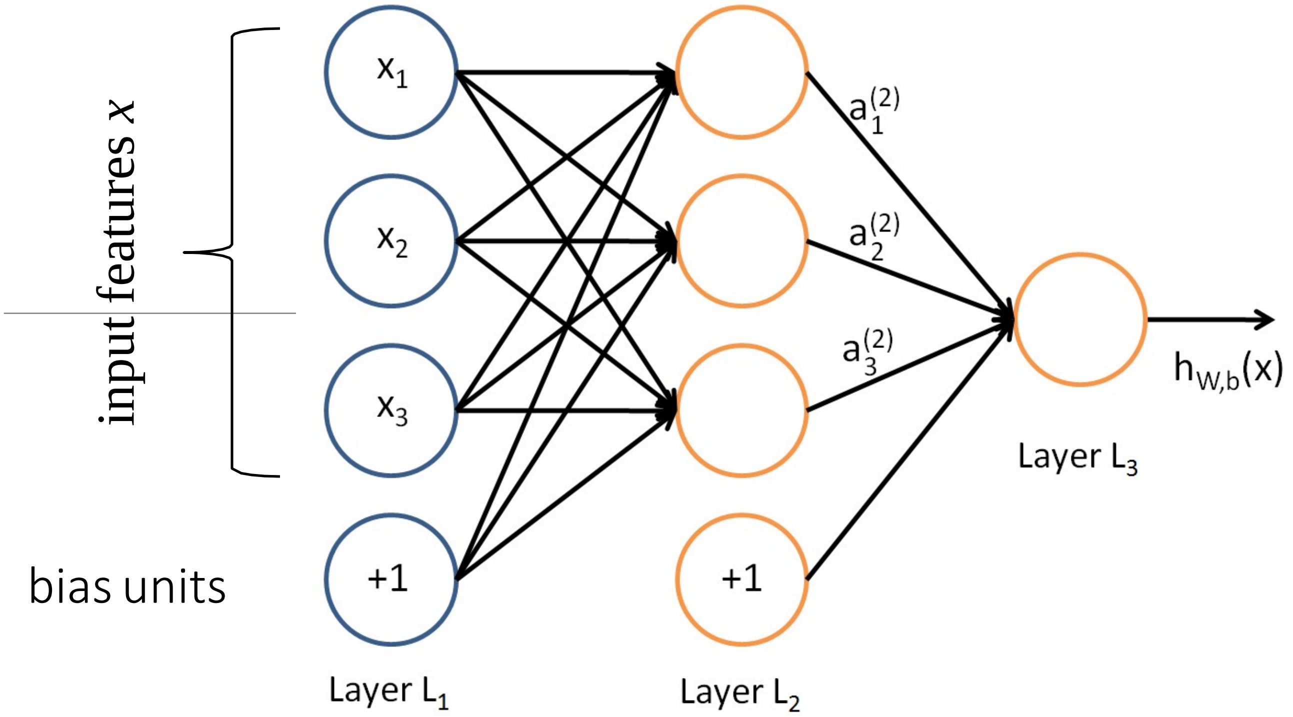
$$F(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i \sigma(\mathbf{w}_i^T \mathbf{x} + b_i)$$

$$|F(\mathbf{x}) - f(\mathbf{x})| < \varepsilon, \ \forall \mathbf{x} \in I_m$$

Neural Network Model

- Put together many neurons in layers, such that the output of a neuron can be the input of another:





- n_l =3 is the number of **layers**.
  - L_1 is the input layer, L_3 is the output layer
- (**W**, **b**) = ($W^{(1)}$, $b^{(1)}$, $W^{(2)}$, $b^{(2)}$) are the parameters:
  - $W^{(l)}_{ij}$ is the **weight** of the connection between unit j in layer l and unit i in layer l + 1.
  - $b^{(l)}_i$ is the **bias** associated unit i in layer l + 1.
- $a^{(l)}_i$ is the **activation** of unit i in layer l, e.g. $a^{(1)}_i = x_i$ and $a^{(3)}_1 = h_{W,b}(x)$.

Inference: Forward Propagation

- The activations in the hidden layer are:

$$a_1^{(2)} = f(W_{11}^{(1)}x_1 + W_{12}^{(1)}x_2 + W_{13}^{(1)}x_3 + b_1^{(1)})$$
$$a_2^{(2)} = f(W_{21}^{(1)}x_1 + W_{22}^{(1)}x_2 + W_{23}^{(1)}x_3 + b_2^{(1)})$$
$$a_3^{(2)} = f(W_{31}^{(1)}x_1 + W_{32}^{(1)}x_2 + W_{33}^{(1)}x_3 + b_3^{(1)})$$

- The activations in the output layer are:

$$h_{W,b}(x) = a_1^{(3)} = f(W_{11}^{(2)}a_1^{(2)} + W_{12}^{(2)}a_2^{(2)} + W_{13}^{(2)}a_3^{(2)} + b_1^{(2)})$$

- Compressed notation:

$a_i^{(l)} = f(z_i^{(l)})$ where $z_i^{(2)} = \sum_{j=1}^{n} W_{ij}^{(1)}x_j + b_i^{(1)}$

Forward Propagation

- Forward propagation (unrolled):

$$a_1^{(2)} = f(W_{11}^{(1)}x_1 + W_{12}^{(1)}x_2 + W_{13}^{(1)}x_3 + b_1^{(1)})$$
$$a_2^{(2)} = f(W_{21}^{(1)}x_1 + W_{22}^{(1)}x_2 + W_{23}^{(1)}x_3 + b_2^{(1)})$$
$$a_3^{(2)} = f(W_{31}^{(1)}x_1 + W_{32}^{(1)}x_2 + W_{33}^{(1)}x_3 + b_3^{(1)})$$
$$h_{W,b}(x) = a_1^{(3)} = f(W_{11}^{(2)}a_1^{(2)} + W_{12}^{(2)}a_2^{(2)} + W_{13}^{(2)}a_3^{(2)} + b_1^{(2)})$$

- Forward propagation (compressed):

$$z^{(2)} = W^{(1)}x + b^{(1)}$$
$$a^{(2)} = f(z^{(2)})$$
$$z^{(3)} = W^{(2)}a^{(2)} + b^{(2)}$$
$$h_{W,b}(x) = a^{(3)} = f(z^{(3)})$$

- Element-wise application:

$f(\mathbf{z}) = [f(z_1), f(z_2), f(z_3)]$

Forward Propagation

- Forward propagation (compressed):

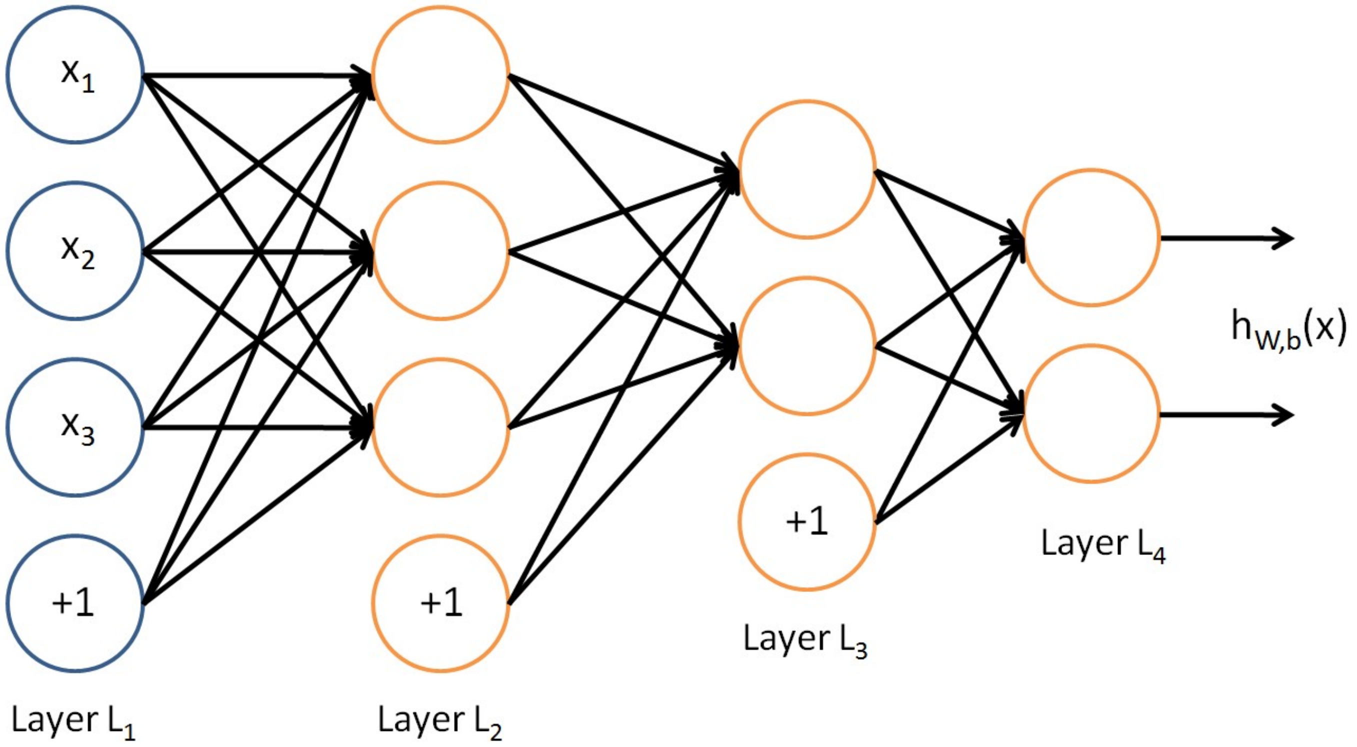
$$
\begin{aligned}
z^{(2)} &= W^{(1)}x + b^{(1)} \\
a^{(2)} &= f(z^{(2)}) \\
z^{(3)} &= W^{(2)}a^{(2)} + b^{(2)} \\
h_{W,b}(x) &= a^{(3)} = f(z^{(3)})
\end{aligned}
$$

- Composed of two ***forward propagation steps***:

$$
\begin{aligned}
z^{(l+1)} &= W^{(l)}a^{(l)} + b^{(l)} \\
a^{(l+1)} &= f(z^{(l+1)})
\end{aligned}
$$

Multiple Hidden Units, Multiple Outputs

- Write down the forward propagation steps for:



Learning: Backpropagation

- Regularized sum of squares error:

$$J(W,b,x,y) = \frac{1}{2}\left\| h_{W,b}(x) - y \right\|^2$$

$$J(W,b) = \frac{1}{m}\sum_{k=1}^{m} J(W,b,x^{(k)},y^{(k)}) + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(W_{ij}^{(l)}\right)^2$$

- Gradient: ?

$$\frac{\partial J(W,b)}{\partial W_{ij}^{(l)}} = \frac{1}{m}\sum_{k=1}^{m}\frac{\partial J(W,b,x^{(k)},y^{(k)})}{\partial W_{ij}^{(l)}} + \lambda W_{ij}^{(l)}$$

$$\frac{\partial J(W,b)}{\partial b_{i}^{(l)}} = \frac{1}{m}\sum_{k=1}^{m}\frac{\partial J(W,b,x^{(k)},y^{(k)})}{\partial b_{i}^{(l)}}$$

Backpropagation

- Need to compute the gradient of the squared error with respect to a single training example (x, y):

$$J(W,b,x,y) = \frac{1}{2}\left\| h_{W,b}(x) - y \right\|^2 = \frac{1}{2}\left\| a^{(n_l)} - y \right\|^2$$

$$\frac{\partial J}{\partial W_{ij}^{(l)}} = ? \qquad \frac{\partial J}{\partial b_i^{(l)}} = ?$$

Univariate Chain Rule for Differentiation

- Univariate Chain Rule:

$$f = f \circ g \circ h = f(g(h(x)))$$

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial h}\frac{\partial h}{\partial x}$$

- Example:

$$f(g(x)) = 2g(x)^2 - 3g(x) + 1$$

$$g(x) = x^3 + 2x$$

Multivariate Chain Rule for Differentiation

- Multivariate Chain Rule:

$$f = f(g_1(x), g_2(x), \ldots, g_n(x))$$

$$\frac{\partial f}{\partial x} = \sum_{i=1}^{n} \frac{\partial f}{\partial g_i} \frac{\partial g_i}{\partial x}$$

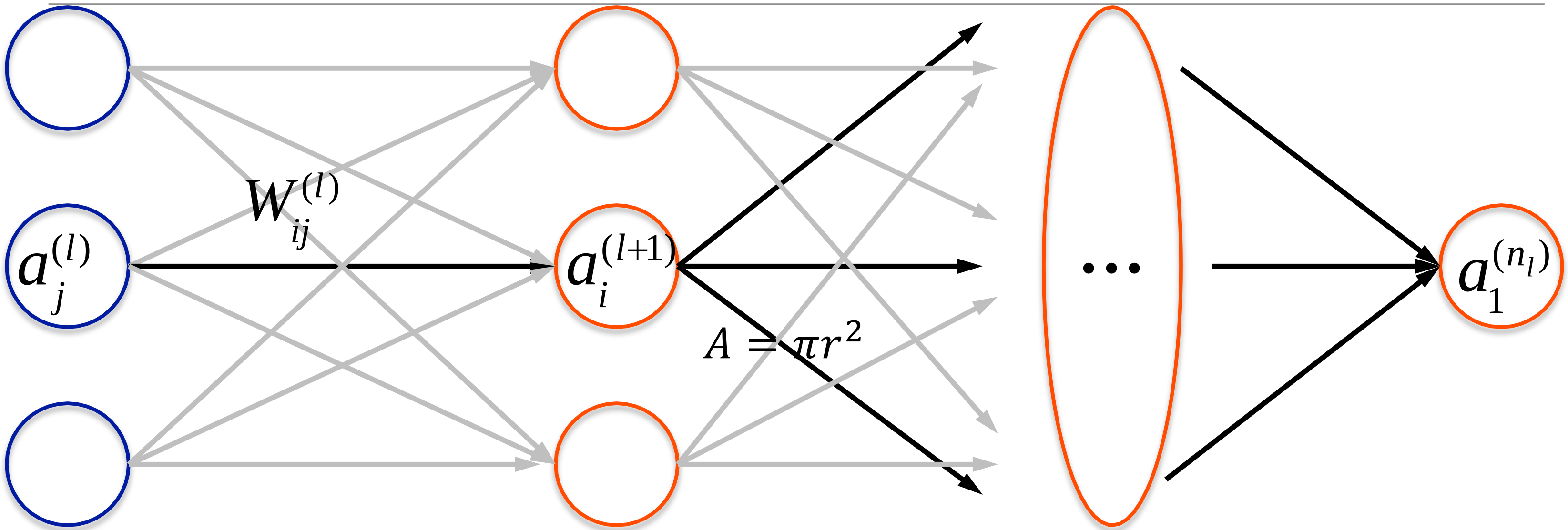
- Example:

$$f(g_1(x), g_2(x)) = 2g_1(x)^2 - 3g_1(x)g_2(x) + 1$$

$$g_1(x) = 3x$$

$$g_2(x) = x^2 + 2x$$

Backpropagation: $W^{(l)}_{ij}$



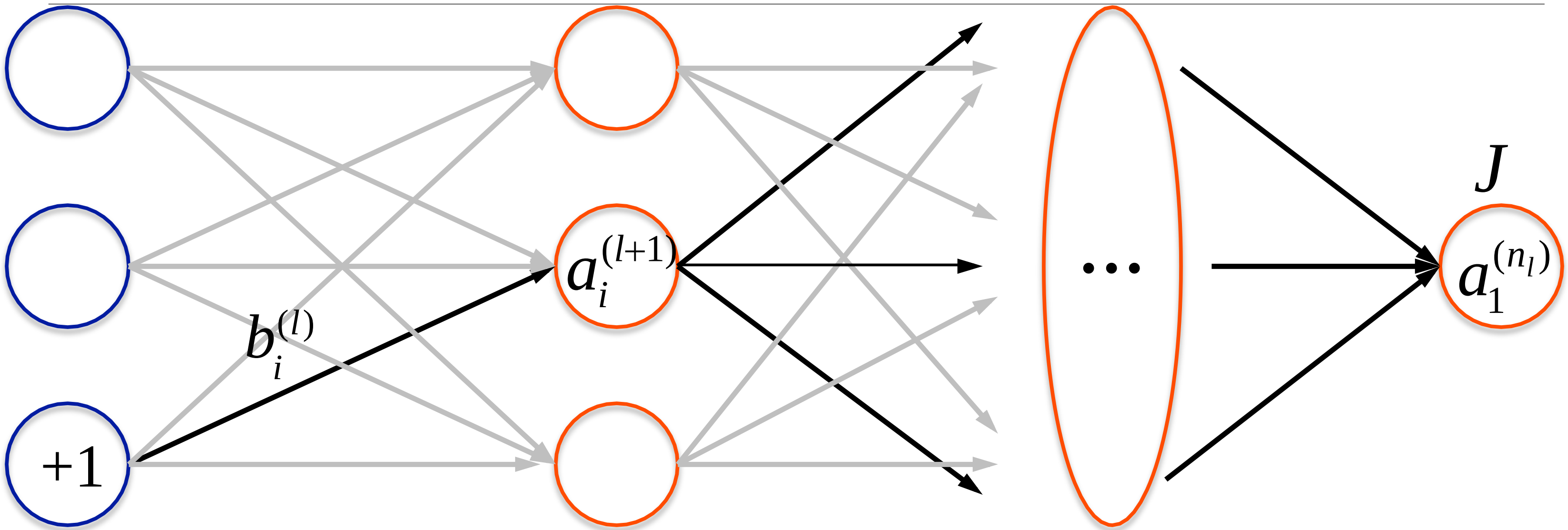
- J depends on $W_{ij}^{(l)}$ only through $a_i^{(l+1)}$, which depends on $W_{ij}^{(l)}$ only through $z_i^{(l+1)}$.

$$a_i^{(l+1)} = f\left(z_i^{(l+1)}\right)$$

$$J(W, b, x, y) = \frac{1}{2}||a^{(n_l)} - y||^2$$

$$z_i^{(l+1)} = \sum_{j=1}^{s_l} w_{ij}^{(l)} a_j^{(l)} + b_i^{(l)}$$

Backpropagation: $b_i^{(l)}$



- J depends on $b_i^{(l)}$ only through $a_i^{(l+1)}$, which depends on $b_i^{(l)}$ only through $z_i^{(l+1)}$.

$$a_i^{(l+1)} = f\left(z_i^{(l+1)}\right)$$

$$J(W,b,x,y) = \tfrac{1}{2}||a^{(n_l)} - y||^2$$

$$z_i^{(l+1)} = \sum_{j=1}^{s_l} w_{ij}^{(l)} a_j^{(l)} + b_i^{(l)}$$

Backpropagation: $W_{ij}^{(l)}$ and $b_i^{(l)}$

$$\frac{\partial J}{\partial W_{ij}^{(l)}} = \underbrace{\frac{\partial J}{\partial a_i^{(l+1)}} \times \frac{\partial a_i^{(l+1)}}{\partial z_i^{(l+1)}}}_{\delta_i^{(l+1)}} \times \underbrace{\frac{\partial z_i^{(l+1)}}{\partial W_{ij}^{(l)}}}_{a_j^{(l)}} = a_j^{(l)} \delta_i^{(l+1)}$$

How to compute $\delta_i^{(l)}$ for all layers l ?

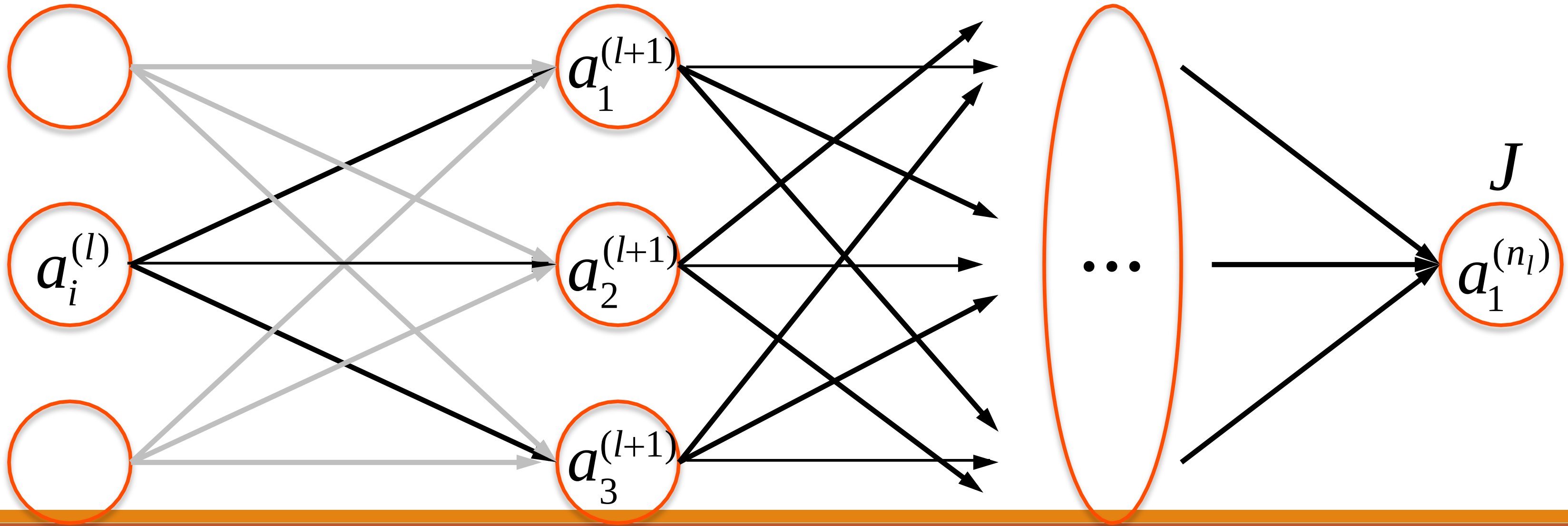
$$\frac{\partial J}{\partial b_i^{(l)}} = \underbrace{\frac{\partial J}{\partial a_i^{(l+1)}} \times \frac{\partial a_i^{(l+1)}}{\partial z_i^{(l+1)}}}_{\delta_i^{(l+1)}} \times \underbrace{\frac{\partial z_i^{(l+1)}}{\partial b_i^{(l)}}}_{+1} = \delta_i^{(l+1)}$$

Backpropagation: $\delta_i^{(l)}$

$$\delta_i^{(l)} = \frac{\partial J}{\partial a_i^{(l)}} \times \frac{\partial a_i^{(l)}}{\partial z_i^{(l)}} = \frac{\partial J}{\partial a_i^{(l)}} \times f'(z_i^{(l)})$$

?

- J depends on $a_i^{(l)}$ only through $a_1^{(l+1)}$, $a_2^{(l+1)}$, …



Backpropagation: $\delta_i^{(l)}$

- J depends on $a_i^{(l)}$ only through $a_1^{(l+1)}$, $a_2^{(l+1)}$, ...

$$\frac{\partial J}{\partial a_i^{(l)}} = \sum_{j=1}^{s_{l+1}} \frac{\partial J}{\partial a_j^{(l+1)}} \times \frac{\partial a_j^{(l+1)}}{\partial a_i^{(l)}} = \sum_{j=1}^{s_{l+1}} \underbrace{\frac{\partial J}{\partial a_j^{(l+1)}} \times \frac{\partial a_j^{(l+1)}}{\partial z_j^{(l+1)}}}_{\delta_j^{(l+1)}} \times \underbrace{\frac{\partial z_j^{(l+1)}}{\partial a_i^{(l)}}}_{W_{ji}^{(l)}}$$

- Therefore, $\delta_i^{(l)}$ can be computed as:

$$\delta_i^{(l)} = \frac{\partial J}{\partial a_i^{(l)}} \times f'(z_i^{(l)}) = \left(\sum_{j=1}^{s_{l+1}} W_{ji}^{(l)} \delta_j^{(l+1)} \right) \times f'(z_i^{(l)})$$

Backpropagation: $\delta_i^{(l)}$

- Start computing δ's for the output layer:

$$\delta_i^{(n_l)} = \frac{\partial J}{\partial a_i^{(n_l)}} \times \frac{\partial a_i^{(n_l)}}{\partial z_i^{(n_l)}} = \frac{\partial J}{\partial a_i^{(n_l)}} \times f'(z_i^{(n_l)})$$

$$J = \frac{1}{2}\left\|a^{(n_l)} - y\right\|^2 \Rightarrow \frac{\partial J}{\partial a_i^{(n_l)}} = \left(a_i^{(n_l)} - y_i\right)$$

$$\delta_i^{(n_l)} = \left(a_i^{(n_l)} - y_i\right) \times f'(z_i^{(n_l)})$$

Backpropagation Algorithm

1. Feedforward pass on x to compute activations $a_i^{(l)}$
2. For each output unit i compute:

$$\delta_i^{(n_l)} = \left(a_i^{(n_l)} - y_i\right) \times f'(z_i^{(n_l)})$$

3. For $l = n_l-1, n_l-2, n_l-3, ..., 2$ compute:

$$\delta_i^{(l)} = \left(\sum_{j=1}^{s_{l+1}} W_{ji}^{(l)} \delta_j^{(l+1)}\right) \times f'(z_i^{(l)})$$

4. Compute the partial derivatives of the cost $J(W, b, x, y)$

$$\frac{\partial J}{\partial W_{ij}^{(l)}} = a_j^{(l)} \delta_i^{(l+1)} \qquad \frac{\partial J}{\partial b_i^{(l)}} = \delta_i^{(l+1)}$$

Backpropagation Algorithm: Vectorization

1. Feedforward pass on x to compute activations $a_i^{(l)}$
2. For each output unit i compute:

$$\delta^{(n_l)} = (a^{(n_l)} - y) * f'(z^{(n_l)})$$

3. For $l = n_l-1, n_l-2, n_l-3, \ldots, 2$ compute:

$$\delta^{(l)} = ((W^{(l)})^T \delta^{(l+1)}) f'(z^{(l)})$$

4. Compute the partial derivatives of the cost $J(W, b, x, y)$

$$\nabla_{W^{(l)}} J = \delta^{(l+1)} \left(a^{(l)}\right)^T \qquad \nabla_{b^{(l)}} J = \delta^{(l+1)}$$