Gradient Descent and Backpropagation

Prof. Ankur Sinha

Indian Institute of Management Ahmedabad

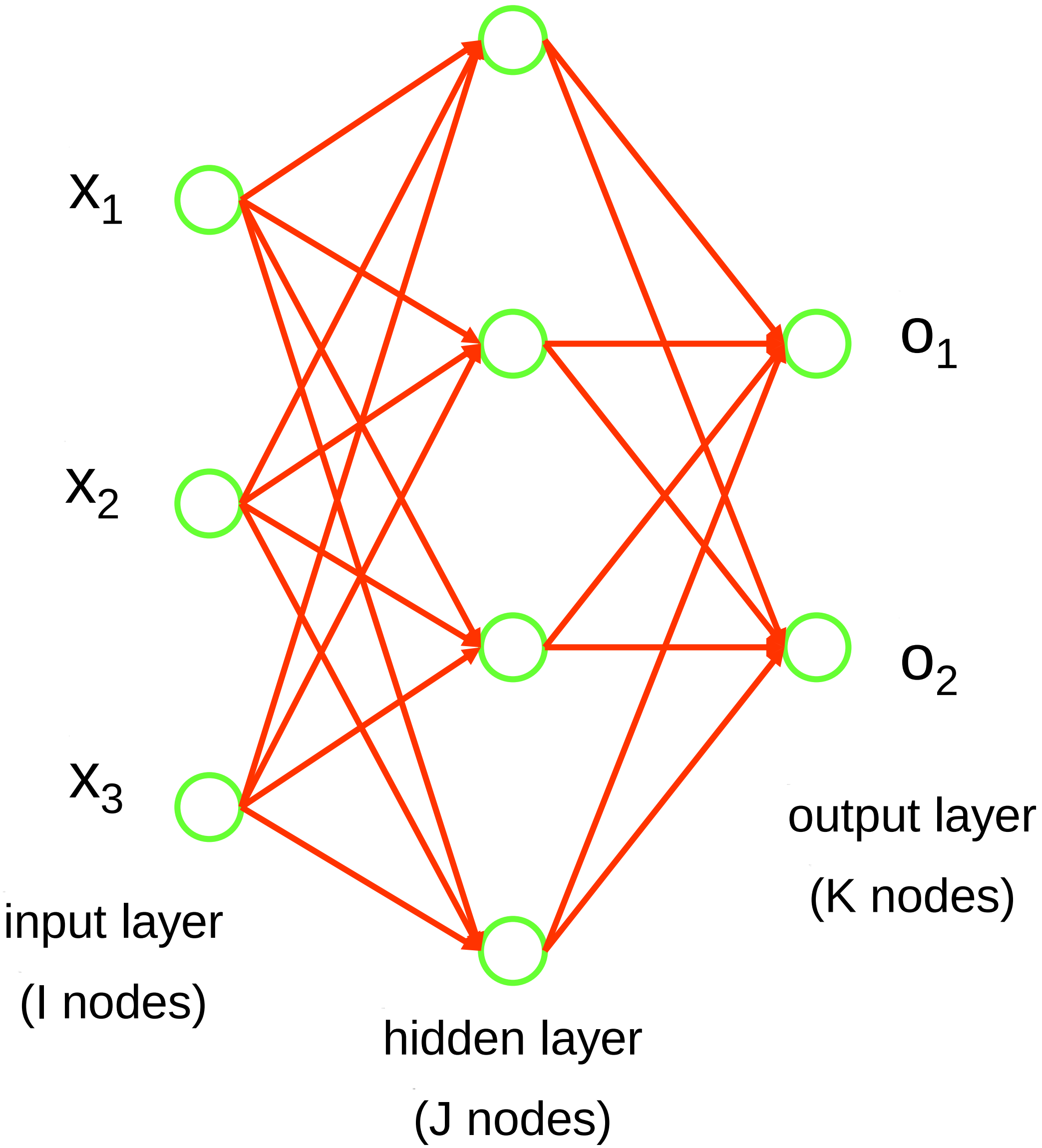
Gradient descent

- Moving in the direction which is the negative of the gradient leads to a local optimum of an unconstrained objective function
- Neural network attempt to minimize loss (error) for a large number of training samples
- The model parameters (weights) of the neural network are expected to be optimized, which give the minimum loss
- The following provides the first derivative (gradient) and the second derivative (hessian) of a multivariate function

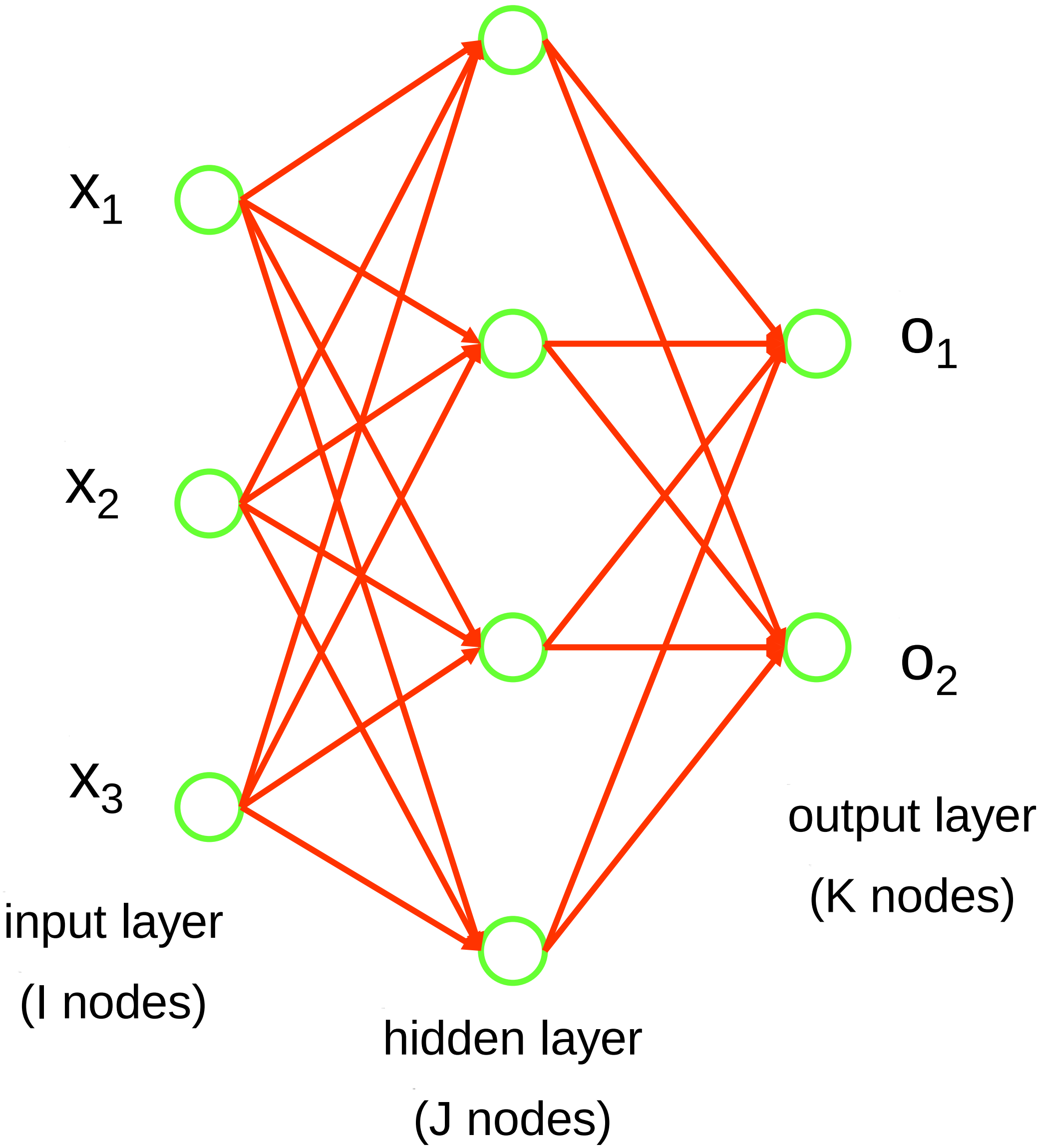
Note that the derivative is a vector that represents a direction

$$\nabla f = \begin{pmatrix} \dfrac{\partial f}{\partial x_1} \\ \dfrac{\partial f}{\partial x_2} \\ \vdots \\ \dfrac{\partial f}{\partial x_n} \end{pmatrix} \quad \text{and} \quad \nabla^2 f = \begin{pmatrix} \dfrac{\partial^2 f}{\partial x_1^2} & \dfrac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_1 \partial x_n} \\ \dfrac{\partial^2 f}{\partial x_2 \partial x_1} & \dfrac{\partial^2 f}{\partial x_2^2} & & \dfrac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & & \ddots & \vdots \\ \dfrac{\partial^2 f}{\partial x_n \partial x_1} & \dfrac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_n^2} \end{pmatrix}$$

Network function f: $\mathbf{R}^3 \rightarrow \mathbf{R}^2$

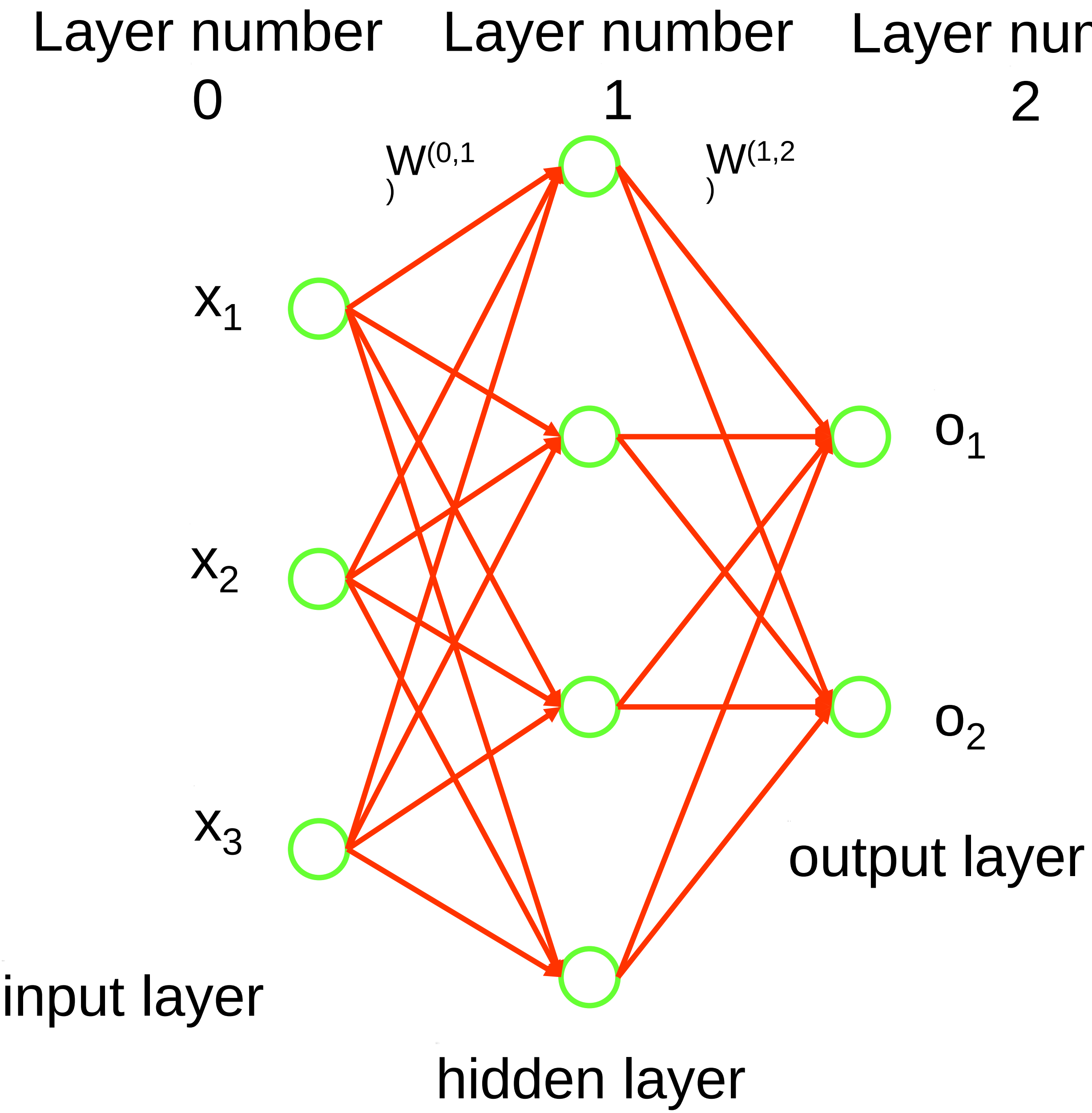


Network function f: $\mathbf{R}^3 \rightarrow \mathbf{R}^2$



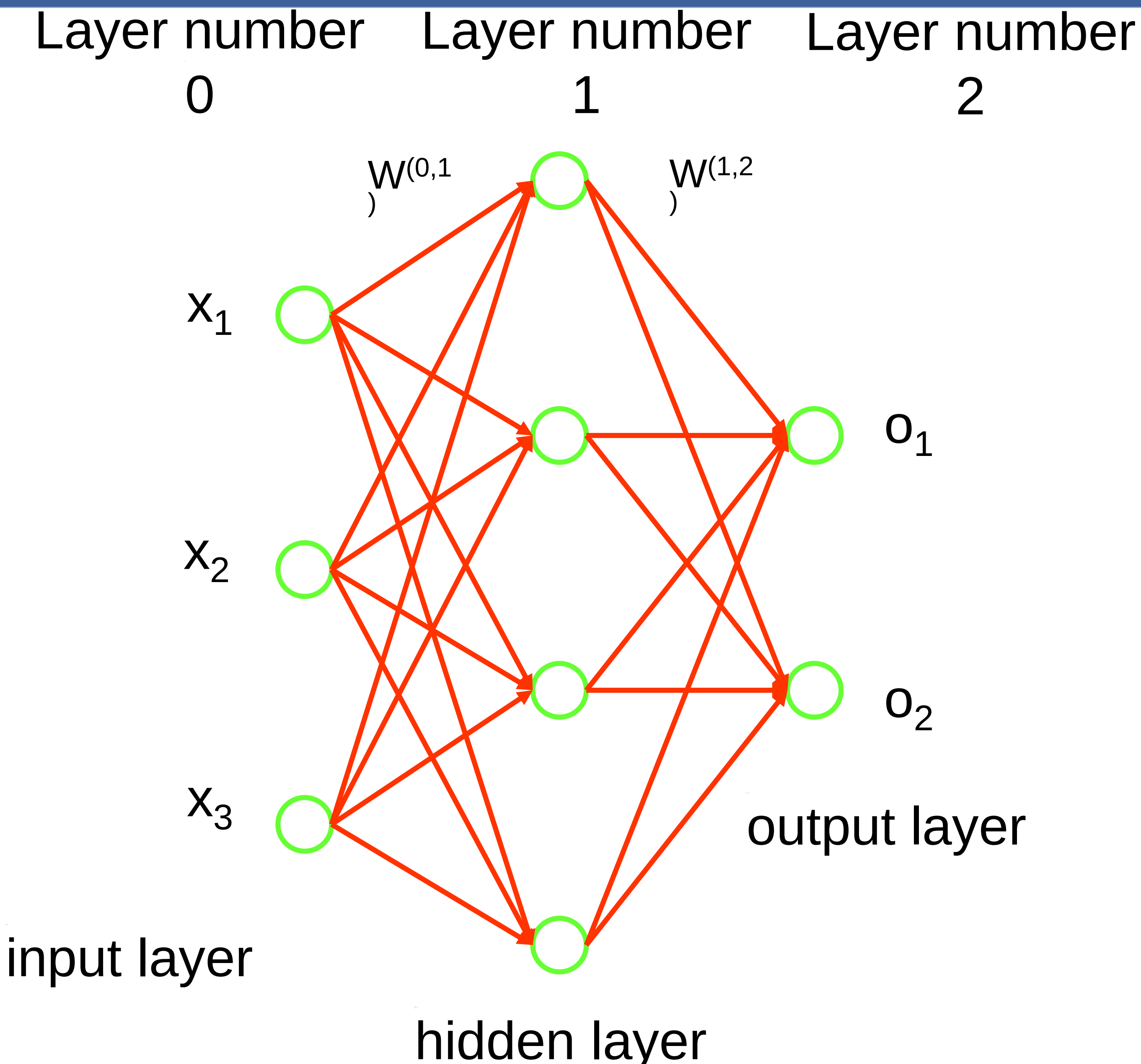
Input neurons:	i=1, … , I
Hidden neurons:	j=1, … , J
Output neurons:	k=1, … , K
Training data:	n=1, … , N

Input-output pairs
$\{(\mathbf{x}_n, \mathbf{d}_n) \mid n = 1, \ldots, N\}$ constitutes the training set



$W^{(0,1)}$ Weight matrix
From input to hidden layer

$W^{(1,2)}$ Weight matrix
From hidden to output layer



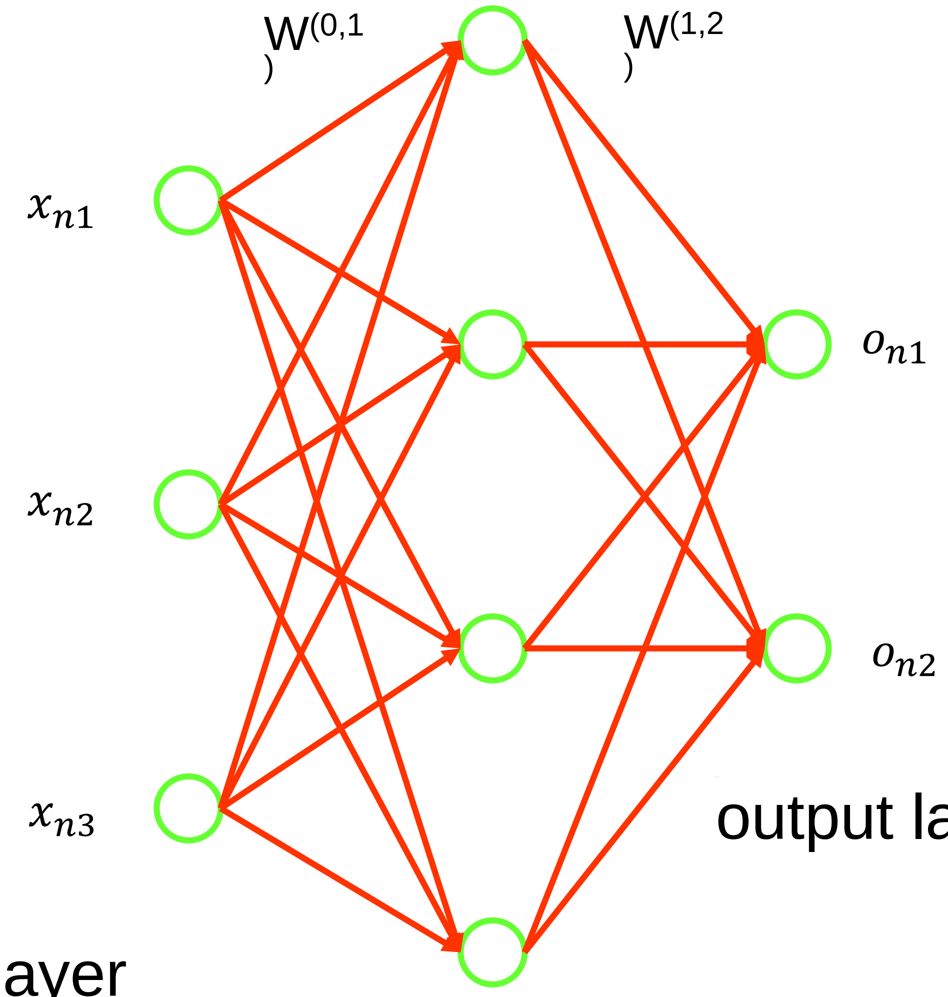
Net input to j-th node in hidden layer:

$$net_{nj}^{(0,1)} = \sum_{i=1}^{I} w_{ij}^{(0,1)} x_{ni}$$

Output of j-th node in hidden layer:

$$y_{nj}^{(1,2)} = S\left(\sum_{i=1}^{I} w_{ij}^{(0,1)} x_{ni}\right)$$

Layer number 0 | Layer number 1 | Layer number 2



output layer

input layer

hidden layer

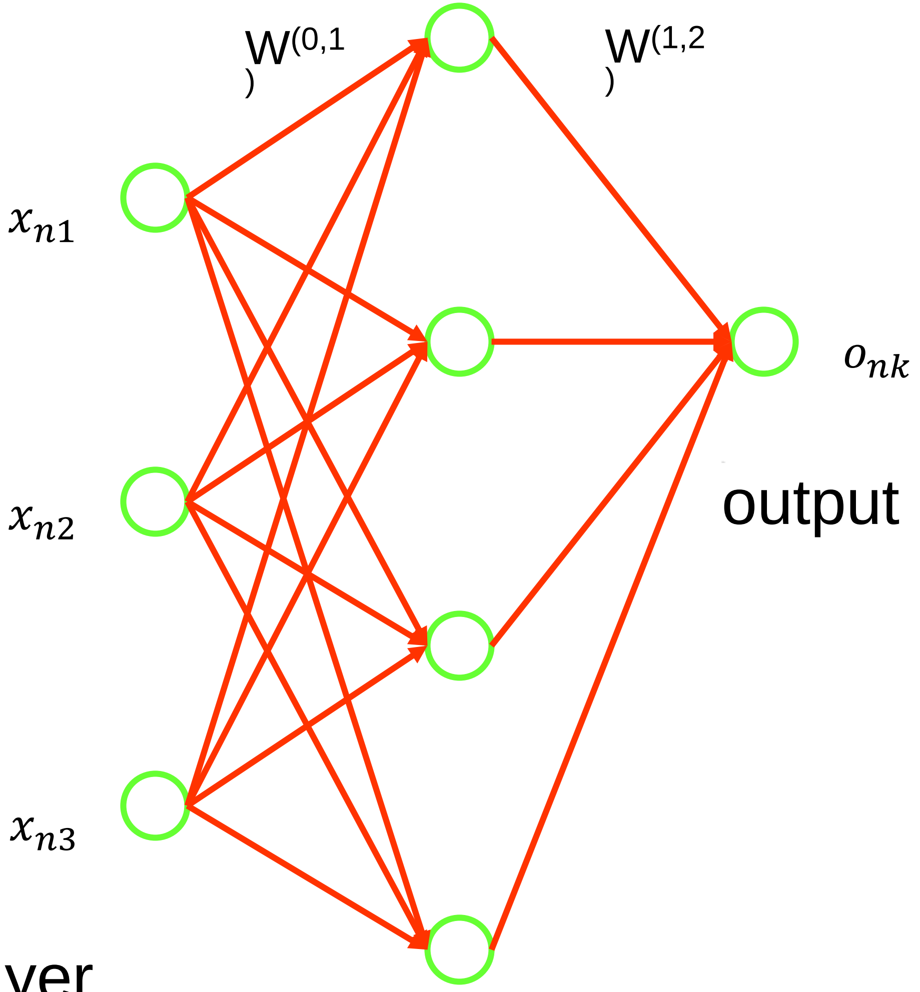
Net input to k-th node in output layer:

$$net_{nk}^{(1,2)} = \sum_{j=1}^{J} w_{jk}^{(1,2)} y_{nj}^{(1,2)}$$

Output of k-th node in output layer:

$$o_{nk} = S\left(\sum_{j=1}^{J} w_{jk}^{(1,2)} y_{nj}^{(1,2)}\right)$$

Layer number 0 | Layer number 1 | Layer number 2



output layer

input layer

hidden layer

For simplicity consider a single output

$$E_n = (l_{nk})^2 = (o_{nk} - d_{nk})^2$$

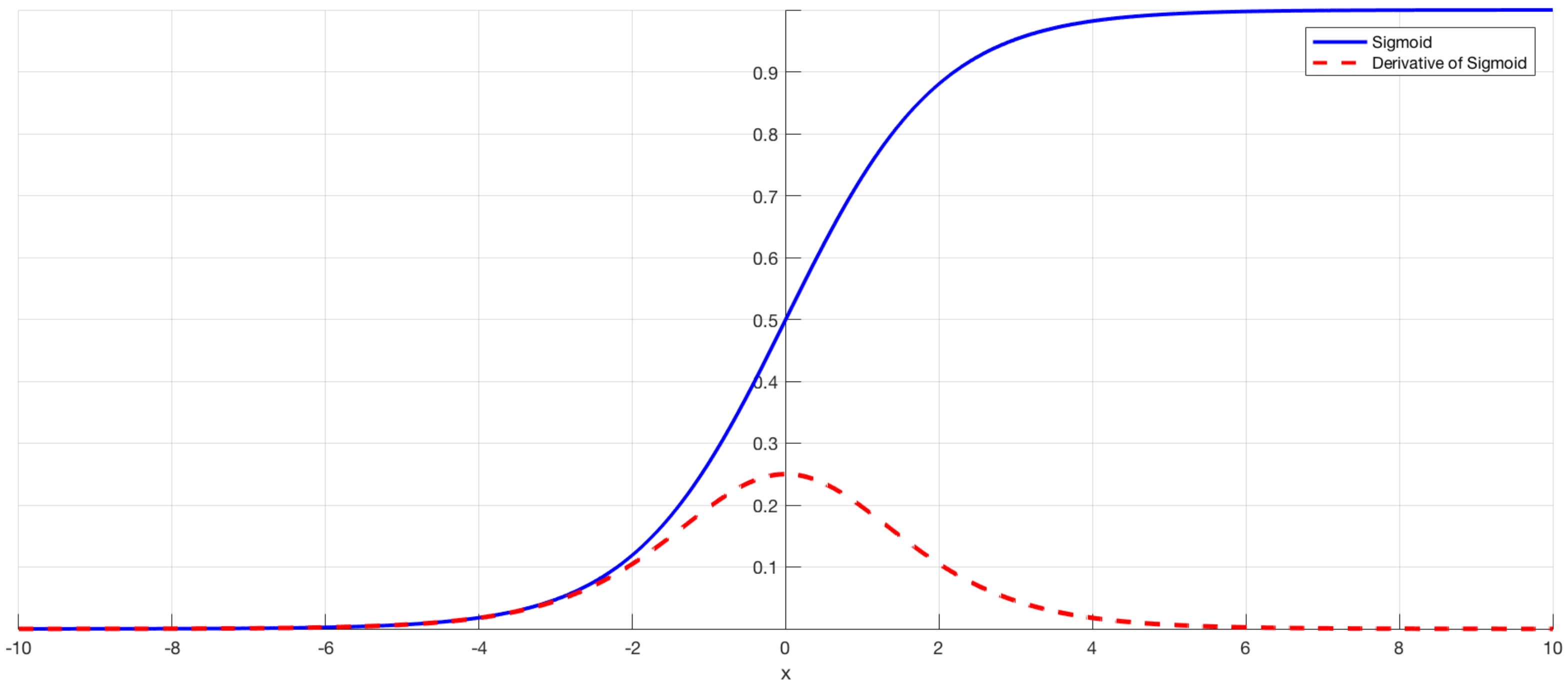
$$\text{MSE} = \frac{1}{N}\sum_{n=1}^{N}(l_{nk})^2$$

Network error for kth output

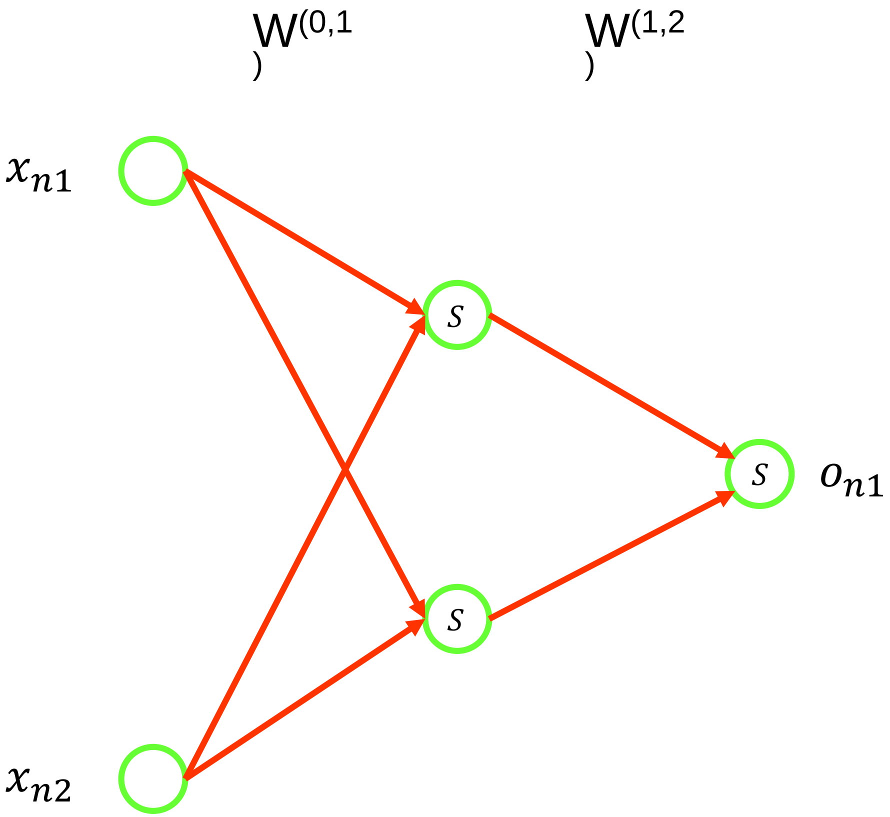
Let S be a sigmoid function

$$S(z) = \frac{1}{1 + e^{-z}}$$

$$S'(z) = S(z)(1 - S(z))$$

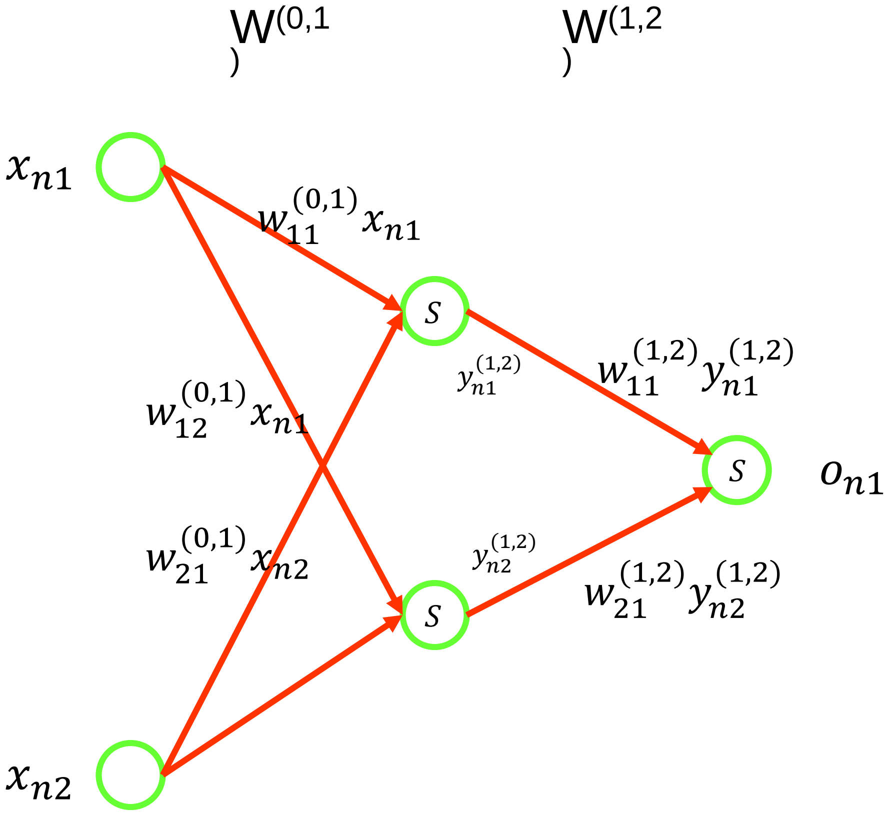


Exercise: Fill up the network with symbols



$$E = \frac{1}{N}\sum_{n=1}^{N}(o_{nk} - d_{nk})^2$$

Exercise: Identify gradients



$$E = \frac{1}{N}\sum_{n=1}^{N}(o_{nk} - d_{nk})^2$$

$$\Delta w_{jk}^{(1,2)} \propto \frac{-\partial E}{\partial w_{jk}^{(1,2)}}$$

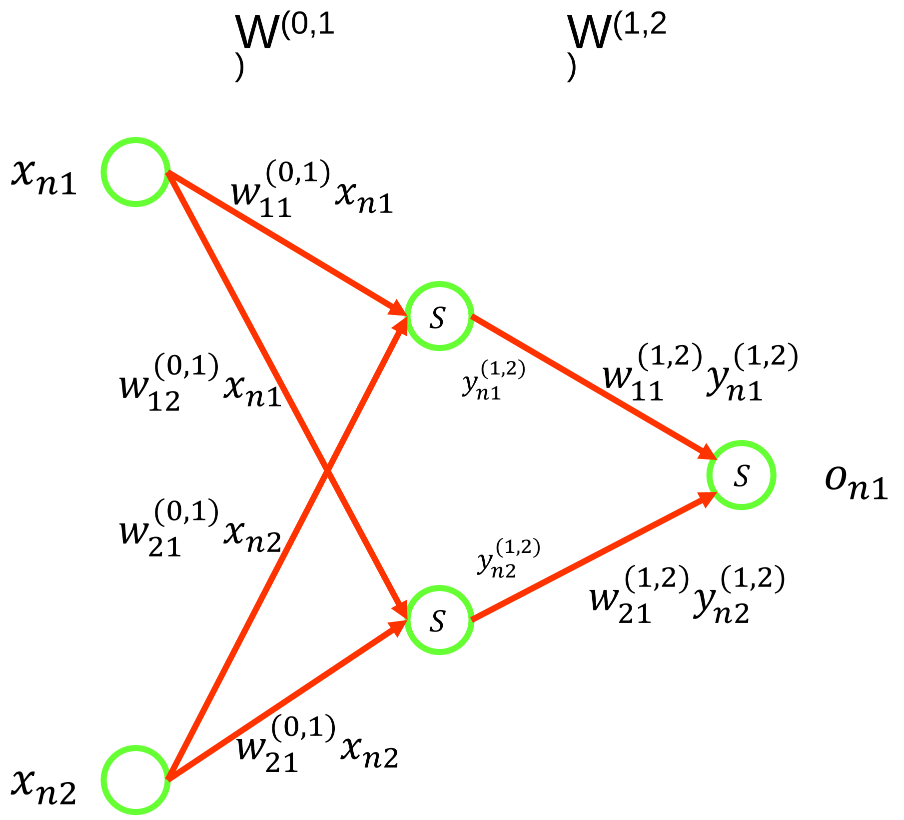
$$\Delta w_{ij}^{(0,1)} \propto \frac{-\partial E}{\partial w_{ij}^{(0,1)}}$$

Identify $\frac{-\partial E}{\partial w_{11}^{(1,2)}}, \frac{-\partial E}{\partial w_{21}^{(1,2)}}$

Identify $\frac{-\partial E}{\partial w_{11}^{(0,1)}}, \frac{-\partial E}{\partial w_{12}^{(0,1)}}, \frac{-\partial E}{\partial w_{21}^{(0,1)}}, \frac{-\partial E}{\partial w_{22}^{(0,1)}}$

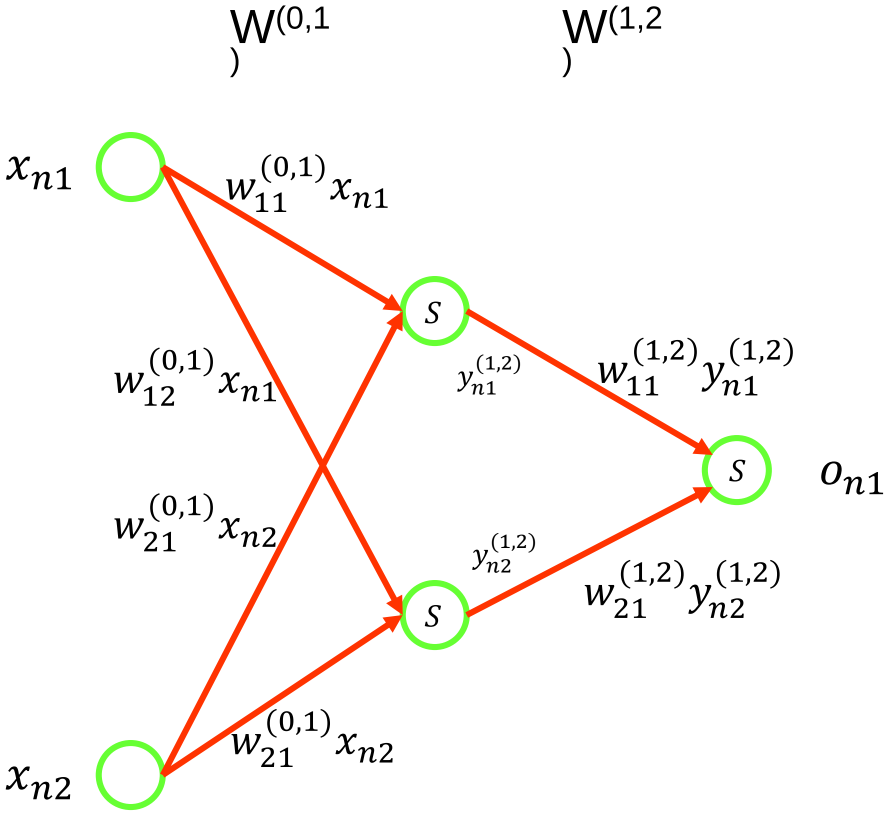
Exercise: Solution

$$E = \frac{1}{N}\sum_{n=1}^{N}(o_{nk} - d_{nk})^2$$



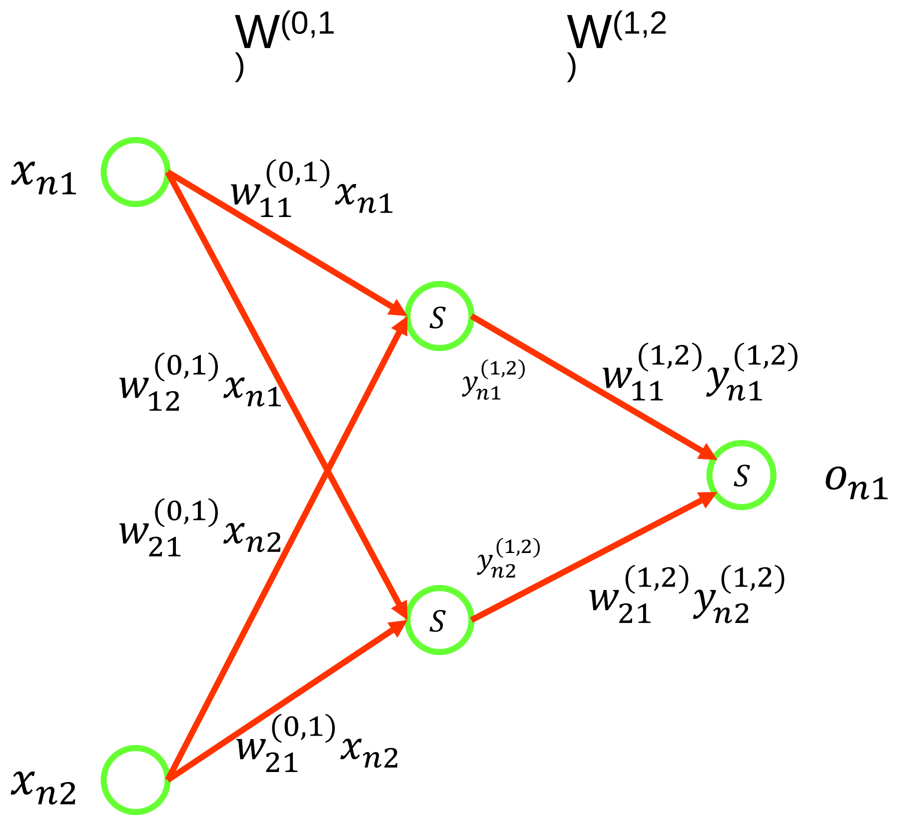
Exercise: Solution

$$E = \frac{1}{N}\sum_{n=1}^{N}(o_{nk} - d_{nk})^2$$

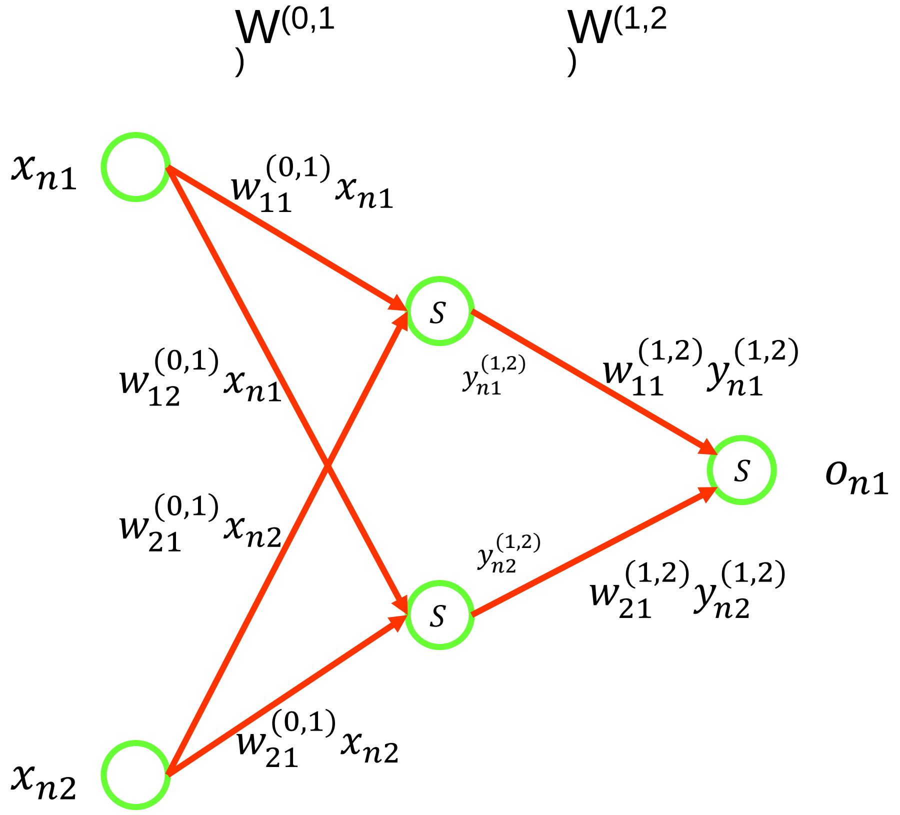


Exercise: Solution

$$E = \frac{1}{N}\sum_{n=1}^{N}(o_{nk} - d_{nk})^2$$



Exercise: Solution



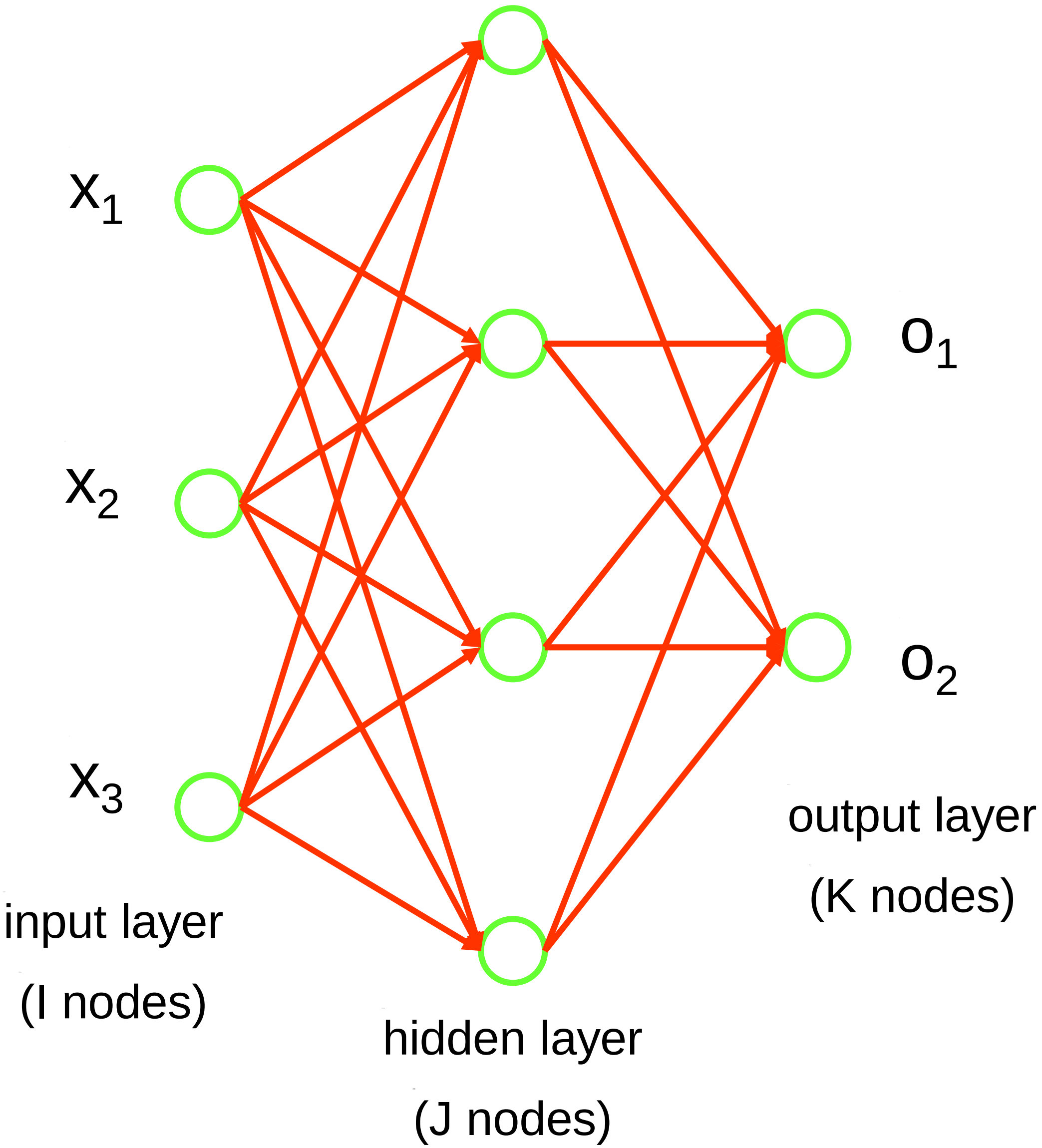
$S'\left(net_{n1}^{(1,2)}\right) y_{n1}^{(1)}$

$$\frac{\partial E}{\partial w_{11}^{(1,2)}} = \frac{1}{N}\sum_{n=1}^{N} 2(o_{nk} - d_{nk}) \frac{\partial o_{nk}}{\partial w_{11}^{(1,2)}}$$

$$\frac{\partial E}{\partial w_{11}^{(0,1)}} = \frac{1}{N}\sum_{n=1}^{N} 2(o_{nk} - d_{nk}) \frac{\partial o_{nk}}{\partial w_{11}^{(1,2)}} \frac{\partial y_{n1}^{(1,2)}}{\partial w_{11}^{(0,1)}}$$

$S'\left(net_{n1}^{(0,1)}\right) x_{n1}$

Network function f: $\mathbf{R}^3 \rightarrow \mathbf{R}^2$



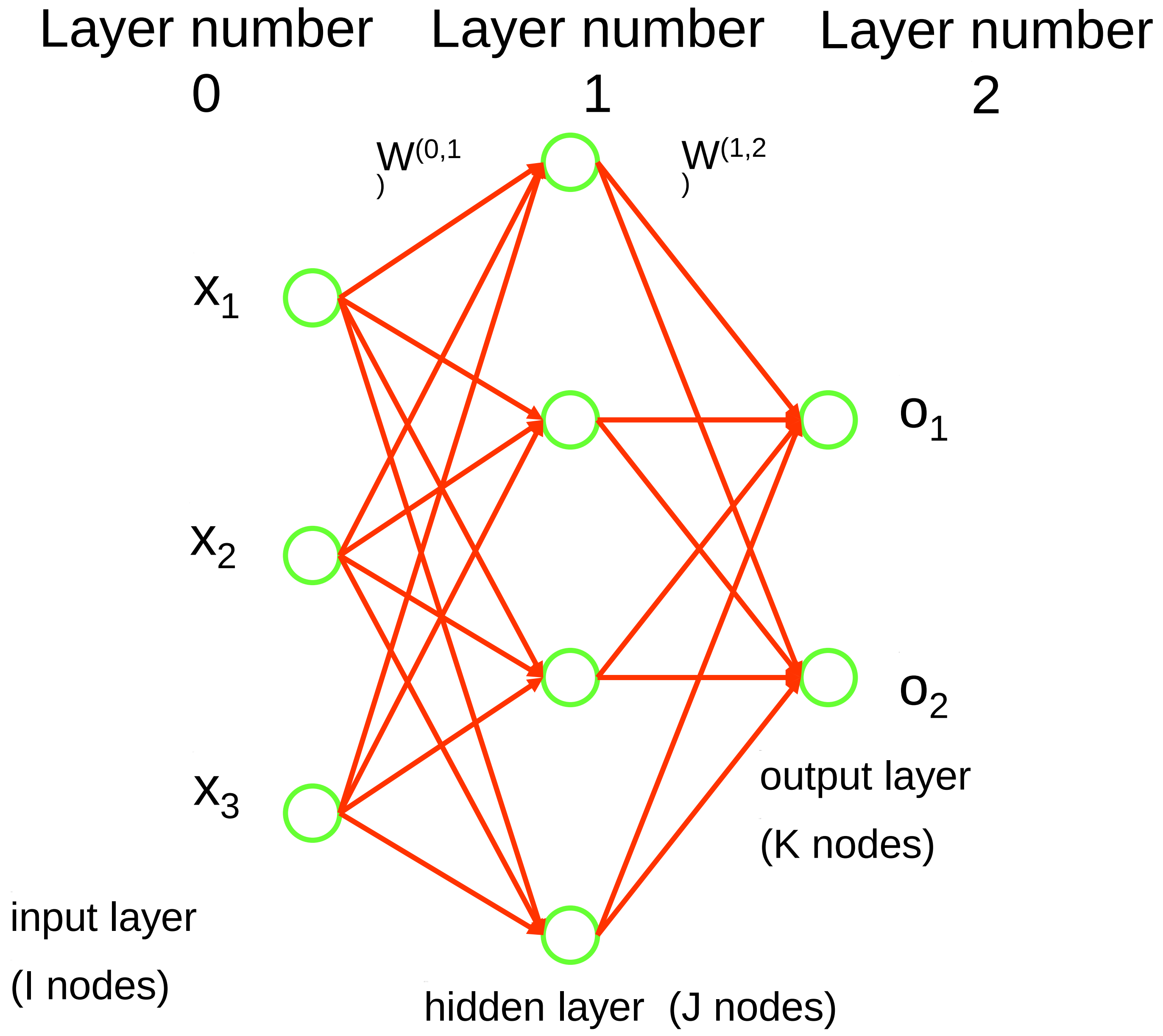
Input neurons:	i=1, … , I
Hidden neurons:	j=1, … , J
Output neurons:	k=1, … , K
Training data:	n=1, … , N

Input-output pairs

$\{(\mathbf{x}_n, \mathbf{d}_n) \mid n = 1, \ldots, N\}$ constitutes the training set

$$E_n = \sum_{k=1}^{K} (l_{nk})^2 = \sum_{k=1}^{K} (d_{nk} - o_{nk})^2$$

$$\mathrm{MSE} = \frac{1}{N} \sum_{n=1}^{N} \sum_{k=1}^{K} (l_{nk})^2$$

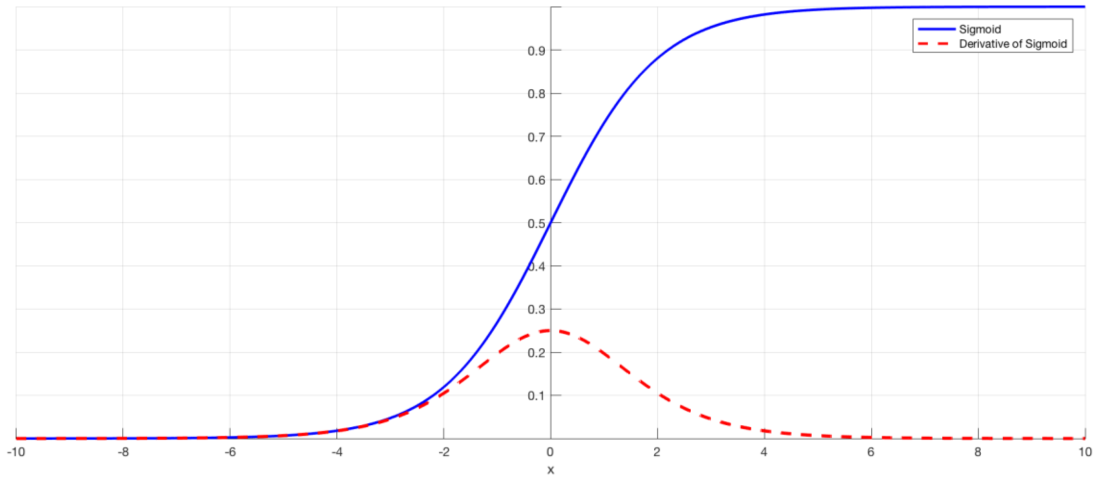


$$\Delta w_{jk}^{(1,2)} \propto \frac{-\partial E}{\partial w_{jk}^{(1,2)}}$$

$$\Delta w_{ij}^{(0,1)} \propto \frac{-\partial E}{\partial w_{ij}^{(0,1)}}$$

Vanishing and Exploding Gradients

- For large negative or positive inputs, the derivative of sigmoid is zero leading to no propagation of gradients backward. Hence learning stops.
- Choosing some other activation function that does not suffer from this issue may help with avoiding vanishing gradient
- Exploding gradient is opposite of vanishing gradient where large gradients accumulate leading to an unstable network with large weights
- Gradient clipping and weight regularization are used to handle exploding gradients



$$S(z) = \frac{1}{1 + e^{-z}}$$

$$S'(z) = S(z)(1 - S(z))$$

Thank you