Backpropagation

Dripta Mj

Department of Mathematics

Ramakrishna Mission Vivekananda Educational and Research Institute

Belur Math, India

Machine Learning

Sem 3, 2018-19

Sem 3, 2018-19

Training a neural network

- Goal – Optimize for weights:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_{n=1}^{N} L\big(\mathbf{y}^{(n)}, \mathbf{y}^{*(n)}\big)$$

  where $\mathbf{y}^{*(n)}$ is the prediction of the neural network.

- Select an appropriate loss function:
  - Squared loss:

$$L\big(\mathbf{y}^{(n)}, \mathbf{y}^{*(n)}\big) = \frac{1}{2} \sum_{j=1}^{J} \big(y_j^{(n)} - y_j^{*(n)}\big)^2$$

  - Binary cross-entropy loss:

$$L\big(y^{(n)}, y^{*(n)}\big) = -y^{(n)} \log(y^{*(n)}) - (1 - y^{(n)}) \log(1 - y^{*(n)})$$

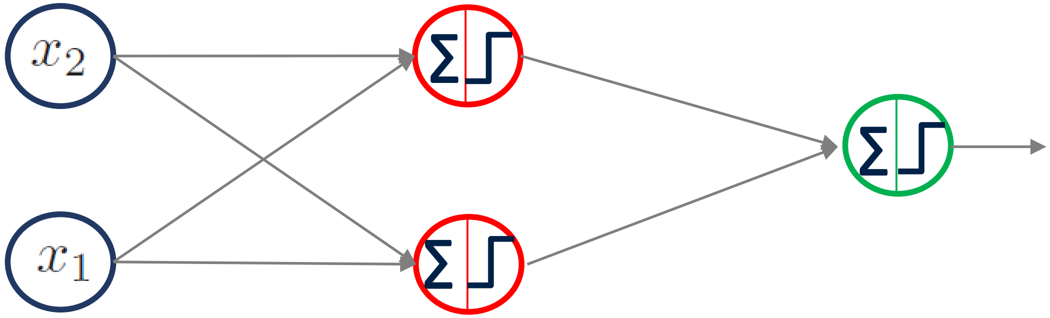
  - Cross-entropy loss:

$$L\big(\mathbf{y}^{(n)}, \mathbf{y}^{*(n)}\big) = -\sum_{j=1}^{J} y_j^{(n)} \log y_j^{*(n)}$$

- Gradient descent:

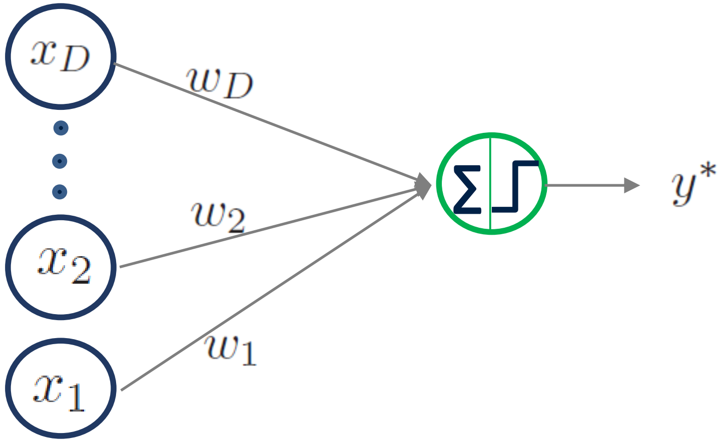
$$\mathbf{w}^{t+1} = \mathbf{w}^t - \xi \frac{\partial L}{\partial \mathbf{w}^t}$$

Procedure



- Training is achieved in two steps:
  - Step 1: Forward pass the inputs through the network.
  - Step 2: In order to adjust the parameters we go backwards. Parameters are updated using gradients.

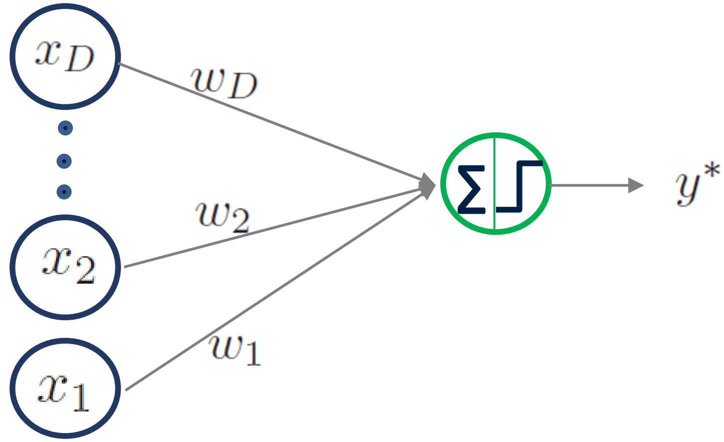
Single layer



- Problem: Binary classification
- Inputs: $x_1,\ x_2,\ \ldots x_D$
- Output: $y^* = \mathcal{A}\big(w_1 x_1 + w_2 x_2 + \ldots. + w_D x_D\big)$
  $= \mathcal{A}\big(\mathbf{w}^{\mathrm{T}}\mathbf{x}\big)$
- Loss function
  - Binary cross-entropy – this is the negative of log likelihood.

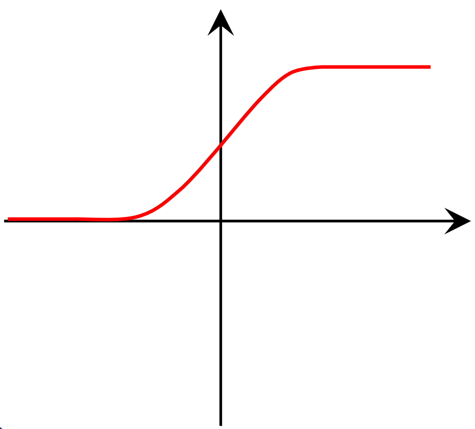
$$L = -y\log(y^*) - (1-y)\log(1-y^*)$$

Activation function



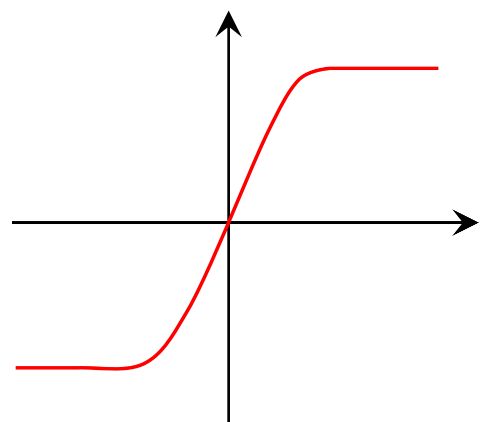
Sigmoid

$$\mathcal{A}(z) = \frac{1}{1 + \exp(-z)}$$



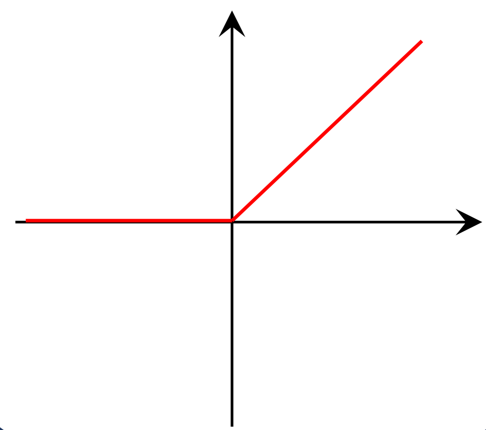
tanh

$$\mathcal{A}(z) = \frac{\exp(z) - \exp(-z)}{\exp(z) + \exp(-z)}$$

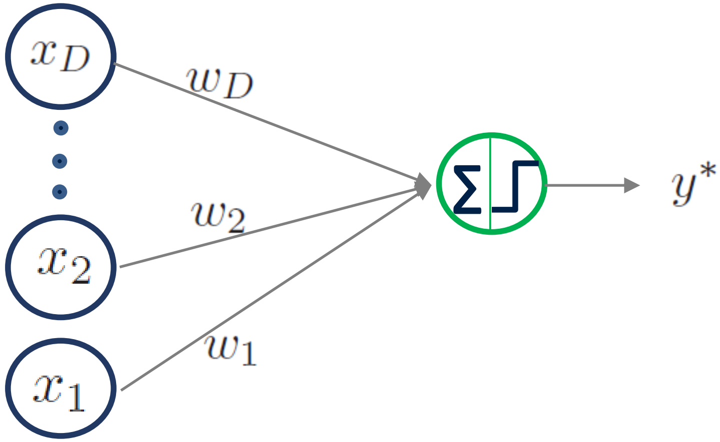


ReLU

$$\mathcal{A}(z) = \max(z, 0)$$



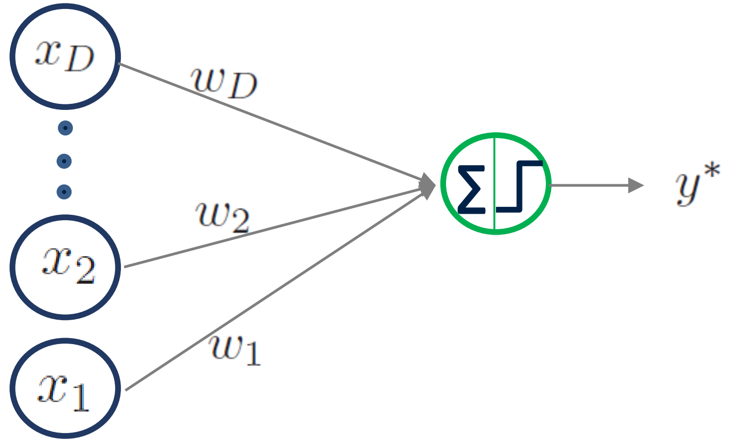
Backpropagation



- Update: $w_i := w_i - \xi \frac{\partial L}{\partial w_i}$
- The gradient can be computed as

$$\begin{aligned}
\frac{\partial L}{\partial w_i} &= \frac{\partial}{\partial w_i}\Big(- y \log(y^*) - (1-y)\log(1-y^*)\Big) \\
&= \left(- \frac{y}{y^*} + \frac{(1-y)}{(1-y^*)}\right) \frac{\partial y^*}{\partial w_i} \\
&= \left(\frac{y^* - y}{y^*(1-y^*)}\right) \frac{\partial \mathcal{A}\big(w_1 x_1 + w_D x_2 + \ldots + w_D x_D\big)}{\partial w_i} \\
&= \left(\frac{y^* - y}{y^*(1-y^*)}\right) \big(y^*(1-y^*)\big) x_i \\
&= (y^* - y) x_i
\end{aligned}$$

Backpropagation



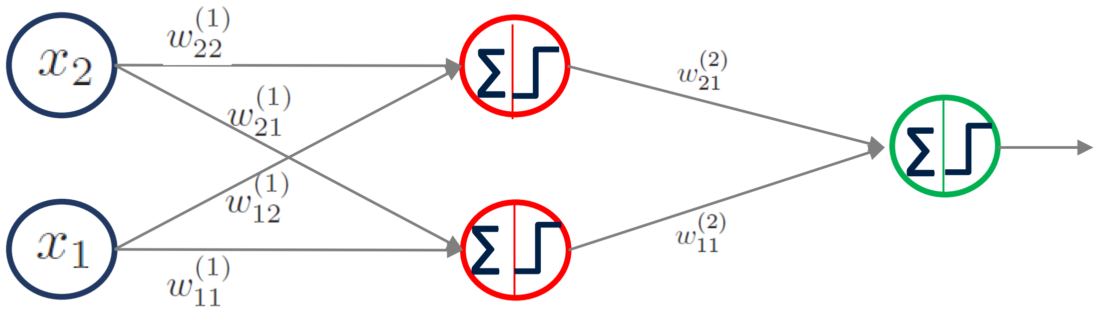
- Update: $$w_i := w_i - \xi \frac{\partial L}{\partial w_i}$$ $$:= w_i - \xi \left(y^* - y\right) x_i$$

- Vectorial notation:

  Let $\mathbf{x} = [x_1, x_2, ..., x_D]^{\mathrm{T}}$ and $\mathbf{w} = [w_1, w_2, ..., w_D]^{\mathrm{T}}$, then

$$\mathbf{w} := \mathbf{w} - \xi \left(y^* - y\right) \mathbf{x}$$

2 layers feedforward

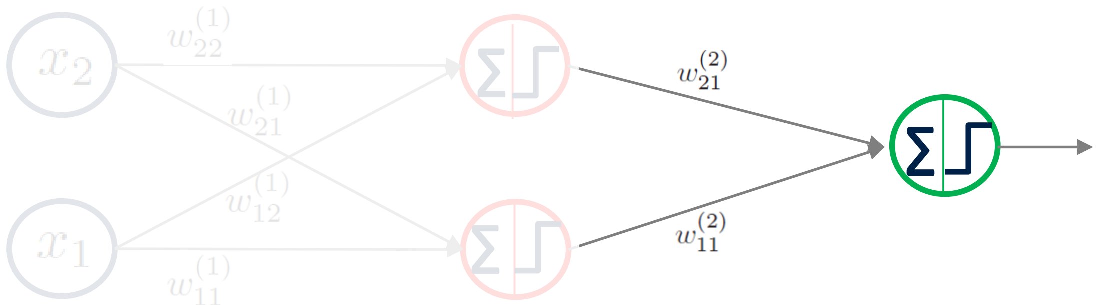


- Hidden layer outputs
  - $z_1 = \mathcal{A}\left(w_{11}^{(1)} x_1 + w_{12}^{(1)} x_2\right)$
  - $z_2 = \mathcal{A}\left(w_{21}^{(1)} x_1 + w_{22}^{(1)} x_2\right)$
- Final output: $y^* = \mathcal{A}\left(w_{11}^{(2)} z_1 + w_{21}^{(2)} z_2\right)$
- Activation function – Sigmoid

$$\mathcal{A}(z) = \frac{1}{1 + \exp(-z)}$$

- Loss function: $L = -y \log(y^*) - (1 - y) \log(1 - y^*)$

Looking backwards....



- Derivative of the loss function with respect to weights:

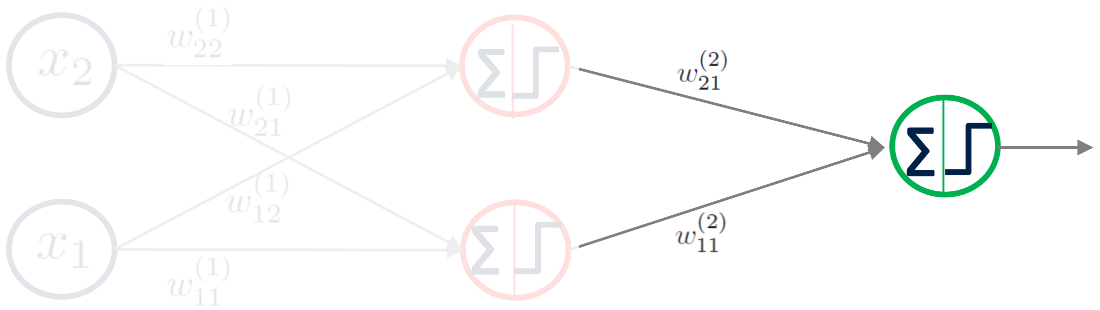
$$\frac{\partial L}{\partial w_{j1}^{(2)}} = \frac{\partial\big(-y\log(y^*) - (1-y)\log(1-y^*)\big)}{\partial w_{j1}^{(2)}}$$

$$= \left(-\frac{y}{y^*} + \frac{(1-y)}{(1-y^*)}\right)\frac{\partial\mathcal{A}\big(\sum_{j=1}^{2} w_{j1}^{(2)} z_j\big)}{\partial w_{j1}^{(2)}}$$

$$= \left(\frac{y^* - y}{y^*(1-y^*)}\right)\mathcal{A}'\big(\mathbf{v}_1^{(2)}\big)z_j \qquad \text{where } \mathbf{v}_1^{(2)} = \big(\mathbf{w}^{(2)}\big)^{\mathrm{T}}\mathbf{z}$$

$$= (y^* - y)z_j$$

Looking backwards: second layer

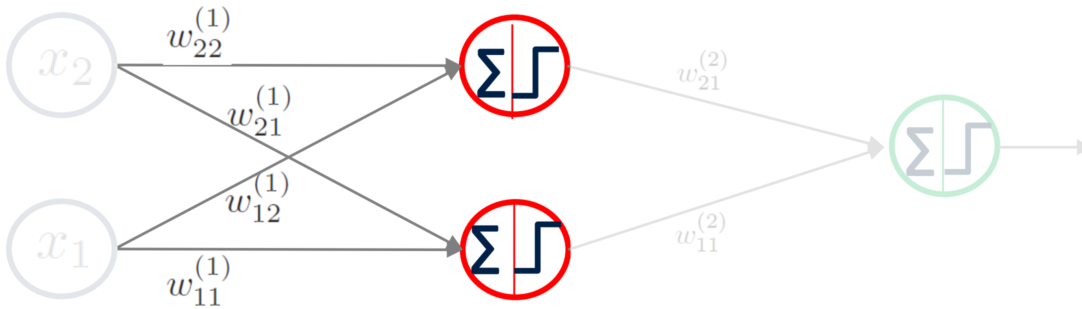


- Weight update at the second layer:

$$w_{j1}^{(2)} := w_{j1}^{(2)} - \xi \frac{\partial L}{\partial w_{j1}^{(2)}}$$

$$:= w_{j1}^{(2)} - \xi \Big(y^* - y \Big) z_j$$

Looking backwards: first layer



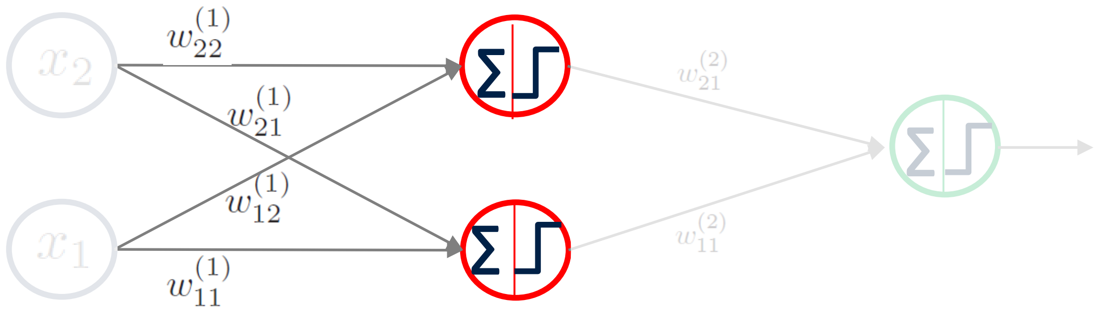
- Update of weights in the first layer:
  - Computing the gradients first

$$\frac{\partial L}{\partial w_{ji}^{(1)}} = \frac{\partial\big(- y \log(y^*) - (1-y)\log(1-y^*)\big)}{\partial w_{ji}^{(1)}}$$

$$= \Big(- \frac{y}{y^*} + \frac{(1-y)}{(1-y^*)}\Big) \frac{\partial}{\partial w_{ji}^{(1)}} \Big(\mathcal{A}\big(\sum_{k=1}^{2} w_{k1}^{(2)} z_k\big)\Big)$$

$$= \Big(\frac{y^* - y}{y^*(1-y^*)}\Big) \frac{\partial}{\partial w_{ji}^{(1)}} \Big(\mathcal{A}\big(\sum_{k=1}^{2} w_{k1}^{(2)} \mathcal{A}\big(\sum_{m=1}^{D} w_{mk}^{(1)} x_m\big)\big)\Big)$$

Looking backwards: first layer



- Update of weights in the first layer:
  - Computing the gradients first

$$\frac{\partial L}{\partial w_{ji}^{(1)}} = \left(\frac{y^* - y}{y^*(1 - y^*)}\right)\mathcal{A}'(\mathbf{v}_1^{(2)})w_{i1}^{(2)}\frac{\partial \mathcal{A}\left(\sum_{m=1}^{D} w_{mk}^{(1)} x_m\right)}{\partial w_{ji}^{(1)}}$$

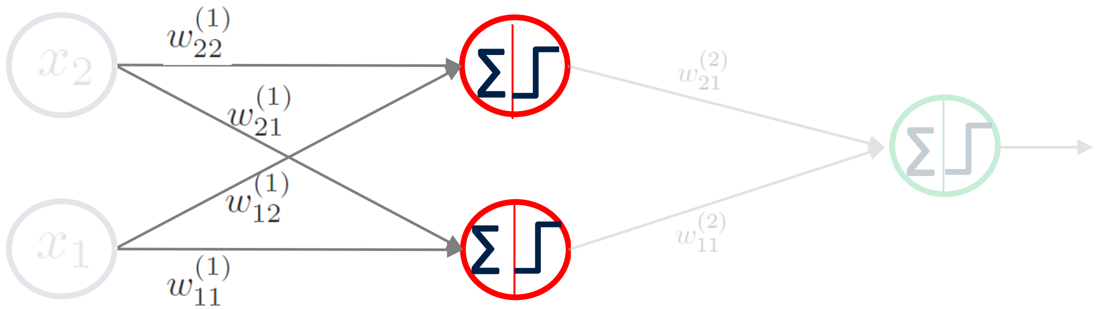
$$= \left(\frac{y^* - y}{y^*(1 - y^*)}\right)\mathcal{A}'(\mathbf{v}_1^{(2)})w_{i1}^{(2)}\mathcal{A}'(\mathbf{v}_i^{(1)})x_j$$

For sigmoid activation function

$$\mathcal{A}'(\mathbf{v}_1^{(2)}) = y^*(1 - y^*)$$

$$\mathcal{A}'(\mathbf{v}_i^{(1)}) = z_i(1 - z_i)$$

Looking backwards: first layer



- Update of weights in the first layer:
  - Computing the gradients first

$$\frac{\partial L}{\partial w_{ji}^{(1)}} = \Big(\frac{y^* - y}{y^*(1-y^*)}\Big)\mathcal{A}'(\mathbf{v}_1^{(2)})w_{i1}^{(2)}\frac{\partial \mathcal{A}(\sum_{m=1}^{D} w_{mk}^{(1)} x_m)}{\partial w_{ji}^{(1)}}$$

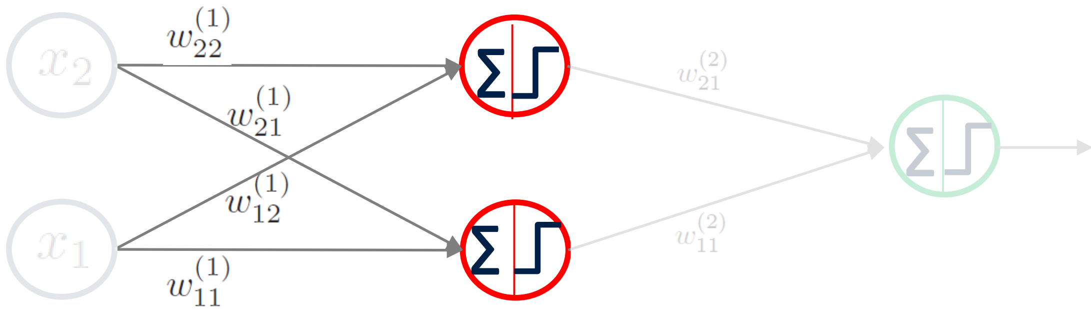
$$= \Big(\frac{y^* - y}{y^*(1-y^*)}\Big)\mathcal{A}'(\mathbf{v}_1^{(2)})w_{i1}^{(2)}\mathcal{A}'(\mathbf{v}_i^{(1)})x_j$$

On substitution we get

$$= \Big(\frac{y^* - y}{y^*(1-y^*)}\Big)y^*(1-y^*)w_{i1}^{(2)} z_i(1-z_i)x_j$$

$$= (y^* - y)z_i(1-z_i)w_{i1}^{(2)}x_j$$

First layer weight updates

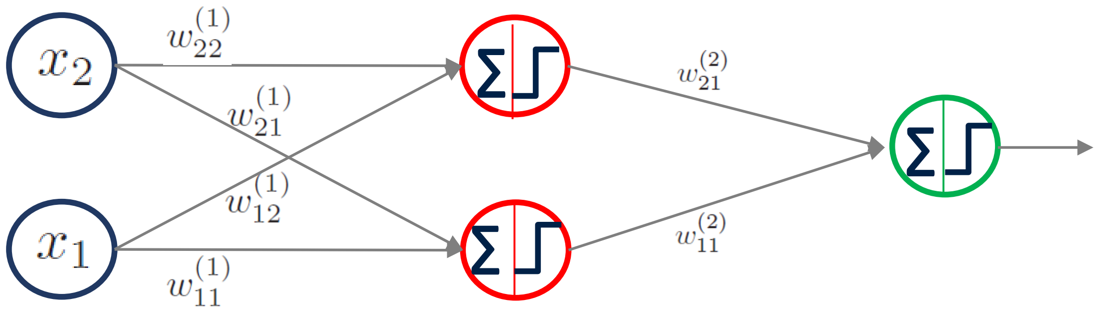


- Update of weights in the first layer:

$$w_{ji}^{(1)} := w_{ji}^{(1)} - \xi \frac{\partial L}{\partial w_{ji}^{(1)}}$$

$$:= w_{ji}^{(1)} - \xi w_{i1}^{(2)} \big(y^* - y \big) z_i (1 - z_i) x_j$$

Weight updates: compact representation



- Weight update in the second layer:

$$w_{j1}^{(2)} := w_{j1}^{(2)} - \xi \delta^{(2)} z_j$$

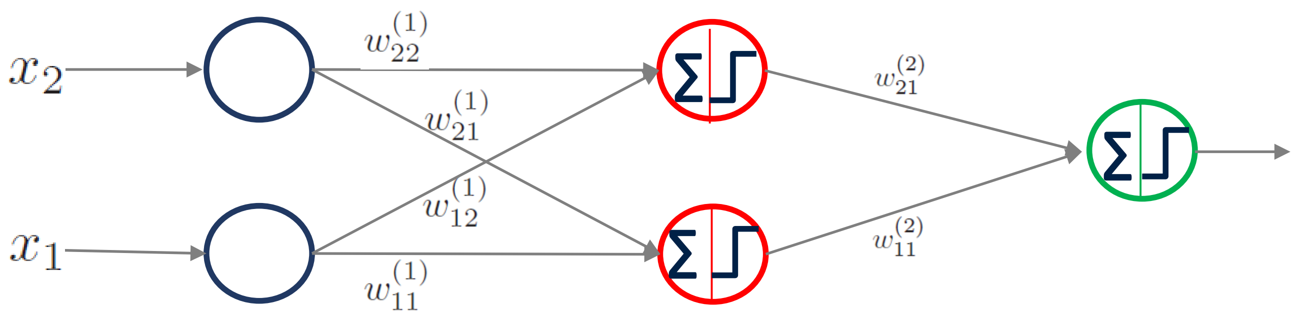
where $\delta^{(2)} = (y^* - y)$.

- Weight update in the first layer:

$$w_{ji}^{(1)} := w_{ji}^{(1)} - \xi \delta_i^{(1)} x_j$$

where $\delta_i^{(1)} = \delta^{(2)} w_{i1}^{(2)} z_i (1 - z_i)$.

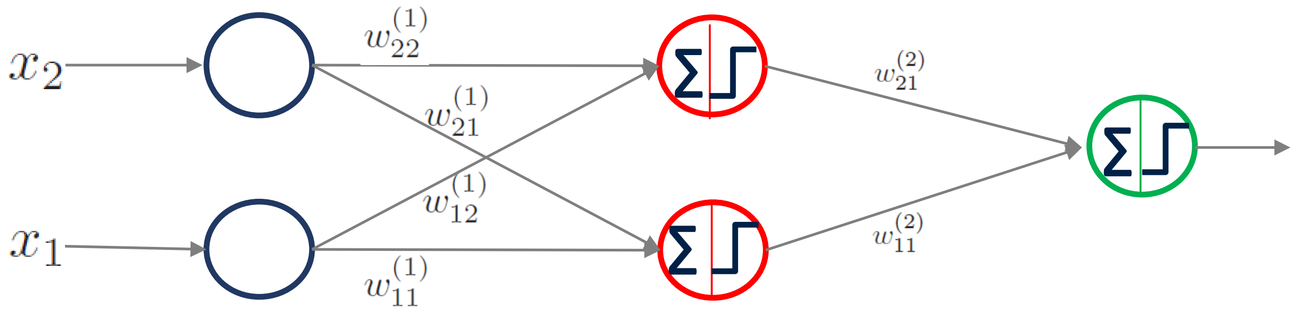
Representation in vector form



- The weights in each layer can be represented in vector form.

$$\mathbf{w}^{(1)} = \begin{bmatrix} w^{(1)}_{11} & w^{(1)}_{12} & . & . & w^{(1)}_{1H} \\ w^{(1)}_{21} & w^{(1)}_{22} & . & . & w^{(1)}_{2H} \\ . & . & . & . & . \\ . & . & . & . & . \\ w^{(1)}_{D1} & w^{(1)}_{D2} & . & . & w^{(1)}_{DH} \end{bmatrix} \qquad \mathbf{w}^{(2)} = \begin{bmatrix} w^{(2)}_{11} \\ w^{(2)}_{21} \\ . \\ . \\ w^{(2)}_{H1} \end{bmatrix}$$

Representation in vector form



- Vectorial representation of inputs:

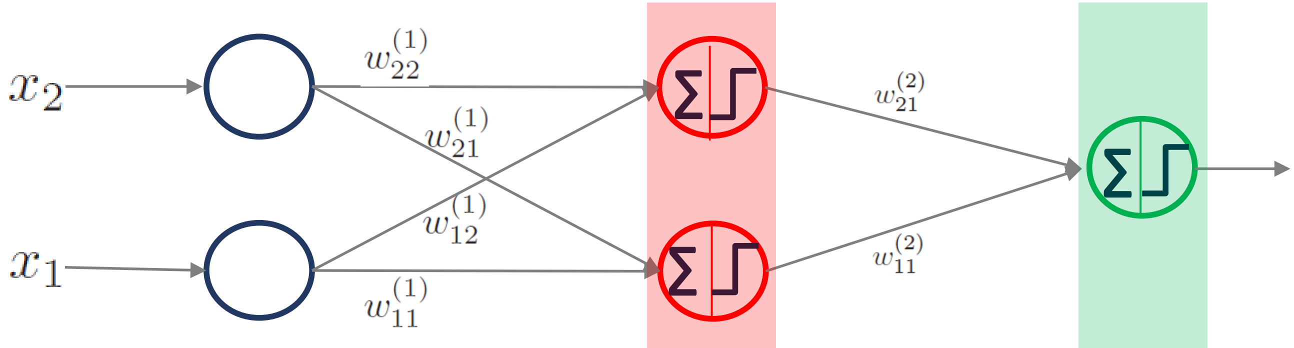
Inputs to Layer 1

$$\mathbf{u}^{(1)} = \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ x_D \end{bmatrix}$$

Inputs to Layer 2

$$\mathbf{u}^{(2)} = \begin{bmatrix} z_1 \\ z_2 \\ \cdot \\ \cdot \\ z_H \end{bmatrix}$$

Going forward....

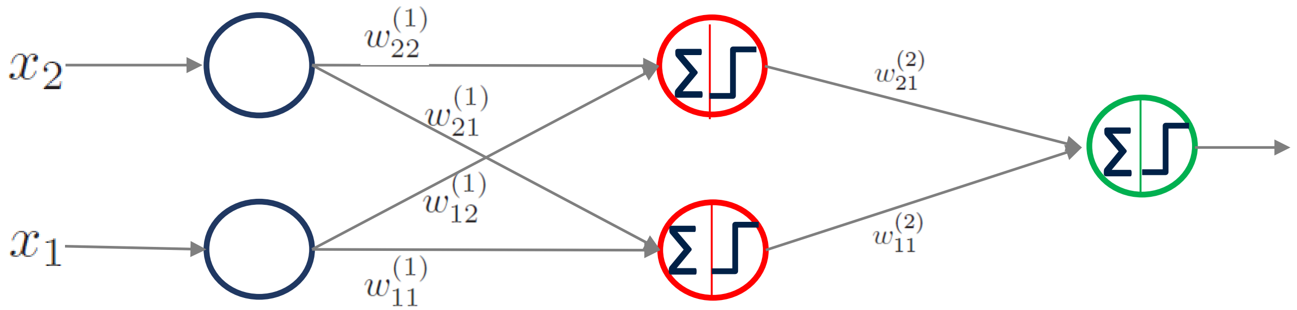


- First layer, first stage: $\mathbf{I}^{(1)} = \left(\mathbf{w}^{(1)}\right)^{\mathrm{T}} \mathbf{u}^{(1)}$
- First layer, second stage: $\mathbf{O}^{(1)} = \mathcal{A}(\mathbf{I}^{(1)}) = \mathbf{u}^{(2)}$

- Output layer, first stage: $\mathbf{I}^{(2)} = \left(\mathbf{w}^{(2)}\right)^{\mathrm{T}} \mathbf{u}^{(2)}$
- Output layer, second stage: $\mathbf{O}^{(2)} = \mathcal{A}(\mathbf{I}^{(2)}) = y^*$

- Loss: $L = -y \log(y^*) - (1-y) \log(1-y^*)$

Going backward....



- Weight update in the second layer:

$$\mathbf{w}^{(2)} = \mathbf{w}^{(2)} - \xi \delta^{(2)} \mathbf{u}^{(2)}$$

where $\delta^{(2)} = (y^* - y)$

- Weight update in the first layer:

$$\mathbf{w}^{(1)} = \mathbf{w}^{(1)} - \xi \delta^{(1)} \mathbf{u}^{(1)}$$

where $\delta^{(1)} = \left(\delta^{(2)} \left(\mathbf{w}^{(2)}\right)^{\mathrm{T}}\right) \odot \mathcal{Z}$ with $\mathcal{Z} = [z_1(1 - z_1), \ldots\ldots, z_H(1 - z_H)]$

and $\odot$ is the Hadamard product.