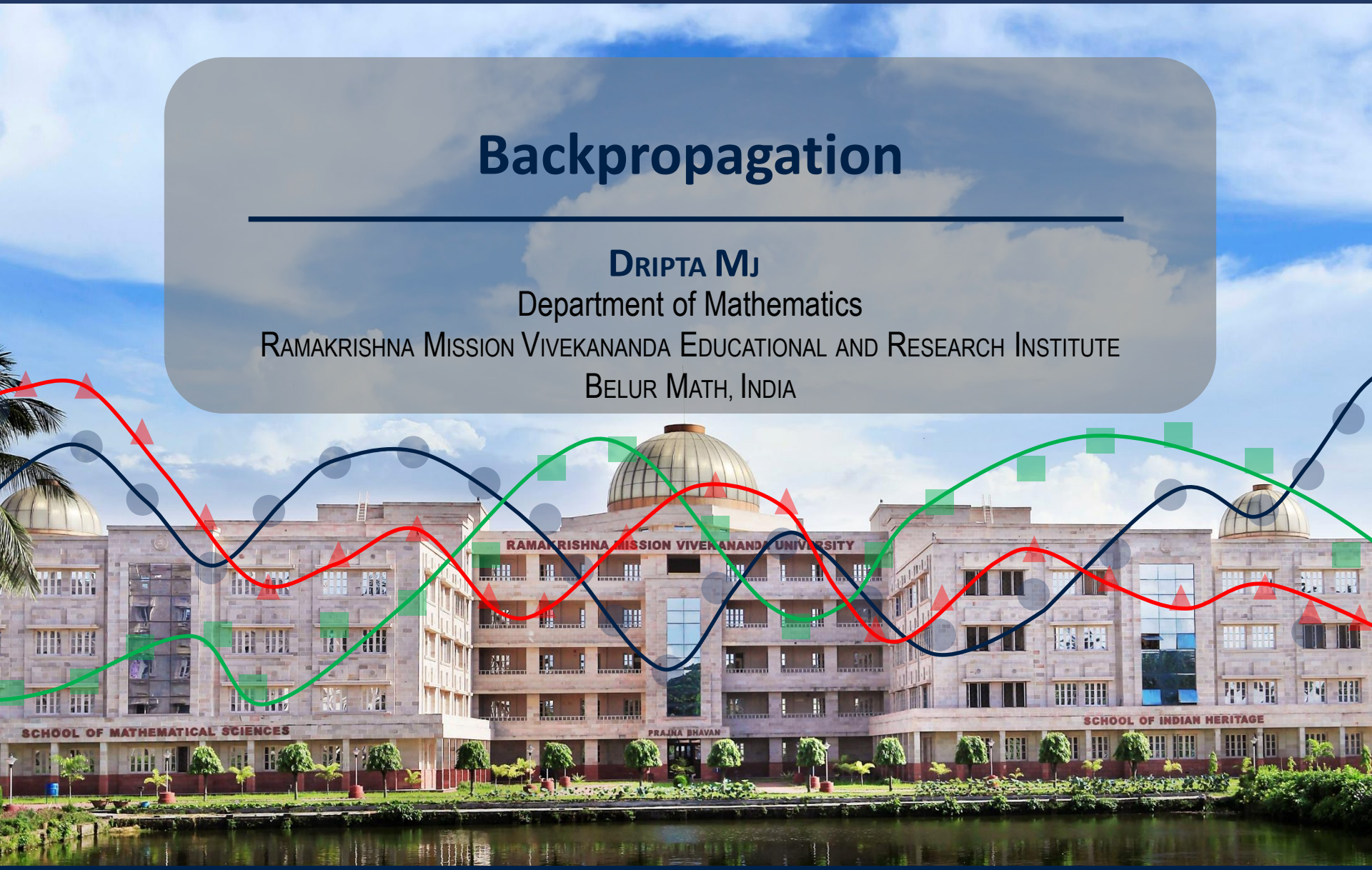
Backpropagation

Dripta Mj
Department of Mathematics
Ramakrishna Mission Vivekananda Educational and Research Institute
Belur Math, India



Training a neural network

- Goal – Optimize for weights:

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

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

- Select an appropriate loss function:
  - Squared loss:

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

  - Binary cross-entropy loss:

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

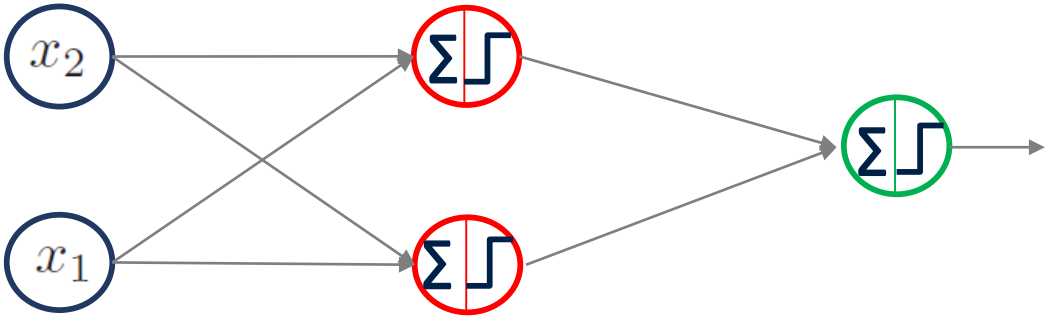
  - Cross-entropy loss:

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

- Gradient descent:

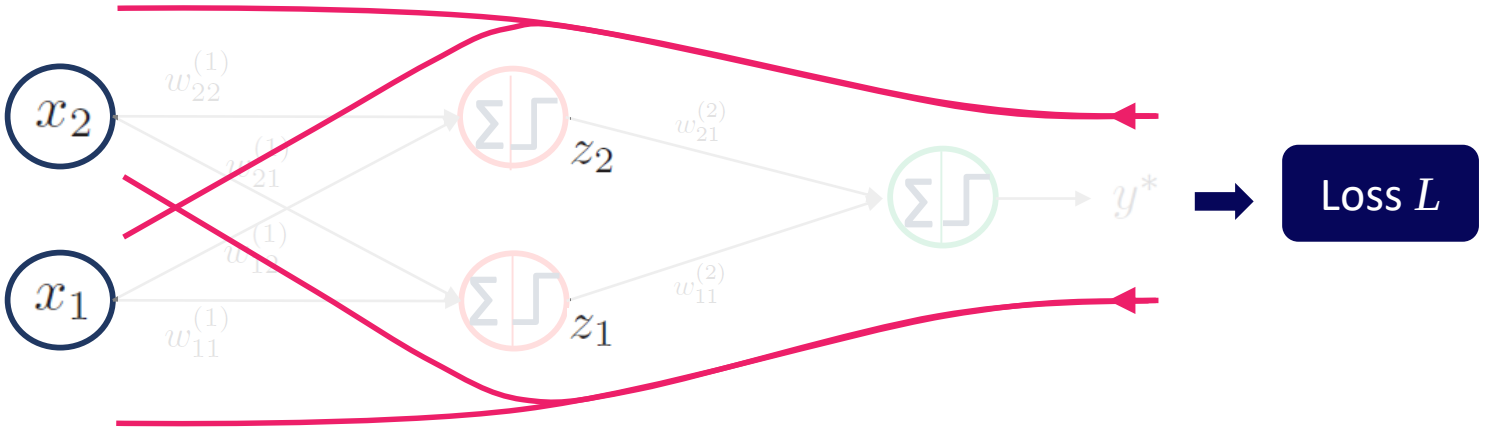
$$\mathbf{w}^{t+1} = \mathbf{w}^t - \xi \nabla_{\mathbf{w}^t} L$$

Procedure

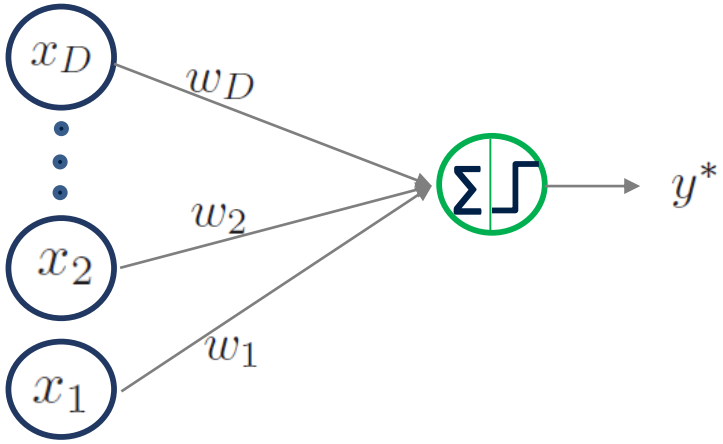


- Training is achieved in two steps:
  - Step 1: Forward pass the inputs through the network.
  - Step 2: In order to adjust the parameters we go backwards. Parameters are updated using gradients.

Forward → Backward ← ….



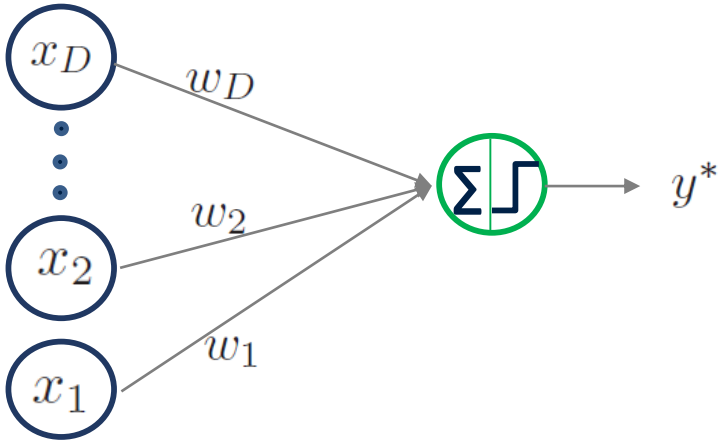
Single layer network



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

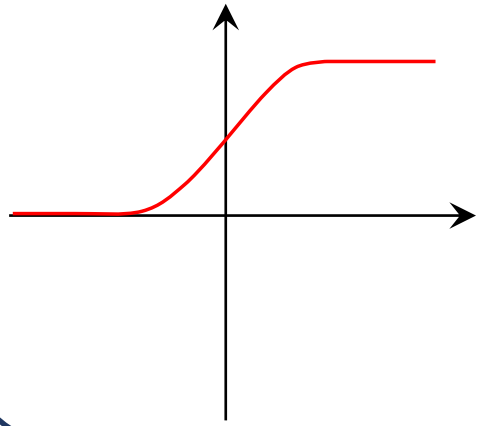
$$L = -y \log(y^*) - (1-y) \log(1-y^*)$$

Activation function



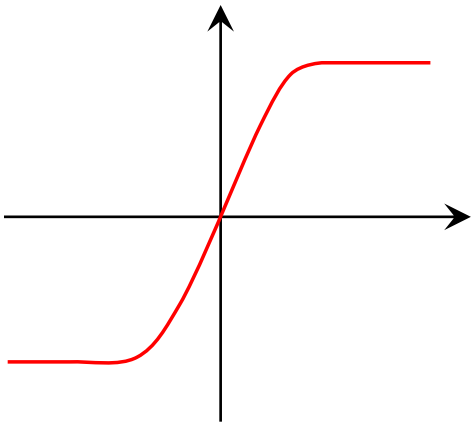
Sigmoid

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



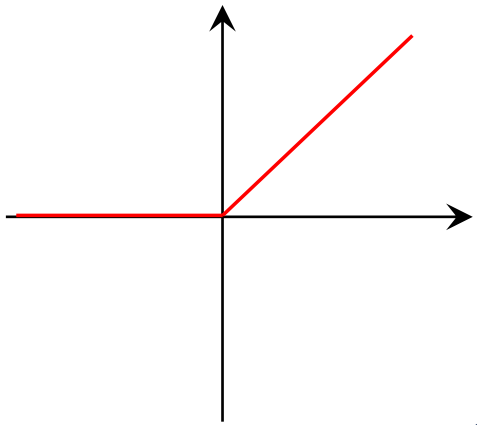
tanh

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

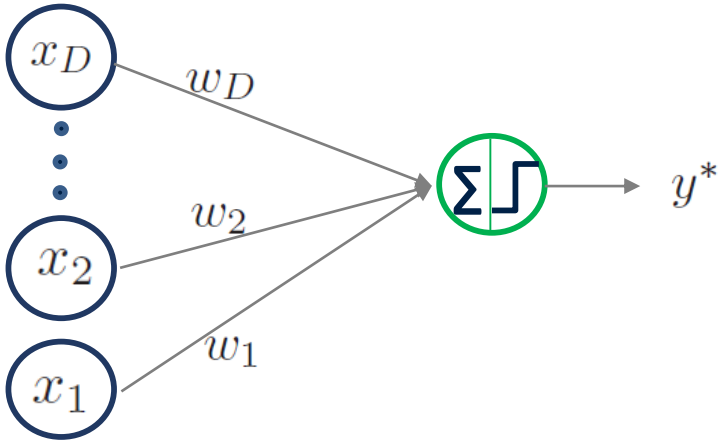


ReLU

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



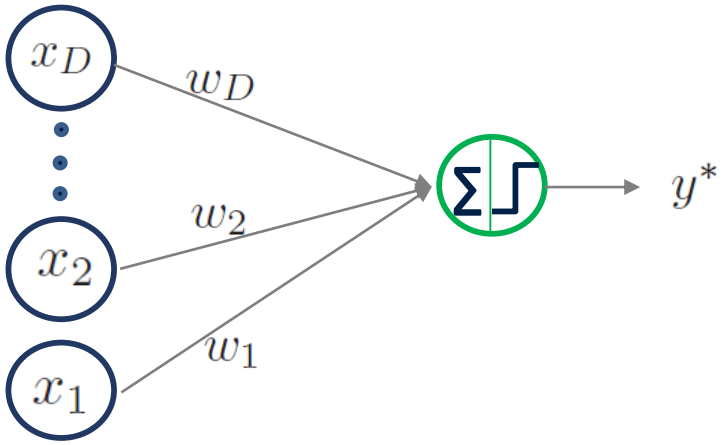
Single layer network – weight update



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

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

Single layer network – weight update



- Update: $w_i := w_i - \xi \dfrac{\partial L}{\partial w_i}$

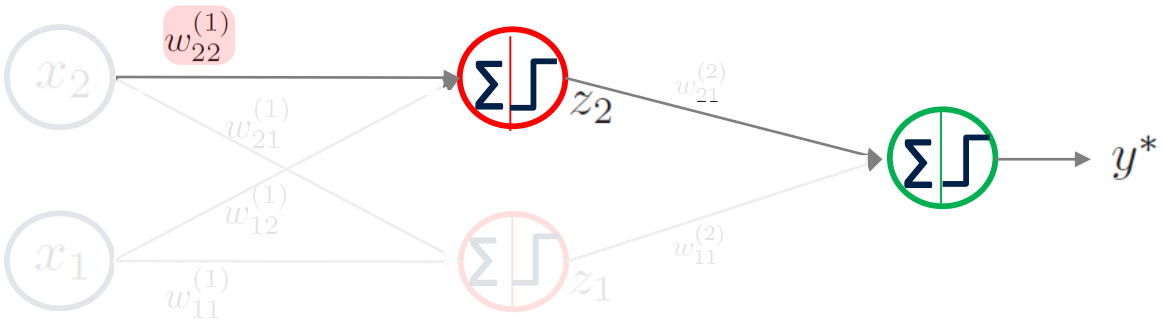
  $:= w_i - \xi(y^* - y)x_i$

- 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

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

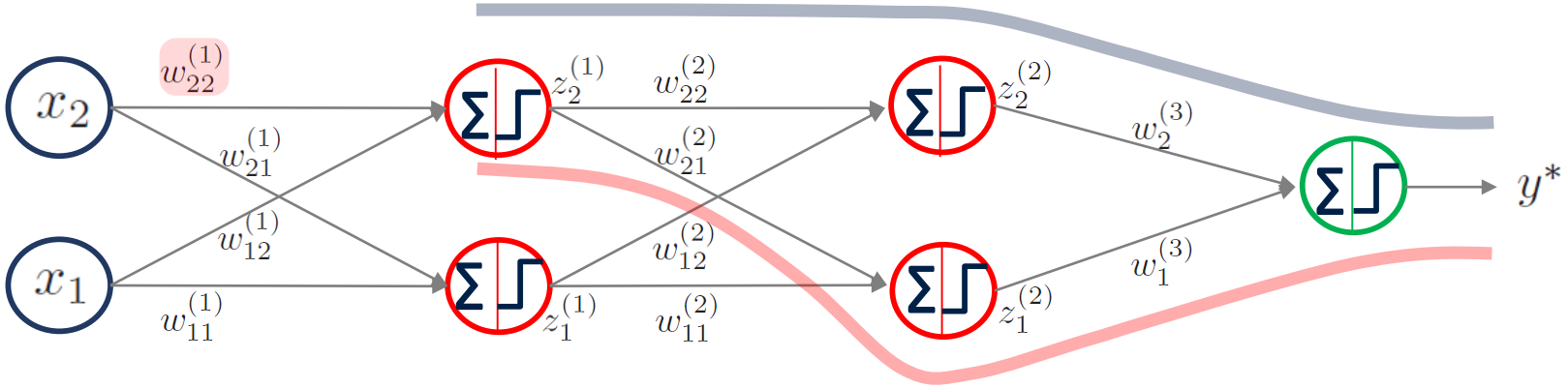
2 layer network



- Gradient of the loss function $L(y, y^*)$ w.r.t. $w_{22}^{(1)}$:

$$\frac{\partial L(y^*, y)}{\partial w_{22}^{(1)}} = \frac{\partial L}{\partial y^*} \frac{\partial y^*}{\partial z_2} \frac{\partial z_2}{\partial w_{22}^{(1)}}$$

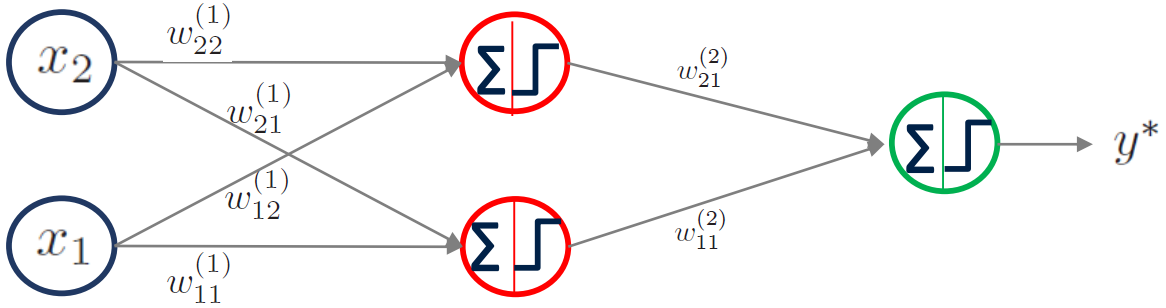
3 layer network



- Gradient of the loss function $L(y, y^*)$ w.r.t. $w_{22}^{(1)}$:

$$\frac{\partial L(y^*, y)}{\partial w_{22}^{(1)}} = \underbrace{\frac{\partial L}{\partial y^*} \frac{\partial y^*}{\partial z_2^{(2)}} \frac{\partial z_2^{(2)}}{\partial z_2^{(1)}} \frac{\partial z_2^{(1)}}{\partial w_{22}^{(1)}}}_{} + \underbrace{\frac{\partial L}{\partial y^*} \frac{\partial y^*}{\partial z_1^{(2)}} \frac{\partial z_1^{(2)}}{\partial z_1^{(1)}} \frac{\partial z_2^{(1)}}{\partial w_{22}^{(1)}}}_{}$$

2 layer network with sigmoid activation

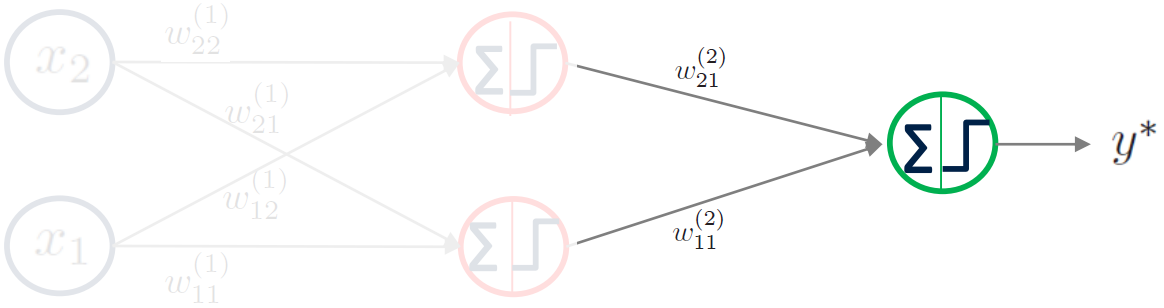


- Hidden layer outputs
  - $z_1 = \mathcal{A}\left(w_{11}^{(1)} x_1 + w_{21}^{(1)} x_2\right)$
  - $z_2 = \mathcal{A}\left(w_{12}^{(1)} x_1 + w_{22}^{(1)} x_2\right)$
- Final output: $y^* = \mathcal{A}\left(w_{11}^{(2)} z_1 + w_{21}^{(2)} z_2\right)$
- Activation function – Sigmoid

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

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

Looking backwards….



- Derivative of the loss function with respect to weights:

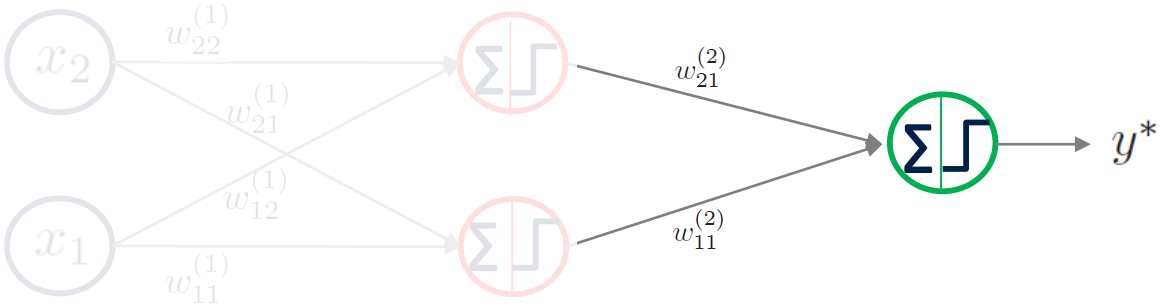
$$\frac{\partial L}{\partial w_{j1}^{(2)}} = \frac{\partial\big(-y\log(y^*)-(1-y)\log(1-y^*)\big)}{\partial w_{j1}^{(2)}}$$

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

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

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

Looking backwards: second layer

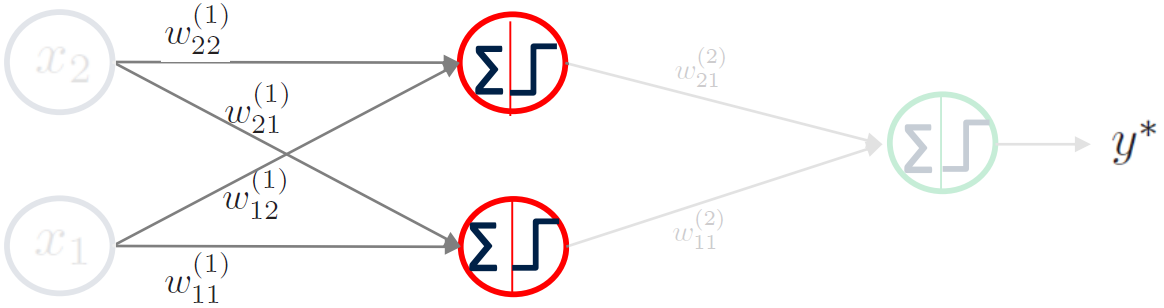


- Weight update at the second layer:

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

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

Looking backwards: first layer



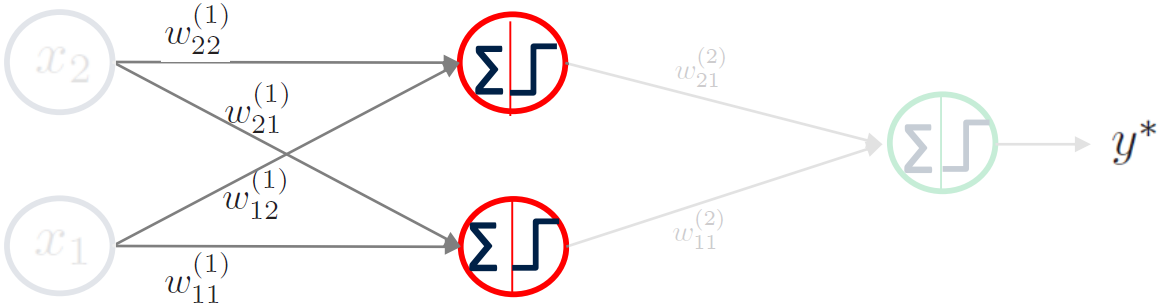
- Update of weights in the first layer:
  - Computing the gradients first

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

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

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

Looking backwards: first layer



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

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

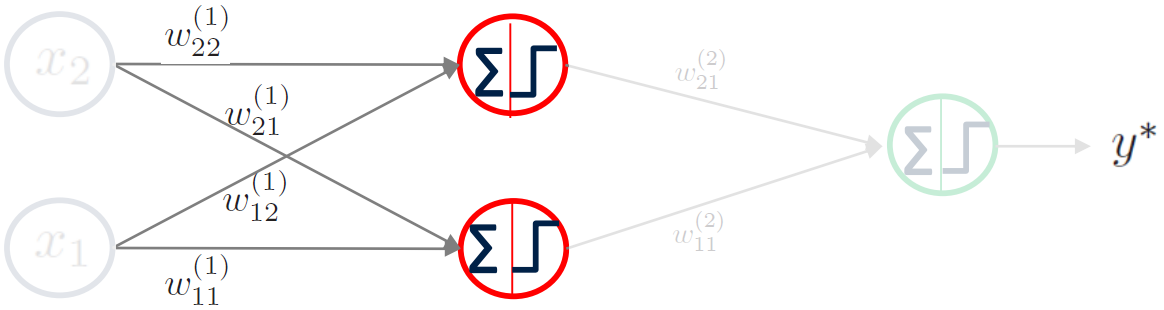
$$= \left(\frac{y^* - y}{y^*(1-y^*)}\right) \mathcal{A}'\left(\mathbf{v}_1^{(2)}\right) w_{i1}^{(2)} \mathcal{A}'\left(\mathbf{v}_i^{(1)}\right) x_j$$

For sigmoid activation function

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

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

Looking backwards: first layer



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

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

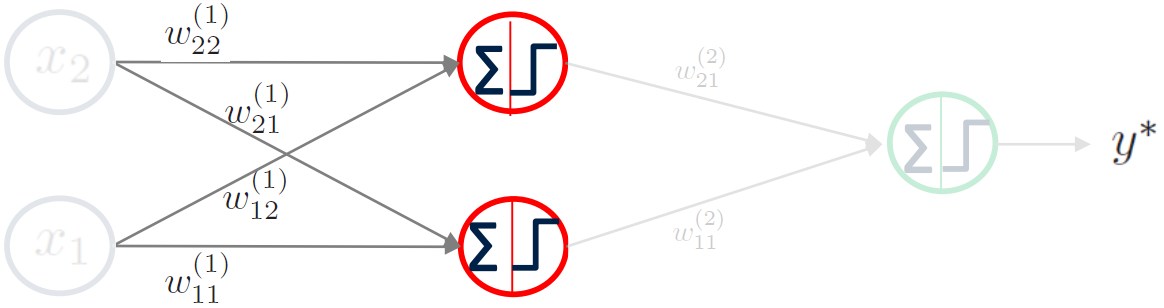
$$= \left(\frac{y^* - y}{y^*(1-y^*)}\right)\mathcal{A}'(\mathbf{v}_1^{(2)})w_{i1}^{(2)}\mathcal{A}'(\mathbf{v}_i^{(1)})x_j$$

On substitution we get

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

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

First layer weight updates

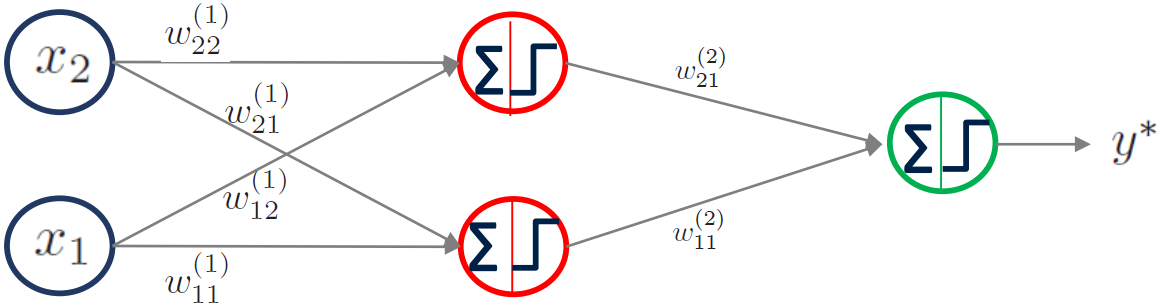


- Update of weights in the first layer:

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

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

Weight updates: compact representation



- Weight update in the second layer:

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

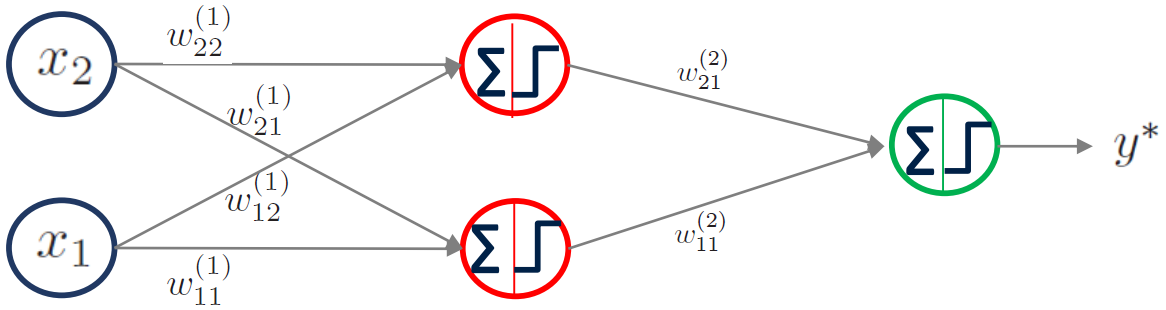
where $\delta^{(2)} = (y^* - y)$.

- Weight update in the first layer:

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

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

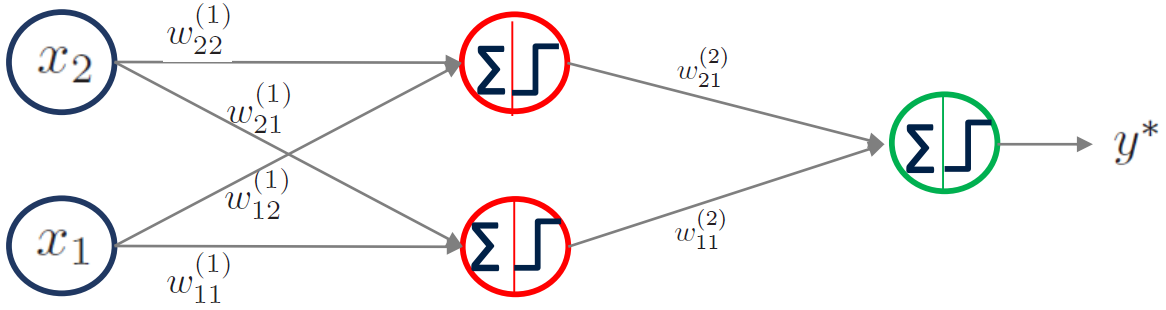
Representation in vector form



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

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

Representation in vector form



- Vectorial representation of inputs:

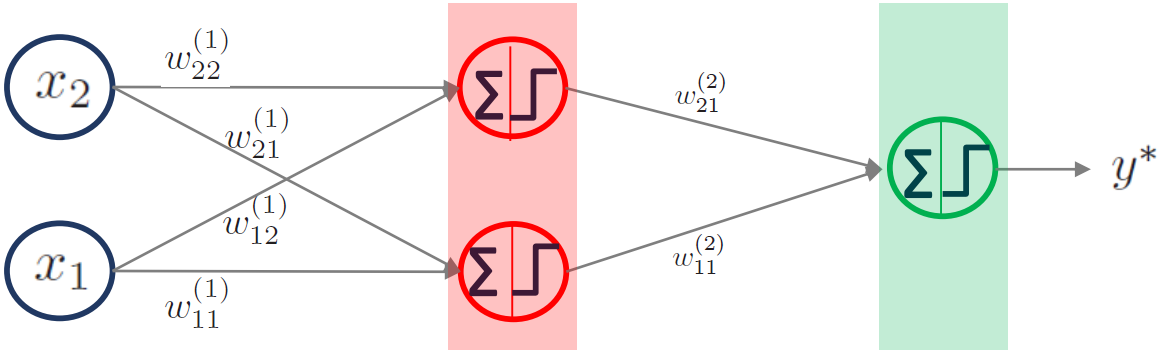
Inputs to Layer 1

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

Inputs to Layer 2

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

Going forward....

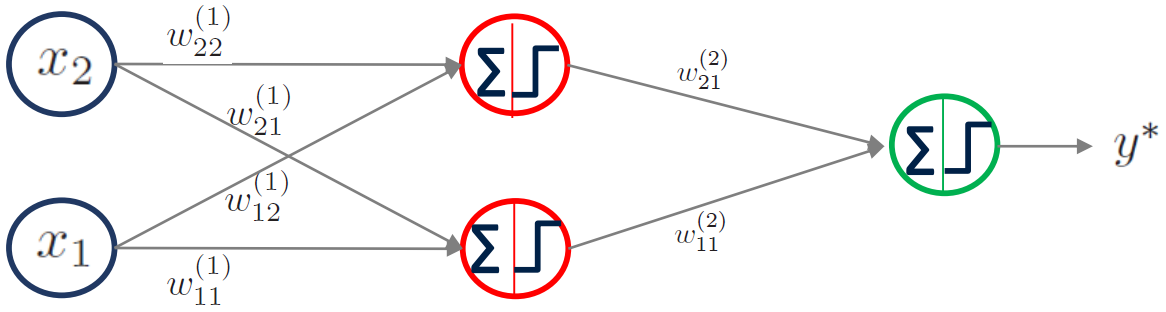


- First layer, first stage: $\mathbf{I}^{(1)} = \left(\mathbf{w}^{(1)}\right)^{\mathrm{T}} \mathbf{u}^{(1)}$
- First layer, second stage: $\mathbf{O}^{(1)} = \mathcal{A}(\mathbf{I}^{(1)}) = \mathbf{u}^{(2)}$

- 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:

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

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

- Weight update in the first layer:

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

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

and $\odot$ is the Hadamard product.