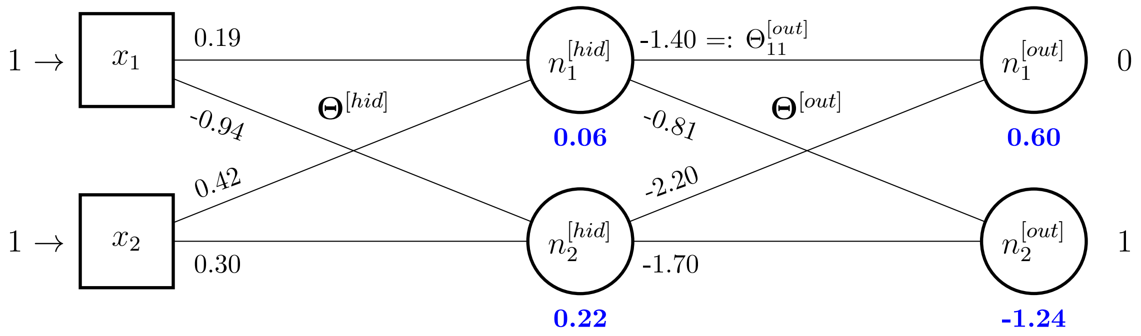
W3WI DS304.1 Applied Machine Learning Fundamentals

Backpropagation Formulas

Input to the network: $\boldsymbol{x} := (1,1)^\intercal$. Desired output: $\boldsymbol{y} := (0,1)^\intercal$. The network is depicted in the following figure:



The following table lists the preactivations and activations of all neurons (result of the forward pass through the network):

Neuron	Preactivation p	Activation z	Activation function $g(\cdot)$
$n_1^{[hid]}$	0.61	0.61	ReLU
$n_2^{[hid]}$	-0.64	0.00	ReLU
$n_1^{[out]}$	-0.85	0.30	Sigmoid
$n_2^{[out]}$	-0.49	0.38	Sigmoid

Step 1) Computation of the error gradient in the output layer:

$$\frac{\partial \mathcal{J}}{\partial z_{n_k^{[out]}}} := 2 \cdot \Big(\overbrace{g(p_{n_k^{[out]}})}^{=z_{n_k^{[out]}}} - y_k\Big) \tag{1}$$

Example for neuron $n_1^{[out]}$: (see blue number below neuron in figure above)

$$\frac{\partial \mathcal{J}}{\partial n_1^{[out]}} = 2 \cdot (0.30 - 0) = 0.60$$

Step 2) Computation of the error gradient in the hidden layer:

$$\frac{\partial \mathcal{J}}{\partial z_{n_t^{[hid]}}} := \sum_k \overbrace{\frac{\partial \mathcal{J}}{\partial z_{n_k^{[out]}}}}^{\text{see } \mathbf{1)}} \cdot g'\left(p_{n_k^{[out]}}\right) \cdot \Theta_{kt}^{[out]} \tag{2}$$

Example for neuron $n_1^{[hid]}$: (see blue number below neuron in figure above)

$$\frac{\partial \mathcal{J}}{\partial n_1^{[hid]}} = \left[0.60 \cdot \overbrace{g(-0.85) \cdot (1 - g(-0.85))}^{\text{Derivative Sigmoid}} \cdot (-1.4)\right] + \left[-1.24 \cdot g(-0.49) \cdot (1 - 0.49) \cdot (-0.81)\right]$$
$$= 0.06$$

Step 3) Computation of the weight gradient in the output layer:

$$\frac{\partial \mathcal{J}}{\partial \Theta_{kt}^{[out]}} := \overbrace{\frac{\partial \mathcal{J}}{\partial z_{n_k^{[out]}}}}^{\text{see } \mathbf{1)}} \cdot g'(p_{n_k^{[out]}}) \cdot g(p_{n_t^{[hid]}}) \tag{3}$$

Example for weight $\Theta_{11}^{[out]}$:

$$\frac{\partial \mathcal{J}}{\partial \Theta_{11}^{[out]}} = 0.60 \cdot \overbrace{g(-0.85) \cdot (1 - g(-0.85))}^{\text{Derivative Sigmoid}} \cdot g(0.61)$$
$$= 0.077$$

Step 4) Computation of the weight gradient in the hidden layer:

The weight gradient in the hidden layer is computed analogously to step 3. However, we use the input to the network instead of $g(p_{n_t^{[hid]}})$.

Step 5) Update the network parameters:

The parameters are updated according to the gradient descent update rule.

Example for weight $\Theta_{11}^{[out]}$ with learning rate $\alpha := 0.1$:

$$\begin{aligned}
\Theta_{11}^{[out]} &\longleftarrow \Theta_{11}^{[out]} - \alpha \cdot \overbrace{\frac{\partial \mathcal{J}}{\partial \Theta_{11}^{[out]}}}^{\text{see } \mathbf{3)}} \\
&\longleftarrow -1.40 - 0.1 \cdot 0.077 \\
&\longleftarrow -1.4077
\end{aligned}$$