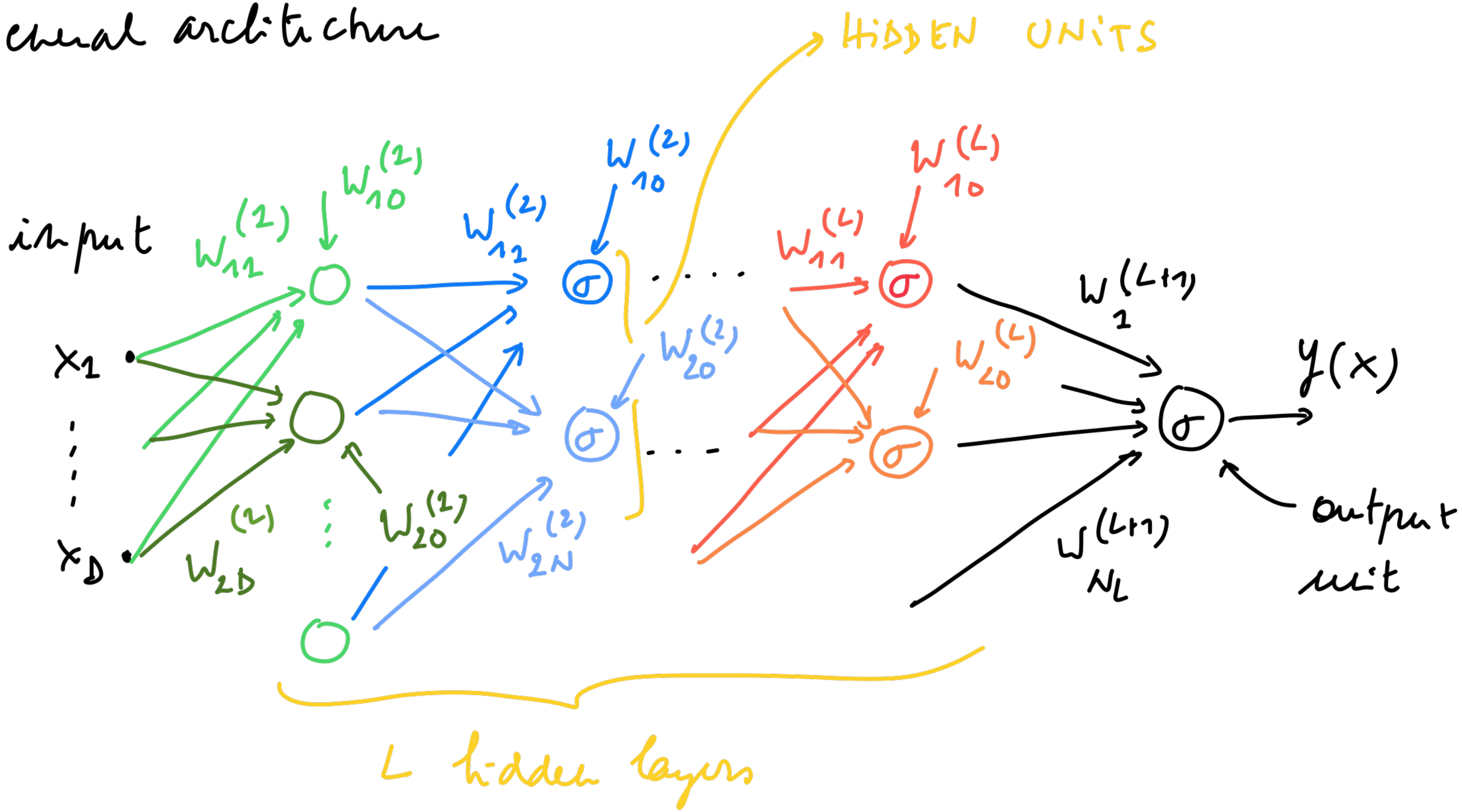
Neural Networks

→ Reminders (forward propagation, architecture)

→ Derivation of gradient through backpropagation

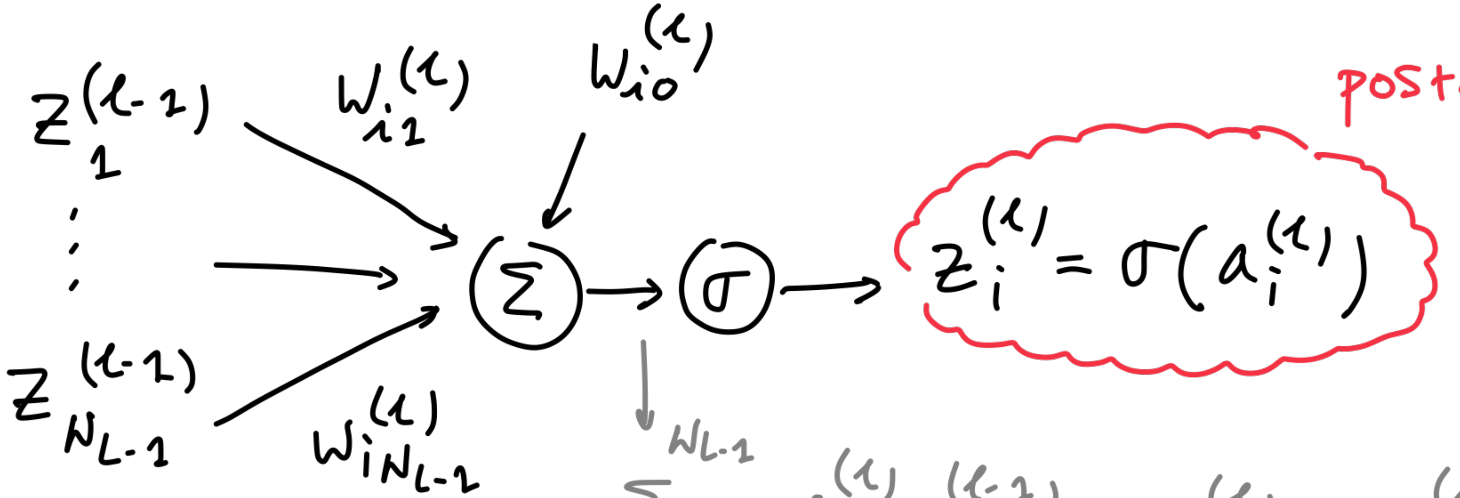
→ Recitation (implementation in numpy)

General architecture



$W_{ij}^{(l)}$ is the weight in the i^{th} neuron from layer l that multiplies the j^{th} input to this neuron

Zooming on a particular neuron i from layer l, we get



$$\sum_{j=1}^{N_{L-1}} W_{ij}^{(l)} z_j^{(l-1)} + W_{i0}^{(l)} = \underbrace{a_i^{(l)}}_{\text{preactivation}}$$

(whatever is passed to the activation function

Using our network to encode class probabilities,

$$p(t^{(i)}=1 \mid x^{(i)}) = y(x^{(i)}; w)$$
$$p(t^{(i)}=0 \mid x^{(i)}) = 1 - y(x^{(i)}; w)$$

→ We can then train the network via a Maximum likelihood approach, corresponding to a minimization of the log loss.

$$\mathcal{L}(w) = - \sum_{i=1}^{N} t^{(i)} \log(y(x^{(i)}; w)) + (1-t^{(i)}) \log(1-y(x^{(i)}; w))$$

→ in particular, if we use SGD, optimizing one sample at a time, the loss reduces to

$$\mathcal{L}(w) = - t^{(i)} \log(y(x^{(i)}; w)) - (1-t^{(i)}) \log(1-y(x^{(i)}; w))$$

To implement the training of the network through SGD, we need the derivatives $\frac{\partial \mathcal{L}}{\partial w_{ij}^{(\ell)}}$

$\rightarrow$ use the fact that $a_i^{(\ell)} = \sum_{j=1}^{N_L} w_{ij}^{(\ell)} z_j^{(\ell-1)} + w_{i0}^{(\ell)}$

$\rightarrow$ From this $\frac{\partial \mathcal{L}}{\partial w_{ij}^{(\ell)}} = \frac{\partial \mathcal{L}}{\partial a_i^{(\ell)}} \cdot \frac{\partial a_i^{(\ell)}}{\partial w_{ij}^{(\ell)}} = \frac{\partial \mathcal{L}}{\partial a_i^{(\ell)}} \cdot z_j^{(\ell-1)}$

We will now use $\delta_i^{(\ell)} = \frac{\partial \mathcal{L}}{\partial a_i^{(\ell)}}$

From this we get $\frac{\partial \mathcal{L}}{\partial w_{ij}^{(\ell)}} = \delta_i^{(\ell)} \, z_j^{(\ell-1)}$

$z_j \rightarrow$ can be derived through forward propagation

$\delta_i^{(l)} \rightarrow$ can be derived through back propagation

Step 1 $\quad \delta^{out} = \dfrac{\partial \mathcal{L}}{\partial a_{out}} \rightarrow$ can be obtained easily

Note that $\quad \dfrac{\partial \mathcal{L}}{\partial a_{out}} = -\dfrac{\partial}{\partial a_{out}}\left(t^{(i)} \log(\sigma(a_{out})) + (1-t^{(i)}) \log(1-\sigma(a_{out}))\right)$

$$\rightarrow \frac{\partial \mathcal{L}}{\partial a_{out}} = -t^{(i)} \frac{\sigma'(a_{out})}{\sigma(a_{out})} + -(1-t^{(i)}) \frac{(-\sigma'(a_{out}))}{1-\sigma(a_{out})} \quad (*)$$

Recall that $\quad \sigma'(x) = \sigma(x)(1-\sigma(x))$

Substituting this in $(*)$

$$\delta^{out} = \frac{\partial \mathcal{L}}{\partial a_{out}} = -t^{(i)}(1-\sigma(a_{out})) + (1-t^{(i)})\,\sigma(a_{out})$$

$$= \sigma(a_{out}) - t^{(i)}$$

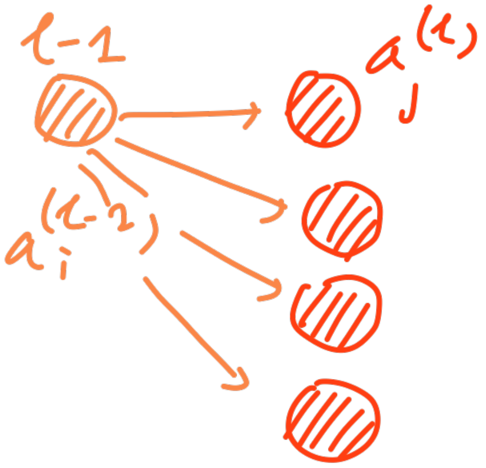
$$\delta^{out} = y(x^{(i)}; w) - t^{(i)}$$

→ Question? Given δ^{out}, how can we derive $\delta_i^{(\ell)}$ for $\ell < L$?

Using the chain rule and the fact that the preactivation $a_i^{(\ell-1)}$ is "fed" to all the neurons at layer ℓ, we can write

$$\underbrace{\frac{\partial \mathcal{L}}{\partial a_i^{(\ell-1)}}}_{\delta_i^{(\ell-1)}} = \sum_{j=1}^{N_L} \underbrace{\frac{\partial \mathcal{L}}{\partial a_j^{(\ell)}}}_{\delta_j^{(\ell)}} \frac{\partial a_j^{(\ell)}}{\partial a_i^{(\ell-1)}}$$

$$\delta_i^{(\ell-1)} = \sum_{j=1}^{N_L} \delta_j^{(\ell)} \frac{\partial a_j^{(\ell)}}{\partial a_i^{(\ell-1)}}$$



$$a_j^{(\ell)} = \sum_{i=1}^{N_{L-1}} w_{ji}^{(\ell)} z_i^{(\ell-1)} + w_{j0}^{(\ell)}$$

$$a_j^{(\ell)} = \sum_{i=1}^{N_{L-1}} w_{ji}^{(\ell)} \sigma(a_i^{(\ell-1)}) + w_{j0}^{(\ell)}$$

$$\delta_i^{(\ell-1)} = \sum_{j=1}^{N_L} \delta_j^{(\ell)} w_{ji}^{(\ell)} \sigma'(a_i^{(\ell-1)})$$

$$\boxed{\delta_i^{(\ell-1)} = \sigma'(a_i^{(\ell-1)}) \sum_{j=1}^{N_L} \delta_j^{(\ell)} w_{ji}^{(\ell)}}$$

BACK PROPAGATION OF THE $\delta_i^{(\ell)}$

→ Derivatives w.r.t intercepts $w_{i0}^{(l)}$ can be obtained similarly by augmenting the post activation vector z with a 1 (i.e generating $\tilde{z}$ as in regression and classification)

→ GENERAL BACKPROPAGATION ALGORITHM

Step 1 Forward propagate input $x^{(i)}$ through the network and derive all pre and post activations $a_i^{(l)}, z_i^{(l)}$

Step 2 Compute δ^{out}

Step 3 Backpropagate the δ^{out} to derive all $\delta_i^{(l)}$

Step 4 Get the gradient as $\dfrac{\partial \mathcal{L}}{\partial w_{ij}^{(l)}} = \delta_i^{(l)} z_j^{(l-1)}$

Step 5 apply the update

$$w_{ij}^{(\ell)} \leftarrow w_{ij}^{(\ell)} - \eta \frac{\partial \mathcal{L}}{\partial w_{ij}^{(\ell)}}$$