

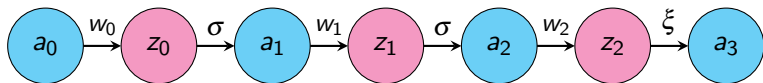
# Deep Learning E1394

Chiara Fusar Bassini

October 9, 2023

## A FNN example

Let  $\sigma$  be the activation function,  $\xi$  be the output function. Then we can represent our FNN using the following flowchart.



Note that at each step  $\{a_i\}_{i=0}^3$  might have different dimensions and that  $a_0 = x$ .

## Forward step

Let  $x$  be the input vector,  $\sigma$  the sigmoid activation function and  $\xi$  the soft-max output function. Then we can see the output of our feed-forward network as:

$$\begin{aligned} f(x) &= a_3 = \xi(w_2 a_2) \\ &= \xi(w_2 \sigma(w_1 a_1)) \\ &= \xi\left(w_2 \sigma(w_1 \sigma(w_0 a_0))\right) = f(a_0) \end{aligned}$$

## Forward step

Let  $x$  be the input vector. Then we can see our feed-forward network as:

$$f(x) = a_3 = \xi(\underbrace{w_2 a_2}_{z_2})$$

$$= \xi(w_2 \sigma(\underbrace{w_1 a_1}_{z_1}))$$

$$= \xi\left(w_2 \sigma\left(w_1 \sigma(\underbrace{w_0 a_0}_{z_0})\right)\right) = f(a_0)$$

## Backward step - last step

Let  $L(y, \hat{y})$  be the loss function between our predicted  $\hat{y}$  and actual  $y$ .  
Then the update for the last layer is:

$$\begin{aligned}\frac{\partial L}{\partial w_2} &= \frac{\partial L}{\partial a_3} \cdot \frac{\partial a_3}{\partial z_2} \cdot \frac{\partial z_2}{\partial w_2} \\ &= L'(a_3) \cdot \xi'(z_2) \cdot a_2\end{aligned}$$

## Backward step - second last step

Let  $L(y, \hat{y})$  be the loss function between our predicted  $\hat{y}$  and actual  $y$ .  
Then the update for the last layer is:

$$\begin{aligned}\frac{\partial L}{\partial w_1} &= \frac{\partial L}{\partial z_2} \cdot \frac{\partial z_2}{\partial w_1} \\&= \frac{\partial L}{\partial z_2} \cdot \frac{\partial z_2}{\partial a_2} \cdot \frac{\partial a_2}{\partial z_1} \cdot \frac{\partial z_1}{\partial w_1} \\&= \frac{\partial L}{\partial z_2} \cdot w_2 \cdot \sigma'(z_1) \cdot a_1\end{aligned}$$

## Backward step - third last step

Let  $L(y, \hat{y})$  be the loss function between our predicted  $\hat{y}$  and actual  $y$ . Then the update for the last layer is:

$$\begin{aligned}\frac{\partial L}{\partial w_0} &= \frac{\partial L}{\partial z_1} \cdot \frac{\partial z_1}{\partial w_0} \\ &= \frac{\partial L}{\partial z_1} \cdot \frac{\partial z_1}{\partial a_1} \cdot \frac{\partial a_1}{\partial z_0} \cdot \frac{\partial z_0}{\partial w_0} \\ &= \frac{\partial L}{\partial z_1} \cdot w_1 \cdot \sigma'(z_0) \cdot a_0\end{aligned}$$

## Using soft-max activation and cross-entropy loss

In `pytorch` there are two ways to implement a soft-max output layer with a cross entropy loss function:

Applying  
`torch.nn.Softmax` to the  
output layer

Use `torch.nn.NLLLoss` as  
a loss function

Applying no activation  
function to the output layer  
(linear activation)

Use  
`torch.nn.CrossEntropyLoss`  
as a loss function

The second way is preferable for numerical stability in the backward step (see [link](#)). In `tensorflow` these two options are implemented using a parameter in the `CategoricalCrossentropy` loss function.