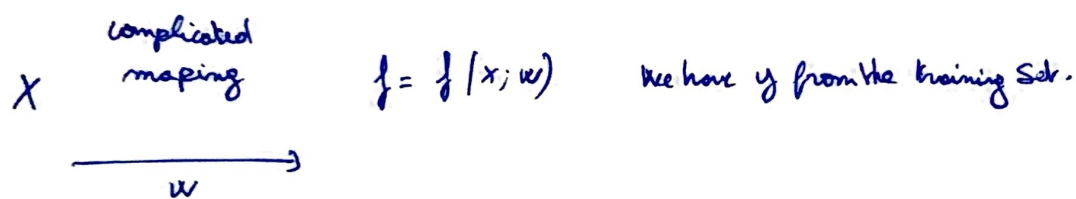


### Objectives:

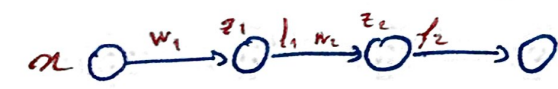
- Write down **Recursive relations** with **back propagation** algorithm to compute the gradient of the loss function with respect to the weight parameters.
- Use the **Stochastic descent Algorithm** to train a feedforward neural network.
- Understand that it is not guaranteed to reach global (only local) optimum with SGD to minimize the training loss.
- Recognize when a network has **overcapacity**.

### Back-propagation Algorithm: (computing the loss)

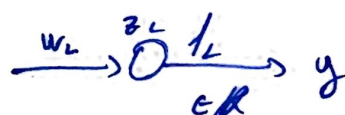
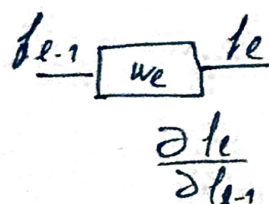


$(x, y)$  we have to nudge them in the reverse direction of the gradient.

$$w_{ij}^e \leftarrow w_{ij}^e - \eta \left[ \frac{\partial \text{loss}(y, f(x; w))}{\partial w_{ij}^e} \right]$$



$z_1 = x w_1$   
 $f_1 = \text{tanh}(x w_1)$   
 $\vdots$   
 $f_L = \text{tanh}(f_{L-1} w_L)$



$$\text{loss}(y, f_L) = \frac{1}{2} (y - f_L)^2$$

how much our prediction differ from our target value

$$\frac{\partial \text{loss}}{\partial w_1} = \frac{\partial f_1}{\partial w_1} \frac{\partial \text{loss}}{\partial f_1}$$

$(1 - f_1^2)x$       ?

$$\frac{\partial \text{loss}}{\partial f_1} = \frac{\partial f_L}{\partial f_1} \frac{\partial \text{loss}}{\partial f_L}$$

$\nwarrow (1 - f_L^2)w_2$   
 backward

Once we set up the architecture of our (feed-forward) neural Network, our goal will be to find weight parameters that minimize our loss function. We will use the Stochastic Gradient Descent Algorithm.

$$\theta \leftarrow \theta - \eta \nabla_{\theta} \text{Loss}$$

$\mathcal{L}(y, f_c)$  denote the loss function as a function of the prediction  $f_c$  and the true label  $y$ .

$$z_1 = x w_1$$

$$\text{for } i=2 \dots L: z_i = f_{i-1} w_i \quad \text{where } f_{i-1} = f(z_{i-1})$$

$$\text{let } \delta_i = \frac{\partial \mathcal{L}}{\partial z_i}$$

The 1<sup>st</sup> step to updating any weight  $w$  is to calculate  $\frac{\partial \mathcal{L}}{\partial w}$



$$\frac{\partial \mathcal{L}}{\partial w_1} = \frac{\partial z_1}{\partial w_1} \frac{\partial \mathcal{L}}{\partial z_1} \quad \text{since } z_1 = w_1 x:$$

$$\frac{\partial z_1}{\partial w_1} = \frac{\partial (w_1 x)}{\partial w_1} = x \Rightarrow \frac{\partial \mathcal{L}}{\partial w_1} = x \cdot \frac{\partial \mathcal{L}}{\partial z_1} = x \cdot \delta_1$$

### Recursive Expression:

In this problem, we derive a recurrence relation between  $\delta_i$  and  $\delta_{i+1}$

$$f(x) = \tanh(x)$$

$$f'(x) = (1 - \tanh^2(x))$$

Expression of  $\delta_1$  in terms of  $\delta_2$ ?

The chain Rule gives:  $\delta_1 = \frac{\partial \mathcal{L}}{\partial z_1} = \frac{\partial f_1}{\partial z_1} \frac{\partial z_2}{\partial f_1} \frac{\partial \mathcal{L}}{\partial z_2}$

Since  $f_1 = \tanh(z_1) \Rightarrow \frac{\partial f_1}{\partial z_1} = (1 - f_1^2)$

Since  $z_2 = w_2 \cdot f_1 \Rightarrow \frac{\partial z_2}{\partial f_1} = w_2$

Substituting the values of  $\frac{\partial f_1}{\partial z_1}$ ,  $\frac{\partial z_2}{\partial f_1}$  into the main expression for  $\delta_1$  we get:

$$\delta_1 = (1 - f_1^2) \cdot w_2 \cdot \frac{\partial \mathcal{L}}{\partial z_2} = (1 - f_1^2) \cdot w_2 \cdot \delta_2$$

## Final Expression of the Gradient:

Let  $\mathcal{L}(y, f_L)$  denote the loss function as a function of the predictions  $f_L$  and the true label  $y$ .

$$\delta_i = \frac{\partial \mathcal{L}}{\partial z_i}$$

$$\mathcal{L}(y, f_L) = (y - f_L)^2$$

Compute  $\frac{\partial \mathcal{L}}{\partial w_1}$ :

From the previous problem:  $\frac{\partial \mathcal{L}}{\partial w_1} = \alpha \delta_1$

$$\delta_1 = (1 - f_1^2) \cdot w_2 \cdot \delta_2$$

Similarly,  $\delta_2, \delta_3, \dots, \delta_L$  can be given as follows:

$$\delta_2 = (1 - f_2^2) w_3 \cdot \delta_3$$

$$\delta_3 = (1 - f_3^2) w_4 \cdot \delta_4$$

$\vdots$

$$\delta_{L-1} = (1 - f_{L-1}^2) \cdot w_L \cdot \delta_L$$

$$\delta_L = \frac{\partial \mathcal{L}}{\partial z_L}$$

$$\delta_L = \frac{\partial \mathcal{L}}{\partial f_L} \cdot \frac{\partial f_L}{\partial z_L} \rightarrow \delta_L = \frac{\partial (f_L - y)^2}{\partial f_L} \cdot \frac{\partial f_L}{\partial z_L} = 2(f_L - y) \frac{\partial f_L}{\partial z_L}$$

$$\delta_L = 2(f_L - y)(1 - f_L^2)$$

Plugging the above equations into the expression for  $\frac{\partial \mathcal{L}}{\partial w_1}$  we get:

$$\frac{\partial \mathcal{L}}{\partial w_1} = \alpha \cdot \delta_1$$

$$\frac{\partial \mathcal{L}}{\partial w_1} = \alpha \cdot (1 - f_1^2) \cdot w_2 \cdot \delta_2$$

$$\frac{\partial \mathcal{L}}{\partial w_1} = \alpha \cdot (1 - f_1^2) \cdot w_2 \cdot (1 - f_2^2) \cdot w_3 \cdot \delta_3$$

$\vdots$

$$\frac{\partial \mathcal{L}}{\partial w_1} = \alpha (1 - f_1^2) (1 - f_2^2) \cdots (1 - f_L^2) w_2 w_3 \cdots w_L (2(f_L - y))$$

- For multi-layer neural networks the loss function is no longer convex and any stochastic gradient descent (SGD) method is not guaranteed to reach global optimum.

- Larger models tend to be easier to learn because their units need to be adjusted so that they are collectively sufficient to solve the task.