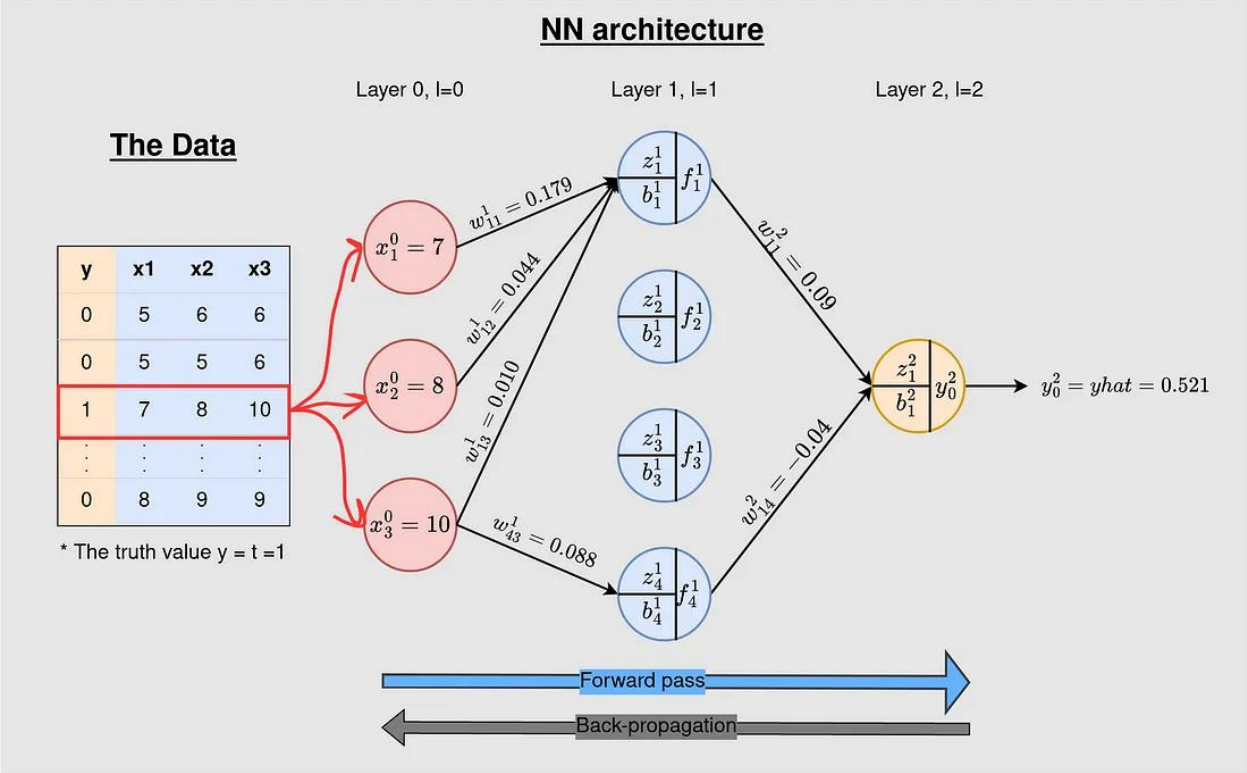
Cairo University
Faculty of Computers & Artificial Intelligence
Supervised Learning (2022/2023)

Neural Network and Multiclass

Neural Network Training

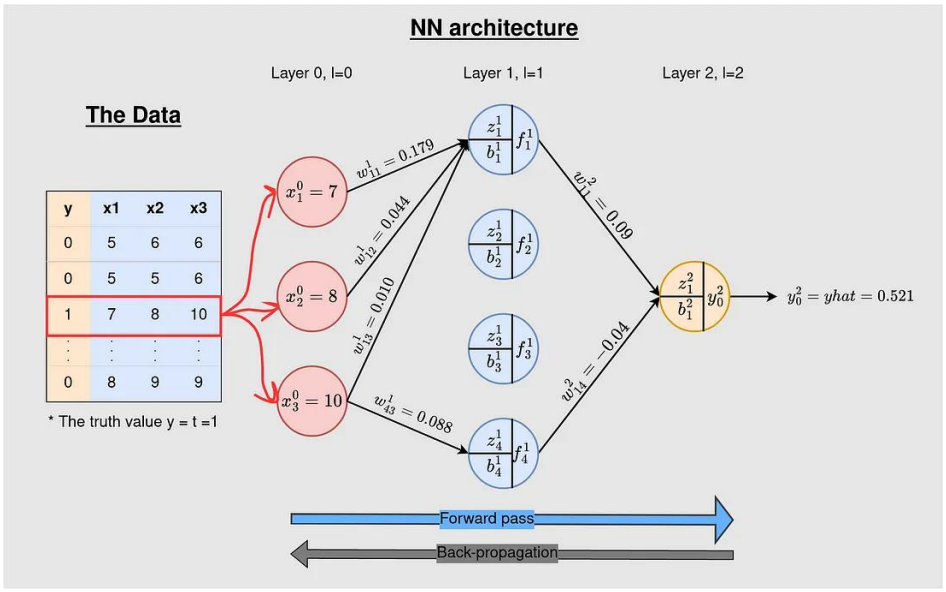
Neural Network training consists of 2 main steps.

1- Forward Pass
2- Back Propagation



Forward Pass consists mainly of multiplying weights and input then applying any differentiable activation function which we will assume here to be sigmoid function. After reaching the output layer, The error will be calculated by applying the loss function between yhat and y.

This is the state after the first forward pass



Input layer (l=0)

	i	value
x_i^0	1	7
	2	8
	3	10

- **i** - is the node in the previous layer, **l-1**,
- **j** in the node/unit in the current layer, **l**.
- Example, w_{ji}^1, means weight value connecting node i in layer **l-1** (**l=0** - the input) and node **j** in layer **1**.

Hidden layer (l=1)

	j / i	1	2	3	4
w_{ji}^1	1	0.179	-0.186	-0.008	-0.048
	2	0.044	-0.028	-0.063	-0.131
	3	0.01	-0.035	-0.004	0.088
b_j^1		0	0	0	0
f_j^1		0.845	0.132	0.354	0.377

Output layer (l=2)

	i	j=1
w_{ji}^2	1	0.088
	2	0.171
	3	0.005
	4	-0.04
b_j^2		0
yhat		0.521

* The output of a forward pass is yhat = **0.521**

* Remember that the truth value y=1.

To program the forward pass:

1- You need to initialize your weights W. Your weights should be stored in a list where the size of the list is the number of your Neural Network layers – 1 (In our example, size = 2). Each item in the list contains a numpy array. Its size is determined based on the neurons in the layer it represents and the layer before it. For example, The hidden layer weights (weights at index 0) will be 4 * 3. 4 is the hidden layer size and 3 is the input layer size.

2- Initialize List F that represents the output of each layer after applying activation function. Its size should the number of your Neural Network layers. Each item in the list contains a numpy array. Its size is determined based on the neurons in the layer it represents. For example, The hidden layer outputs F (outputs at index 0) will be of length 4. The first item in the list f (index 0) will represent the input samples

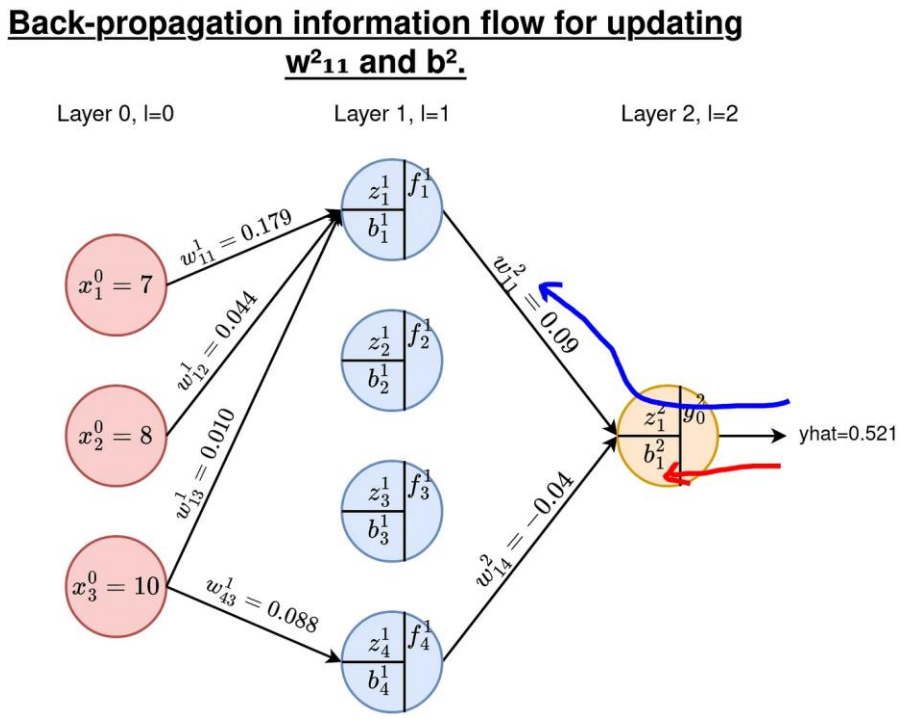
3- For each layer from layer after input layer to output, Multiply its weights by the output of the previous layer f then apply activation function and store the result in its Index in f

4- Calculate the error

Back propagation

Updating Parameters on the Output-Hidden Layer

Updating the weights will be done through the following function



$$w_{t+1} := w_t - \epsilon \frac{\partial E(\theta)}{\partial w}$$

$$\frac{\partial E}{\partial w_{11}^2} = \frac{\partial E}{\partial yhat} \frac{\partial yhat}{z_1^2} \frac{\partial z_1^2}{\partial w_{11}^2}$$

and,

$$\frac{\partial z_1^2}{\partial w_{11}^2} = \frac{\partial}{\partial w_{11}^2}(f_1^1 w_{11}^2 + f_2^1 w_{12}^2 + f_3^1 w_{13}^2 + f_4^1 w_{14}^2 + b_1^2)$$
$$= f_1^1$$

$$\frac{\partial yhat}{\partial z_1^2} = \frac{\partial}{\partial z_1^2} g(z_1^2) = \frac{\partial}{\partial z_1^2}\left(\frac{1}{1+\exp^{-z_1^2}}\right) = (1-yhat)yhat$$

$$\begin{aligned}\frac{\partial E}{\partial yhat} &= \frac{\partial}{\partial yhat}(-[t\ln yhat + (1-t)\ln(1-yhat)]) \\ &= -\frac{t}{yhat} + \frac{1-t}{1-yhat}\end{aligned}$$

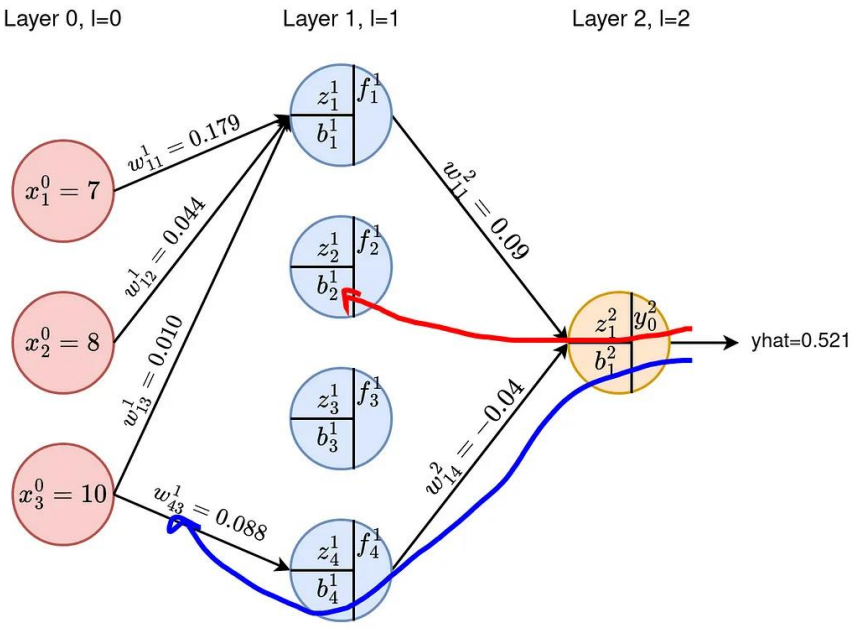
Therefore,

$$\begin{aligned}\frac{\partial E}{\partial w_{11}^2} &= \frac{\partial E}{\partial yhat}\frac{\partial yhat}{z_1^2}\frac{\partial z_1^2}{\partial w_{11}^2} \\ &= -\frac{t}{yhat} + \frac{1-t}{1-yhat}\cdot(1-yhat)yhat\cdot f_1^1, \\ &= (yhat - t)\cdot f_1^1, \\ &= (0.521-1)0.845, \\ &= -0.405\end{aligned}$$

UPDATING PARAMETERS ON THE HIDDEN-INPUT/HIDDEN LAYER

Back-propagation information flow for updating

w^1_{43} and b^1_2.



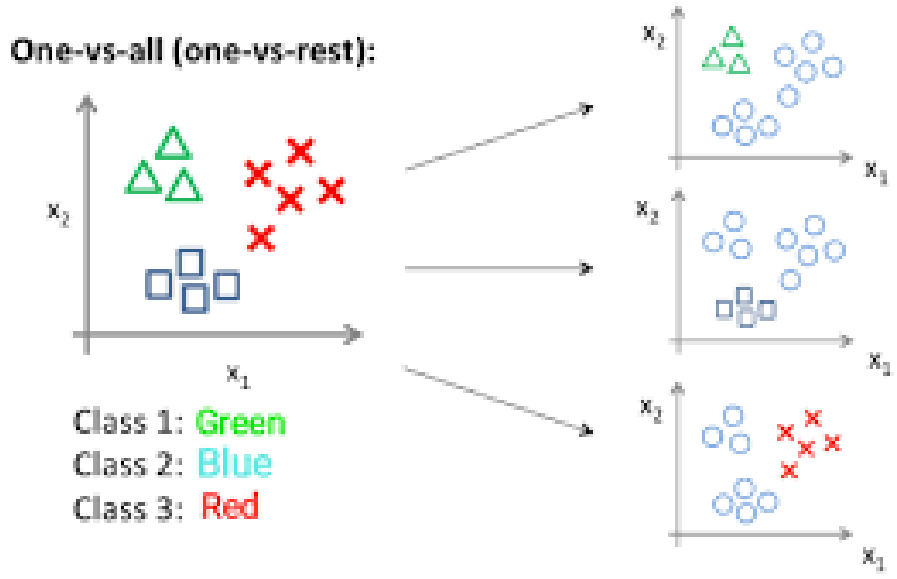
$$\frac{\partial E}{\partial w_{43}^1} = \frac{\partial E}{\partial yhat} \frac{\partial yhat}{\partial z_1^2} \frac{\partial z_1^2}{\partial f_4^1} \frac{\partial f_4^1}{\partial z_4^1} \frac{\partial z_4^1}{\partial w_{43}^1}$$

To program the backpropagation pass:

1- Initialize delta list. Its size is the number of your Neural Network layers – 1 (In our example, size = 2). Each item in the list contains a numpy array. Its size is determined based on the number of neurons in the layer it represents. For example, The hidden layer delta (delta at index 0) will be 4.
2- For each layer L from output to input:
    a. For each neuron j in layer L
        i. If L is output layer:
            1. Calculate $delta^L_j = F_j (1-F_j) (yhat_j – F_j)$
        ii. Else If L is Hidden Layer:
            1. For each neuron k in layer L+1:
                a. Calculate $U = U + delta_k^{L+1} * W[L+1](k, j)$
            2. Calculate $delta^L_j = F_j (1-F_j) U$
        iii. For each neuron i in Layer L-1:
            1. $W[L](j, i) = W[L](j, i) + eta * F_i * delta^L_j$

Multiclass

If we have multiple classes in our problem and we need to use logistic regression, We will train many logistic regression models where each model will be trained on 1 class only.



Task

Edit Your assignment 1 as follows:-

1- **Remove The filtering on 0, 1 classes and remove your k fold code to speed up the training time**
2- **Apply L2 Regularization**
3- **Apply one hot encoding on your dataset labels (1-10)**
4- **Change your weights matrix to be of size (# of classes, # of features) instead of (# of features)**
5- **For each class c :**
    a. **Y = labels[:, c]**
    b. **Train logistic regression and save weights in w[c, :]**
6- **Predict the final class for your data (train or test)**
    a. **For each class c:**
        i. **Z[:, c] = w[:, c] * X**
    b. **Calculate softmax value for each Z where K is the number of classes**

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

    c. **The index that has the highest value is the final class**
    d. **Calculate accuracy.**