ML models for tabular datasets (100 Points)

1 Comparison between some models

Answer the following questions

1.1 (10 Points)

What are the advantages of Random Forests over decision trees?

Ans:

- random forest chooses a subset of data and features so it's less likely to match the noise, and because it uses the average answer of the trees it's less biased and not tending to overfit which was the main problem with the decision trees.

1.2 (10 Points)

What are the advantages of XGBoost over random forest, adaboost and gradient boosting?

Ans:

- the calculation is much faster than decision tree and AdaBoost, and its more accurate,

- And comparing to gradient boost, XGBoost covers one of the problems gradient boosting had, which was not considering the potential loss for possible splits

2 Fitting a model

In this question, we want to fit a model for diabetes according to a given data.

2.1 (30 Points)

How Random Forest and AdaBoost fit a model to the following data?

Blocked Arteries	Weight	Exercise	Genetics	Has Diabetes
No	210	Yes	Yes	No
No	125	No	No	No
Yes	180	Yes	Yes	Yes
Yes	167	No	Yes	Yes

Ans:

- Random Forest:

- 1: RF builds a bootstrapped data set. (Choose random samples and each sample is allowed more than 1 time)

Bootstrapped Dataset				
Blocked Arteries	Weight	Exercise	Genetics	Has diabetes
Yes	180	Yes	Yes	Yes
No	125	No	No	No
Yes	167	No	Yes	Yes
Yes	167	No	Yes	Yes

- 2: RF builds a decision tree from a subset of the bootstrapped data set. (Only some of the features are selected for each node to choose the best split feature)

- 3: RF repeats step 1 and 2 till it reaches the hyperparameter set for it.

- 4: RF runs each test sample (out of bag sample) in all the trees build before and takes the mode or average or weighted average to return the final result.

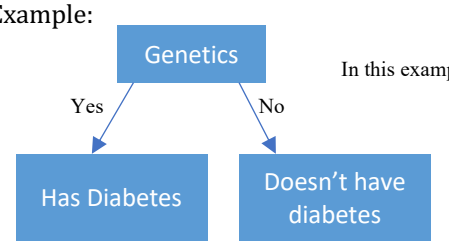
- AdaBoost:

- 1: AB will give each sample a weight, in the first place each sample weight is $(\frac{1}{number\ of\ samples})$

- 2: AB makes the first Stump using the feature that best splits the data (Gini index), then updates the sample weights and this stumps amount of say.

Amount of say = $\frac{1}{2}\ln(\frac{all}{False(negative\ and\ positive)} - 1)$

Example:



In this example amount of say will be:

$$\frac{1}{2}\ln\left(\frac{4}{1} - 1\right) = 0.55$$

- 3: AB will update the weights paying more attention to the false predicted samples, new sample weight for false predicted samples is calculated using this formula:

$$new\ sample\ weight = sample\ weight \times e^{amount\ of\ say}$$

In our example it will be sample 1: $new\ weight\ sample_1 = \frac{1}{4}e^{0.55} = 0.43$

And for true predicted samples:

$$new\ sample\ weight = sample\ weight \times e^{-amount\ of\ say}$$

In our example it will be sample 2,3,4: $new\ weight\ sample_{2,3,4} = \frac{1}{4}e^{-0.55} = 0.14$

And to normalize the weights we divide them by their current sum.

Updated table:

Blocked Arteries	Weight	Exercise	Genetics	Has diabetes	Sample Weight
No	210	Yes	Yes	No	0.50
No	125	No	No	No	0.16
Yes	180	Yes	Yes	Yes	0.16
Yes	167	No	Yes	Yes	0.16

- 4: AB builds a Bootstrapped dataset using sample weights as their probability for choosing.
- 5: AB repeats steps 1 to 4 again…
- 6: Now for the test each sample is run down all the trees and each time it ends up as "Has diabetes" or "doesn't have diabetes" the amount of say of that tree will be added up and then compared and it will be classified to the group which had the better sum.

2.2 (30 Points)

Now suppose that two entries of the data table ("Blocked Arteries" and "Weight" of the 4th data) are missing. Write step by step how random forest can fill the missing data?

Ans:

- 1: First, we make a not so good guess based on other samples with the same value of "has diabetes", here the current sample doesn't have heart disease so for predicting the two missing values we look at the other samples with no heart disease, there are two other samples who don't have heart disease and their value for "blocked arteries" is No, so our prediction for blocked arteries would be No, and their value for "weight" is 210 and 125, because weight is numeric our prediction will be the mean of those which is 167.5
- The table will now look like below:

Blocked Arteries	weight	Exercise	Genetics	Has diabetes
No	210	Yes	Yes	No
No	125	No	No	No
Yes	180	Yes	Yes	Yes
No	**167.5**	No	Yes	No

- 2: to make our guesses better we make number of decision trees, and we build a similarity matrix which is np.zeroes in the beginning then we look at how many samples ended up in the same leaf and and increase the value of m_{ij} & m_{ji} by one if sample i and sample j ended up in the same node. Then we divide all cells of the matrix by number of trees, now we update the table again using weighted frequency (for categorial data) and weighted average (for numeric data)
- 3: repeat the second step till convergence appears!

3 Gradient Boost (20 Points)

Write step by step how Gradient Boost fits a model to the following data?

Gender	Height (m)	Blood Pressure	Weight (kg)
Female	1.6	Medium	76
Male	1.6	Low	88
Female	1.5	Low	56
Female	1.4	Low	57
Male	1.8	High	73
Male	1.5	Medium	77

Ans:

- 1: First GB calculates the average weight: here it is 71.16
- 2: Then GB will calculate the Residuals
- 3: GB will build a tree predicting "residuals"! Usually small trees, but not stumps, each leaf's residual is the average residual of the samples ending up in that leaf.
- 4: now GB updates the residuals:

$$new\ residual = original\ weight - (Average\ Weight + learning\ rate \times residual\ of\ the\ leaf\ this\ sample\ belongs\ to)$$

This is a small step in the right direction

- 5: GB will repeat step 3
- 6: GB will update the residuals but takes sum of previous trees prediction and current tree.

$$new\ residual = original\ weight - (Average\ Weight + learning\ rate \times \sum residual\ of\ the\ leaf\ this\ sample\ belongs\ to)$$

- 7: GB will repeat this till the number of trees reaches the hyperparameter set for it.

Intro to NNs (100 Points)

4 Descriptive Questions (35 Points)

Give a brief explanation for the following statements.

4.1 (5 Points)

Explain the Exploding Gradient problem. Propose a solution to prevent this issue.

Ans:

- in a neural network we update the weights by calculating the gradient and subtracting it from the current weight, in order to calculate the gradient, we have to use the 'chain rule' which results in multiple multiplications now imagine the case that each derivative becomes a large number, it will grow the gradient exponentially, making our model unstable because overflow may happen.

- one of the solutions to prevent this issue is to use 'Gradient Clipping', for that we consider an upper-bond and check the gradient norm with that if it was bigger, we normalize the gradient ($\frac{\vec{G}}{\|\vec{G}\|}$) so the direction is preserved and then we can multiply it by the upper-bond so it's in the same direction but not too big.

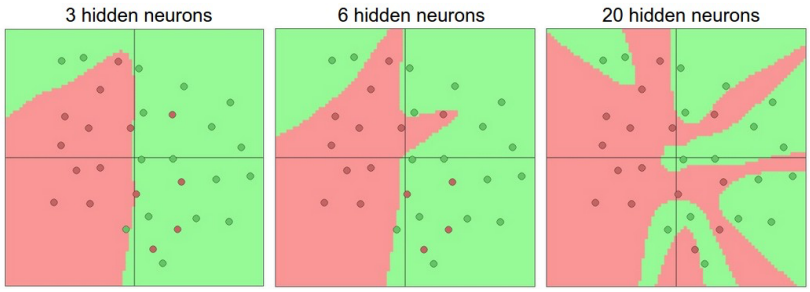
pseudocode
If $\|\vec{G}\| > upperbond$: $\vec{G} = \frac{\vec{G}}{\|\vec{G}\|} \times upperbond$

4.2 (5 Points)

What are the Pros. and Cons. of adding more layers to a deep neural network?

Ans:

+ The benefits of adding more layers to a neural network is that with more layers our model gets more complex and can find more hidden relationships and resulting in minimizing the error rate, which is shown below in the picture:



- But adding more layers to a neural network exponentially increases the number of perceptron and increases the computational cost, it may also result in exploding gradient problem and overfitting problem too.

4.3 (5 Points)

In Stochastic Gradient Descent algorithm, why does the first step of the procedure require shuffling the training dataset?

Ans:

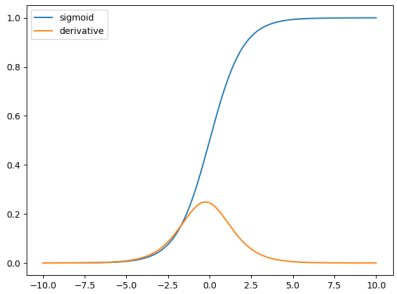
- Because in the plot of the loss function there may be several local minima points and our model may get stuck in one of them in order to avoid that we shuffle the data so every time we are starting from a new point and it reduces the probability of getting stuck in the local minima.

4.4 (5 Points)

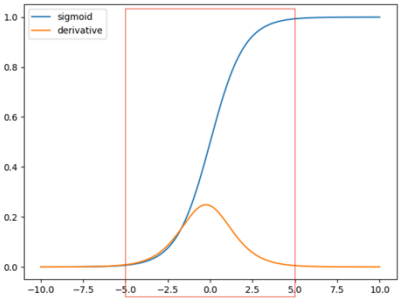
Consider Sigmoid activation function. What will be the gradient value of this function for a very large input? What problem does this cause for neural network training? How could this be solved?

Ans:

- updating the weights using gradient descent requires multiple multiplications explained in the above questions and calculated in the Q6, in these calculations the term $\frac{\partial \sigma(x)}{\partial x} = \sigma(x)\big(1 - \sigma(x)\big)$ is used a lot now take a look at the sigmoid derivative function graph:



So, for large inputs the derivative function is so small, now repeatedly multiplying these small values exponentially reduces the gradient value resulting in the so called 'vanishing gradient' problem, in order to solve the vanishing gradient problem, we can reduce the number of the layers (the simplest solution) or we can use 'batch normalization', in batch normalization we map the input into a space that the sigmoid derivative doesn't cause the vanishing problem (map the input between [-5,5])



4.5 (10 Points)

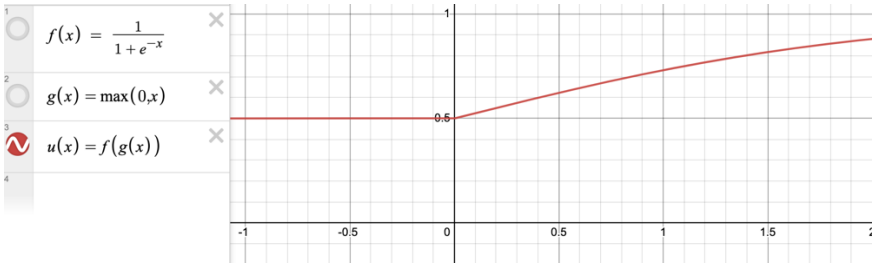
Consider a Multilayer Perceptron which is used to provide a 2-class classifier. Consider the output of the last neuron to be z, and the output of the neural network as:

$$y = \sigma(\text{RELU}(z))$$

The Outputs which are greater than 0.5 are considered as class 1. What is the problem of this neural network?

Ans:

- to answer this lets first take a look at the $u(x) = \sigma(ReLU(x))$ function graph:



- now imagine a case that our model has misclassified a class 1 output as a class 2 now in the back propagation for updating the weights we have $\frac{\partial E_i}{\partial w_j} = (o_i - t_i)\left(\frac{\partial u}{\partial z}\right) \dots$, as it was misclassified to class 2 it means z < 0, now looking at the graph $\frac{\partial u}{\partial z} : z < 0 = 0$, so it means the weights won't update and it will remain misclassified.

4.6 (5 Points)

What would happen to MLPs if we did not have activation functions?

Ans:

- if we consider the input data as vector 'x' and 'W_i' as weight matrix for i_{th} layer of the NN then the result will be like:

$$h_1 = xW_1 + b_1$$

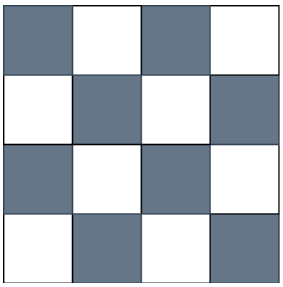
$$h_2 = h_1W_2 + b_2 = (xW_1 + b_1)W_2 + b_2 = xW_1W_2 + b_1W_2 + b_2$$

$$h_n = x\prod_{i=1}^{n} W_i + \sum_{i=0}^{n} b_{n-i}\prod_{j=1}^{i} W_j$$

As you can see it will end up as a linear model, our purpose to use hidden layers were to find nonlinear relations which isn't happening right now and we have to use nonlinear activation functions like tanh, sigmoid, … so our model can learn nonlinear relations.

5 Intuitive Questions (25 Points) 5.1 (20 Points)

Consider a 4*4 image (16 pixels) like the below figure. The black squares have the value 0, and the white ones have the value of 1.
In this task, you have to find the following chess-like pattern by designing an MLP. You have to pick the suitable weights, biases and activation functions (Step, RELU, etc.) for the network.



Hint: The input Layer should contain 16 Neurons. The output node specifies whether if the image matches the pattern or not.

Ans:

- if inputs are zero and one (zero means white tile and one means black tile), then the image matches the pattern then if input (1,3,6,8,9,11,14,16) are 1 and the rest are zero, we can represent this as logical gates and then implement the logical gates using neural networks:

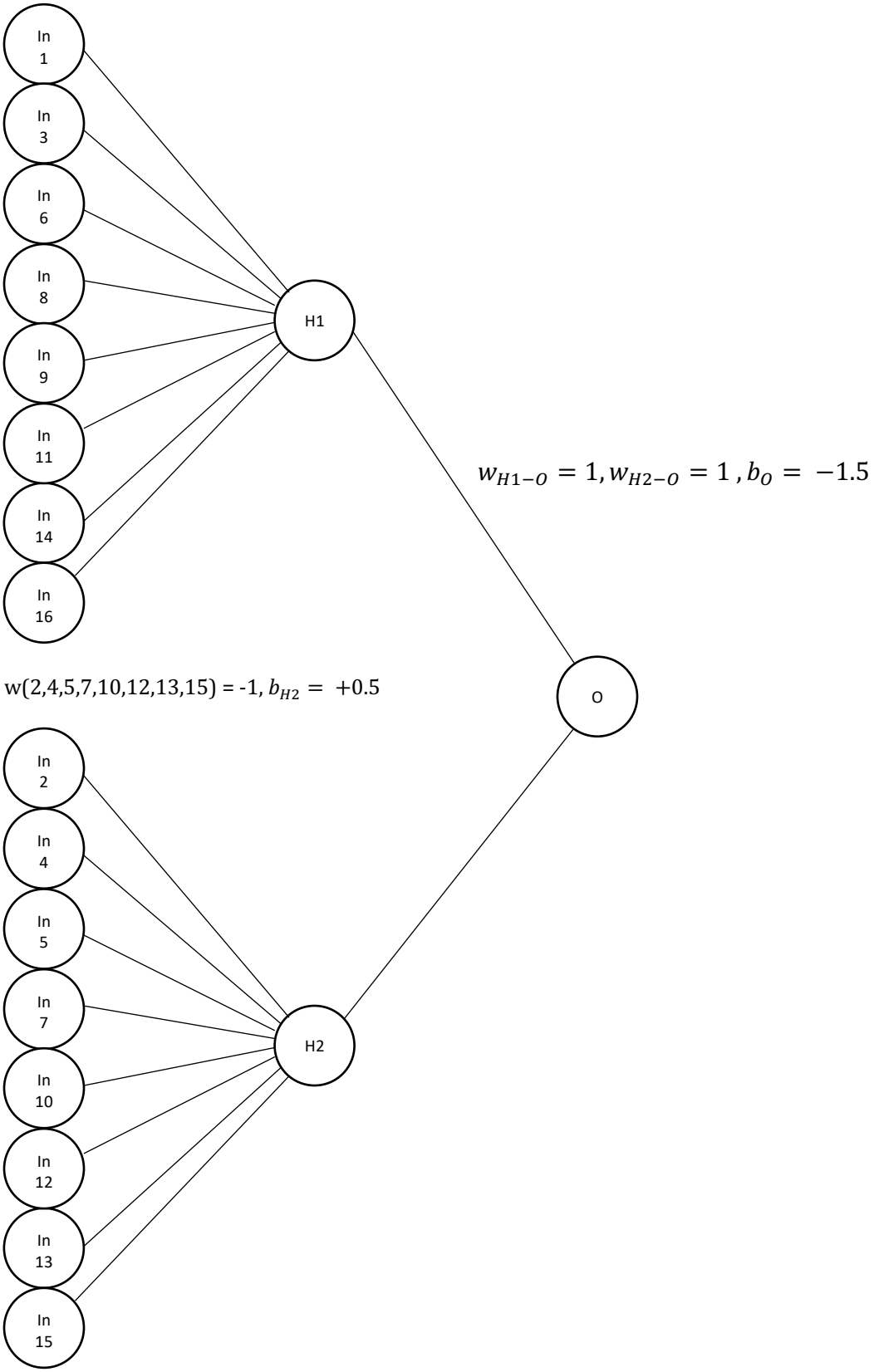
AND (AND (1,3,6,8,9,11,14,16), AND (NOT (2,4,5,7,10,12,13,15)))

- the neural network is designed on the next page:

- weights and biases are written next to the design, and the activation function is STEP.

- our design is not a fully connected neural network.

w(1,3,6,8,9,11,14,16) = 1, $b_{H1} = -7.5$



w(2,4,5,7,10,12,13,15) = -1, $b_{H2} = +0.5$

5.2 (5 Points)

What are the problems of using MLP as an Image/Pattern Classifier?

Ans:

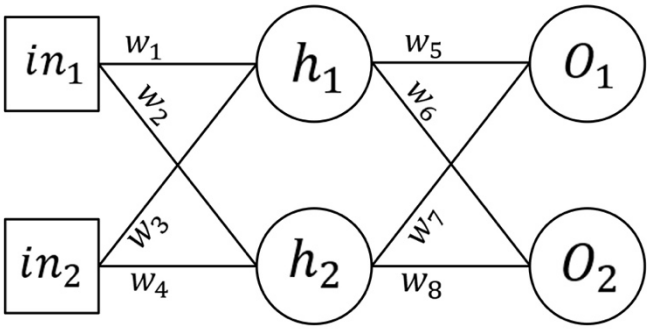
- 1: MLP takes vectors as input so image must be flattened (if its tensor or matrix). Resulting in loss of information sometimes.

-2: if the pattern we are looking to classify appears in different places of the image given for instance one time on top one time on left and etc. MLP cannot handle it and is used to just one place.

-3: if the given vector(image) is so large then the number of parameters in the MLP becomes even larger (for n inputs, $\binom{n}{2}$ connections in each layer) which is redundant.

6 Computational Question (40 Points)

Take a look at this Neural Network. (The activation function is sigmoid)



Consider these parameters:

$w_i = i \times 0.1$, $in_1 = 0.1$, $in_2 = 0.5$, $b_{h1} = 0.25$, $b_{h2} = 0.25$, $b_{o2} = 0.35$, $b_{o2} = 0.35$, $lr = 1$, $t_1 = 0.05$, $t_2 = 0.95$

$$E = \frac{1}{2}\sum(t - o)^2$$

Calculate the updated weights after 1 iteration of forward pass and backward pass. Consider decimal accuracy up to 3 digits.

Ans:

1- Forward pass: in order to calculate the outputs of the neurons pass we have to pass the weighted sum of inputs + bias to the activation function (sigmoid function):

- $h_1 = \frac{1}{1+e^{-sum_1}}$, $sum_1 = 0.1 \times 0.1 + 0.5 \times 0.3 + 0.25 = 0.41$, $h_1 = \frac{1}{1+e^{-0.41}} = 0.601$
- $h_2 = \frac{1}{1+e^{-sum_2}}$, $sum_2 = 0.1 \times 0.2 + 0.5 \times 0.4 + 0.25 = 0.47$, $h_1 = \frac{1}{1+e^{-0.47}} = 0.615$
- $o_1 = \frac{1}{1+e^{-sum_3}}$, $sum_3 = 0.601 \times 0.5 + 0.615 \times 0.7 + 0.35 = 1.081$, $h_1 = \frac{1}{1+e^{-1.081}} = 0.746$
- $o_2 = \frac{1}{1+e^{-sum_4}}$, $sum_4 = 0.601 \times 0.6 + 0.615 \times 0.8 + 0.25 = 0.47$, $h_1 = \frac{1}{1+e^{-0.47}} = 0.768$
- $E = \frac{1}{2}\sum(t - o)^2 = \frac{1}{2}((0.050 - 0.746)^2 + (0.950 - 0.768)^2) = 0.258$, $E_1 = 0.242$, $E_2 = 0.016$
-

2- Backward pass: in order to update weights, we have to see how much is their influence it means we have to take derivative of Error with respect to w_i, for that we use the chain rule:

- $E = \frac{1}{2}\sum(t - o)^2 = \frac{1}{2}((t_1 - o_1)^2 + (t_2 - o_2)^2)$, $\frac{\partial E}{\partial o_1} = o_1 - t_1$, $\frac{\partial E}{\partial o_2} = o_2 - t_2$,
- $\frac{\partial o_1}{\partial sum} = o_1(1 - o_1)$: *because its sigmoid function,*
- $\frac{\partial sum_1}{\partial w_5} = h_1$
- $\frac{\partial E}{\partial w_5} = \frac{\partial E}{\partial o_1} \times \frac{\partial o_1}{\partial sum_3} \times \frac{\partial sum_3}{\partial w_5} = (o_1 - t_1)o_1(1 - o_1)h_1 = 0.072$
- Update the weight: $updated_w = w - lr \times \frac{\partial E}{\partial w}$
- $updated_{w_5} = w_5 - 1 \times 0.072 = 0.428$
- *we repeat the same for* w_6, w_7, w_8:
- $\frac{\partial E}{\partial w_6} = \frac{\partial E}{\partial o_2} \times \frac{\partial o_2}{\partial sum_4} \times \frac{\partial sum_4}{\partial w_6} = (o_2 - t_2)o_2(1 - o_2)h_1 = -0.019$

- $updated_{w_6} = w_6 - 1 \times -0.019 = 0.619$

- $\frac{\partial E}{\partial w_7} = \frac{\partial E}{\partial o_1} \times \frac{\partial o_1}{\partial sum_3} \times \frac{\partial sum_3}{\partial w_7} = (o_1 - t_1)o_1(1 - o_1)h_2 = 0.081$
- $updated_{w_7} = w_7 - 1 \times 0.081 = 0.619$

- $\frac{\partial E}{\partial w_8} = \frac{\partial E}{\partial o_2} \times \frac{\partial o_2}{\partial sum_4} \times \frac{\partial sum_4}{\partial w_8} = (o_2 - t_2)o_2(1 - o_2)h_2 = -0.019$

- $updated_{w_8} = w_8 - 1 \times -0.019 = 0.819$

- For w_{1-4} the chain gets a bit longer we have:

- $\frac{\partial E}{\partial w_1} = \frac{\partial E_1}{\partial w_1} + \frac{\partial E_2}{\partial w_1}$

- $\frac{\partial E_1}{\partial w_1} = \frac{\partial E_1}{\partial o_1} \times \frac{\partial o_1}{\partial sum_3} \times \frac{\partial sum_3}{\partial h_1} \times \frac{\partial h_1}{\partial sum_1} \times \frac{\partial sum_1}{\partial w_1} = (o_1 - t_1)o_1(1 - o_1)w_5h_1(1 - h_1)in_1 = 0.001$

- $\frac{\partial E_2}{\partial w_1} = \frac{\partial E_2}{\partial o_2} \times \frac{\partial o_2}{\partial sum_4} \times \frac{\partial sum_4}{\partial h_1} \times \frac{\partial h_1}{\partial sum_1} \times \frac{\partial sum_1}{\partial w_1} = (o_2 - t_2)o_2(1 - o_2)w_6h_1(1 - h_1)in_1 = -0.000$
- $updated_{w_1} = w_1 - 1 \times (0.001 - 0.000) = 0.099$

- $\frac{\partial E}{\partial w_2} = \frac{\partial E_1}{\partial w_2} + \frac{\partial E_2}{\partial w_2}$

- $\frac{\partial E_1}{\partial w_2} = \frac{\partial E_1}{\partial o_1} \times \frac{\partial o_1}{\partial sum_3} \times \frac{\partial sum_3}{\partial h_2} \times \frac{\partial h_2}{\partial sum_2} \times \frac{\partial sum_2}{\partial w_2} = (o_1 - t_1)o_1(1 - o_1)w_7h_2(1 - h_2)in_1 = 0.002$

- $\frac{\partial E_2}{\partial w_2} = \frac{\partial E_2}{\partial o_2} \times \frac{\partial o_2}{\partial sum_4} \times \frac{\partial sum_4}{\partial h_2} \times \frac{\partial h_2}{\partial sum_2} \times \frac{\partial sum_2}{\partial w_2} = (o_2 - t_2)o_2(1 - o_2)w_8h_2(1 - h_2)in_1 = -0.000$
- $updated_{w_2} = w_2 - 1 \times (0.002 - 0.000) = 0.198$

- $\frac{\partial E}{\partial w_3} = \frac{\partial E_1}{\partial w_3} + \frac{\partial E_2}{\partial w_3}$

- $\frac{\partial E_1}{\partial w_3} = \frac{\partial E_1}{\partial o_1} \times \frac{\partial o_1}{\partial sum_3} \times \frac{\partial sum_3}{\partial h_1} \times \frac{\partial h_1}{\partial sum_1} \times \frac{\partial sum_1}{\partial w_3} = (o_1 - t_1)o_1(1 - o_1)w_5h_1(1 - h_1)in_2 = 0.007$

- $\frac{\partial E_2}{\partial w_3} = \frac{\partial E_2}{\partial o_2} \times \frac{\partial o_2}{\partial sum_4} \times \frac{\partial sum_4}{\partial h_1} \times \frac{\partial h_1}{\partial sum_1} \times \frac{\partial sum_1}{\partial w_3} = (o_2 - t_2)o_2(1 - o_2)w_6h_1(1 - h_1)in_2 = -0.002$
- $updated_{w_3} = w_3 - 1 \times (0.007 - 0.002) = 0.295$

- $\frac{\partial E}{\partial w_4} = \frac{\partial E_1}{\partial w_4} + \frac{\partial E_2}{\partial w_4}$

- $\frac{\partial E_1}{\partial w_4} = \frac{\partial E_1}{\partial o_1} \times \frac{\partial o_1}{\partial sum_3} \times \frac{\partial sum_3}{\partial h_2} \times \frac{\partial h_2}{\partial sum_2} \times \frac{\partial sum_2}{\partial w_4} = (o_1 - t_1)o_1(1 - o_1)w_7h_2(1 - h_2)in_2 = 0.010$

- $\frac{\partial E_2}{\partial w_4} = \frac{\partial E_2}{\partial o_2} \times \frac{\partial o_2}{\partial sum_4} \times \frac{\partial sum_4}{\partial h_2} \times \frac{\partial h_2}{\partial sum_2} \times \frac{\partial sum_2}{\partial w_4} = (o_2 - t_2)o_2(1 - o_2)w_8h_2(1 - h_2)in_2 = -0.003$
- $updated_{w_4} = w_4 - 1 \times (0.010 - 0.003) = 0.393$

One iteration of forward and backward pass is completed.

p.s.: I thought there is no need to update the biases but if it was needed it would be the same as we did for the weights.