1 Question 1

Our data has two features x_1 and x_2, and the "output" is a binary classification (either 0 or 1). There are 7 neurons in the hidden layer, and one output. The diagram below shows the structure of this network.

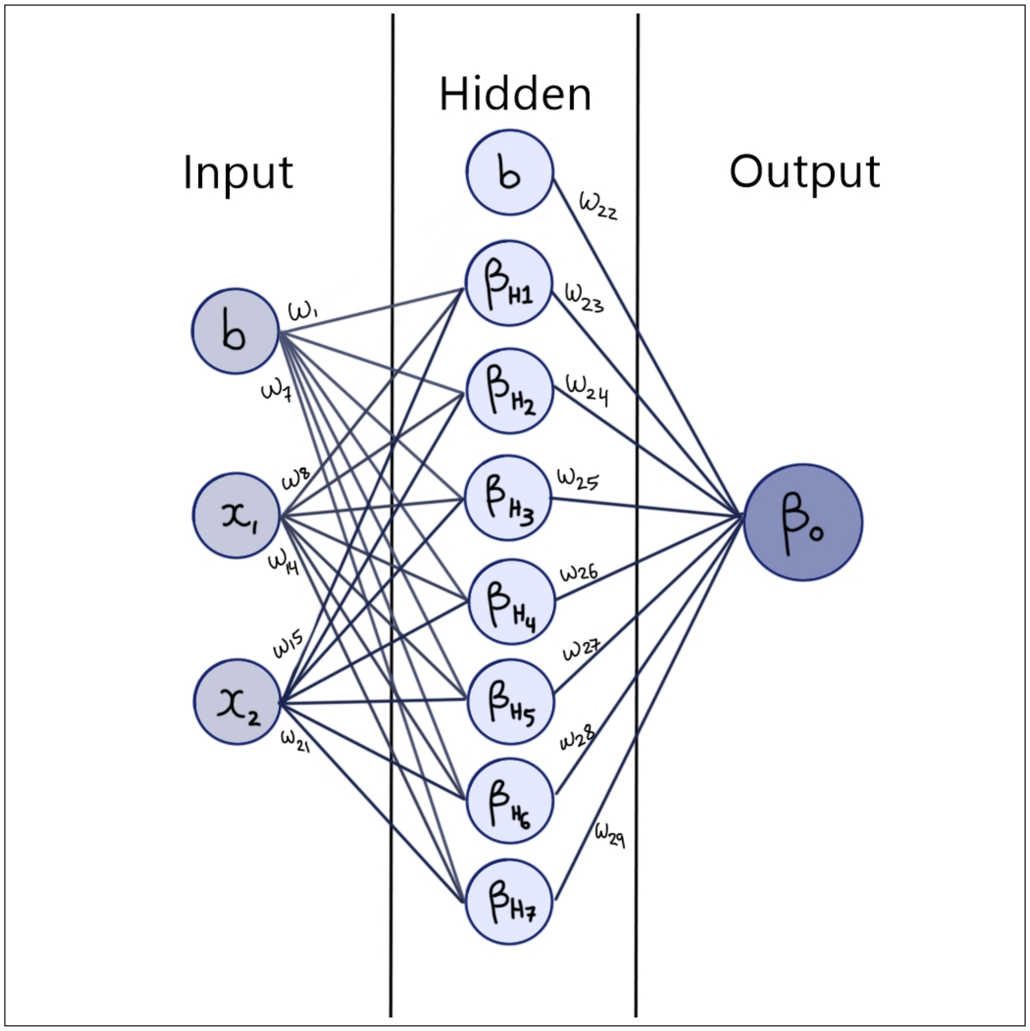


Figure 1: Diagram of neural network - there are 7 hidden layer neurons and an additional bias, marked by 'b'.

1.1 Modelling inputs/outputs

1.1.1 Hidden layer

As illustrated in the diagram, we can define the input into each hidden layer neuron as α_{H_n}, in terms of x_1 and x_2, and the relevant weights linking them to the neuron.

For example, for the first hidden neuron (where w_1 corresponds to the bias):

$$\alpha_{H_1} = (w_1 \times 1) + (w_8 \times x_1) + (w_{15} \times x_2). \tag{1}$$

The relevant activation function I will use is a sigmoid function:

$$S(x) = \frac{1}{1+e^{-x}} \tag{2}$$

Where x is the input into a neuron, that will either be significant enough to activate it or will not meet the threshold for activation.

Therefore, the hidden layer outputs β_{H_n} will be the result of applying the activation function to α_{H_n},

$$\beta_H = S(\alpha_{H_n}) = \frac{1}{1+e^{-\alpha_{H_n}}}. \tag{3}$$

1.1.2 Output layer

There are 8 outer layer inputs including the bias. Therefore, the input into the outer layer is:

$$\alpha_O = w_{22} + w_{23}\beta_{H_1} + w_{24}\beta_{H_2} + w_{25}\beta_{H_3} + w_{26}\beta_{H_4} + w_{27}\beta_{H_5} + w_{28}\beta_{H_6} + w_{29}\beta_{H_7}. \tag{4}$$

The resultant output is found by applying the activation function to these inputs:

$$\beta_O = S(\alpha_O) = \frac{1}{1+e^{-\alpha_O}}. \tag{5}$$

1.2 Defining the loss function

Due to the binary output, I will use a loss function called the Binary Cross-Entropy Loss Function, specific to this case:

$$L = \frac{1}{N}\sum_N \left(c_{true} \times \ln(c_p) + (1-c_{true}) \times \ln(1-c_p)\right) \tag{6}$$

Where N is the number of data points, c_{true} is the true classification (as given by the list of data), and $c_p = \beta_O$ is the predicted classification from a forward pass in the neural network.

In practice, this loss function will need to include an offset, to avoid surpassing the computational limit (when c_p approaches 1 or 0). This offset, ϵ, will ensure the subject of the log is not 0, but will be a very small number so as to not disrupt results.

Therefore, the loss function will actually be implemented as:

$$Loss = \frac{1}{N}\sum_N \left(c_{true} \times \ln(c_p+\epsilon) + (1-c_{true}) \times \ln(1-c_p+\epsilon)\right). \tag{7}$$

1.3 Gradient descent

To correct the allocated weights during the forward and backpropagation of our neural network, we apply the following gradient-descent model:

$$w_{n_{new}} = w_n - \frac{dL}{dw_n}, \tag{8}$$

where $\frac{dL}{dw_n}$ is the derivative of the loss function with respect to the particular weight.

The derivative expressions must be found to implement in the neural network.

Using the chain rule, we can write the derivatives with respect to the output weights w_{O_n} (w_{22} to w_{29}):

$$\frac{dL}{dw_{O_n}} = \frac{dL}{d\beta_O}\frac{d\beta_O}{d\alpha_O}\frac{d\alpha_O}{dw_{O_n}} \tag{9}$$

Similarly, the derivatives with respect to the hidden layer weights w_{H_n} (w_1 to w_{21}) are:

$$\frac{dL}{dw_{H_n}} = \frac{dL}{d\beta_O}\frac{d\beta_O}{d\alpha_O}\frac{d\alpha_O}{d\beta_{H_n}}\frac{\beta_{H_n}}{\alpha_{H_n}}\frac{\alpha_{H_n}}{dw_{O_n}}. \tag{10}$$

The relevant derivatives, using the previously defined functions, are:

$$\frac{dL}{d\beta_O} = -\frac{c_{true}}{\beta_O} + \frac{1-c_{true}}{1-\beta_O} \tag{11}$$

$$\frac{d\beta_O}{d\alpha_O} = \beta_O\left(1-\beta_O\right) \tag{12}$$

$$\frac{d\alpha_O}{dw_{O_n}} = \beta_{H_n} \tag{13}$$

$$\frac{d\alpha_O}{d\beta_{H_n}} = w_{O_n} \tag{14}$$

$$\frac{\beta_{H_n}}{\alpha_{H_n}} = \beta_{H_n}\left(1-\beta_{H_n}\right). \tag{15}$$

Finally,

$$\frac{\alpha_{H_n}}{dw_{O_n}} = 1, x_1 \text{ or } x_2. \tag{16}$$

Therefore, the final expressions for gradients are as set out below.
Output layer weights w_{O_n}:

$$\frac{dL}{dw_{22}} = \left(-\frac{c_{true}}{\beta_O} + \frac{1-c_{true}}{1-\beta_O}\right)\beta_O\left(1-\beta_O\right)\times 1 \tag{17}$$

$$\frac{dL}{dw_{23}} = \left(-\frac{c_{true}}{\beta_O} + \frac{1-c_{true}}{1-\beta_O}\right)\beta_O\left(1-\beta_O\right)\times \beta_{H_1} \tag{18}$$

$$\frac{dL}{dw_{24}} = \left(-\frac{c_{true}}{\beta_O} + \frac{1-c_{true}}{1-\beta_O}\right)\beta_O\left(1-\beta_O\right)\times \beta_{H_2} \tag{19}$$

$$\frac{dL}{dw_{25}} = \left(-\frac{c_{true}}{\beta_O} + \frac{1-c_{true}}{1-\beta_O}\right)\beta_O\left(1-\beta_O\right)\times \beta_{H_3} \tag{20}$$

$$\frac{dL}{dw_{26}} = \left(-\frac{c_{true}}{\beta_O} + \frac{1-c_{true}}{1-\beta_O}\right)\beta_O\left(1-\beta_O\right)\times \beta_{H_4} \tag{21}$$

$$\frac{dL}{dw_{27}} = \left(-\frac{c_{true}}{\beta_O} + \frac{1-c_{true}}{1-\beta_O}\right)\beta_O\left(1-\beta_O\right)\times \beta_{H_5} \tag{22}$$

$$\frac{dL}{dw_{28}} = \left(-\frac{c_{true}}{\beta_O} + \frac{1-c_{true}}{1-\beta_O}\right)\beta_O\left(1-\beta_O\right)\times \beta_{H_6} \tag{23}$$

$$\frac{dL}{dw_{29}} = \left(-\frac{c_{true}}{\beta_O} + \frac{1-c_{true}}{1-\beta_O}\right)\beta_O\left(1-\beta_O\right)\times \beta_{H_7}. \tag{24}$$

Hidden layer weights w_{H_n}:

$$\frac{dL}{dw_1} = \left(-\frac{c_{true}}{\beta_O} + \frac{1-c_{true}}{1-\beta_O}\right)\beta_O\left(1-\beta_O\right)w_{23}\beta_{H_1}\left(1-\beta_{H_1}\right)\times 1 \tag{25}$$
$$\frac{dL}{dw_2} = \left(-\frac{c_{true}}{\beta_O} + \frac{1-c_{true}}{1-\beta_O}\right)\beta_O\left(1-\beta_O\right)w_{24}\beta_{H_2}\left(1-\beta_{H_2}\right)\times 1 \tag{26}$$
$$\frac{dL}{dw_3} = \left(-\frac{c_{true}}{\beta_O} + \frac{1-c_{true}}{1-\beta_O}\right)\beta_O\left(1-\beta_O\right)w_{25}\beta_{H_3}\left(1-\beta_{H_3}\right)\times 1 \tag{27}$$
$$\frac{dL}{dw_4} = \left(-\frac{c_{true}}{\beta_O} + \frac{1-c_{true}}{1-\beta_O}\right)\beta_O\left(1-\beta_O\right)w_{26}\beta_{H_4}\left(1-\beta_{H_4}\right)\times 1 \tag{28}$$
$$\frac{dL}{dw_5} = \left(-\frac{c_{true}}{\beta_O} + \frac{1-c_{true}}{1-\beta_O}\right)\beta_O\left(1-\beta_O\right)w_{27}\beta_{H_5}\left(1-\beta_{H_5}\right)\times 1 \tag{29}$$
$$\frac{dL}{dw_6} = \left(-\frac{c_{true}}{\beta_O} + \frac{1-c_{true}}{1-\beta_O}\right)\beta_O\left(1-\beta_O\right)w_{28}\beta_{H_6}\left(1-\beta_{H_6}\right)\times 1 \tag{30}$$
$$\frac{dL}{dw_7} = \left(-\frac{c_{true}}{\beta_O} + \frac{1-c_{true}}{1-\beta_O}\right)\beta_O\left(1-\beta_O\right)w_{29}\beta_{H_7}\left(1-\beta_{H_7}\right)\times 1. \tag{31}$$

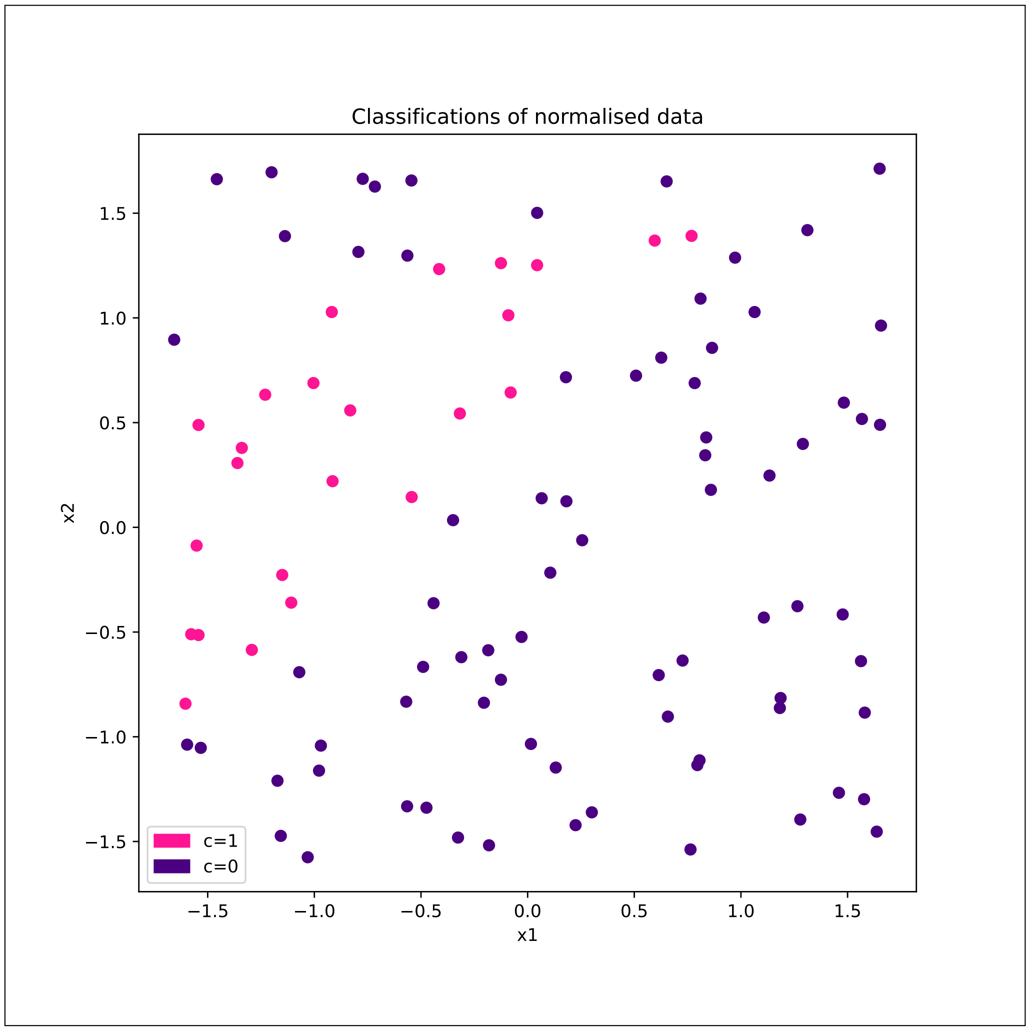
$$\frac{dL}{dw_8} = \left(-\frac{c_{true}}{\beta_O} + \frac{1-c_{true}}{1-\beta_O}\right)\beta_O\left(1-\beta_O\right)w_{23}\beta_{H_1}\left(1-\beta_{H_1}\right)\times x_1 \tag{32}$$
$$\frac{dL}{dw_9} = \left(-\frac{c_{true}}{\beta_O} + \frac{1-c_{true}}{1-\beta_O}\right)\beta_O\left(1-\beta_O\right)w_{24}\beta_{H_2}\left(1-\beta_{H_2}\right)\times x_1 \tag{33}$$
$$\frac{dL}{dw_{10}} = \left(-\frac{c_{true}}{\beta_O} + \frac{1-c_{true}}{1-\beta_O}\right)\beta_O\left(1-\beta_O\right)w_{25}\beta_{H_3}\left(1-\beta_{H_3}\right)\times x_1 \tag{34}$$
$$\frac{dL}{dw_{11}} = \left(-\frac{c_{true}}{\beta_O} + \frac{1-c_{true}}{1-\beta_O}\right)\beta_O\left(1-\beta_O\right)w_{26}\beta_{H_4}\left(1-\beta_{H_4}\right)\times x_1 \tag{35}$$
$$\frac{dL}{dw_{12}} = \left(-\frac{c_{true}}{\beta_O} + \frac{1-c_{true}}{1-\beta_O}\right)\beta_O\left(1-\beta_O\right)w_{27}\beta_{H_5}\left(1-\beta_{H_5}\right)\times x_1 \tag{36}$$
$$\frac{dL}{dw_{13}} = \left(-\frac{c_{true}}{\beta_O} + \frac{1-c_{true}}{1-\beta_O}\right)\beta_O\left(1-\beta_O\right)w_{28}\beta_{H_6}\left(1-\beta_{H_6}\right)\times x_1 \tag{37}$$
$$\frac{dL}{dw_{14}} = \left(-\frac{c_{true}}{\beta_O} + \frac{1-c_{true}}{1-\beta_O}\right)\beta_O\left(1-\beta_O\right)w_{29}\beta_{H_7}\left(1-\beta_{H_7}\right)\times x_1. \tag{38}$$

$$\frac{dL}{dw_{15}} = \left(-\frac{c_{true}}{\beta_O} + \frac{1-c_{true}}{1-\beta_O}\right)\beta_O\left(1-\beta_O\right)w_{23}\beta_{H_1}\left(1-\beta_{H_1}\right)\times x_2 \tag{39}$$
$$\frac{dL}{dw_{16}} = \left(-\frac{c_{true}}{\beta_O} + \frac{1-c_{true}}{1-\beta_O}\right)\beta_O\left(1-\beta_O\right)w_{24}\beta_{H_2}\left(1-\beta_{H_2}\right)\times x_2 \tag{40}$$
$$\frac{dL}{dw_{17}} = \left(-\frac{c_{true}}{\beta_O} + \frac{1-c_{true}}{1-\beta_O}\right)\beta_O\left(1-\beta_O\right)w_{25}\beta_{H_3}\left(1-\beta_{H_3}\right)\times x_2 \tag{41}$$
$$\frac{dL}{dw_{18}} = \left(-\frac{c_{true}}{\beta_O} + \frac{1-c_{true}}{1-\beta_O}\right)\beta_O\left(1-\beta_O\right)w_{26}\beta_{H_4}\left(1-\beta_{H_4}\right)\times x_2 \tag{42}$$
$$\frac{dL}{dw_{19}} = \left(-\frac{c_{true}}{\beta_O} + \frac{1-c_{true}}{1-\beta_O}\right)\beta_O\left(1-\beta_O\right)w_{27}\beta_{H_5}\left(1-\beta_{H_5}\right)\times x_2 \tag{43}$$
$$\frac{dL}{dw_{20}} = \left(-\frac{c_{true}}{\beta_O} + \frac{1-c_{true}}{1-\beta_O}\right)\beta_O\left(1-\beta_O\right)w_{28}\beta_{H_6}\left(1-\beta_{H_6}\right)\times x_2 \tag{44}$$
$$\frac{dL}{dw_{21}} = \left(-\frac{c_{true}}{\beta_O} + \frac{1-c_{true}}{1-\beta_O}\right)\beta_O\left(1-\beta_O\right)w_{29}\beta_{H_7}\left(1-\beta_{H_7}\right)\times x_2. \tag{45}$$

2 Question 2

I trained my neural network on 75% of the data, leaving 25 points as 'test data' to confirm its efficacy. This meant there was sufficient data to develop an accurate model, but enough test data to provide a general assessment of whether the model could predict accurate classifications.

I first plotted the **normalised** data, as shown below.

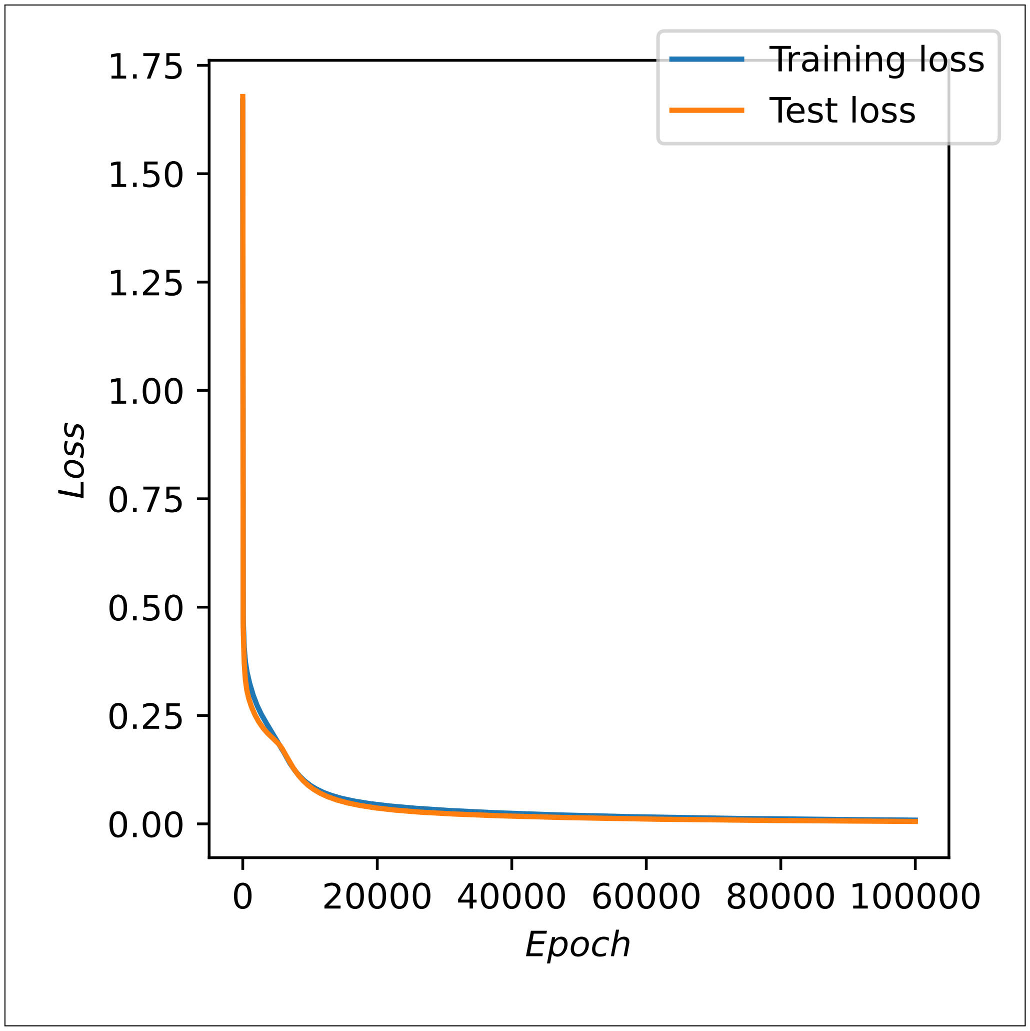


I used a gradient descent method because the system was relatively simple, only having one hidden layer. Therefore, despite being generally slow, it was not too computationally expensive to implement it as a reliable way to ensure a decrease in loss with each epoch.

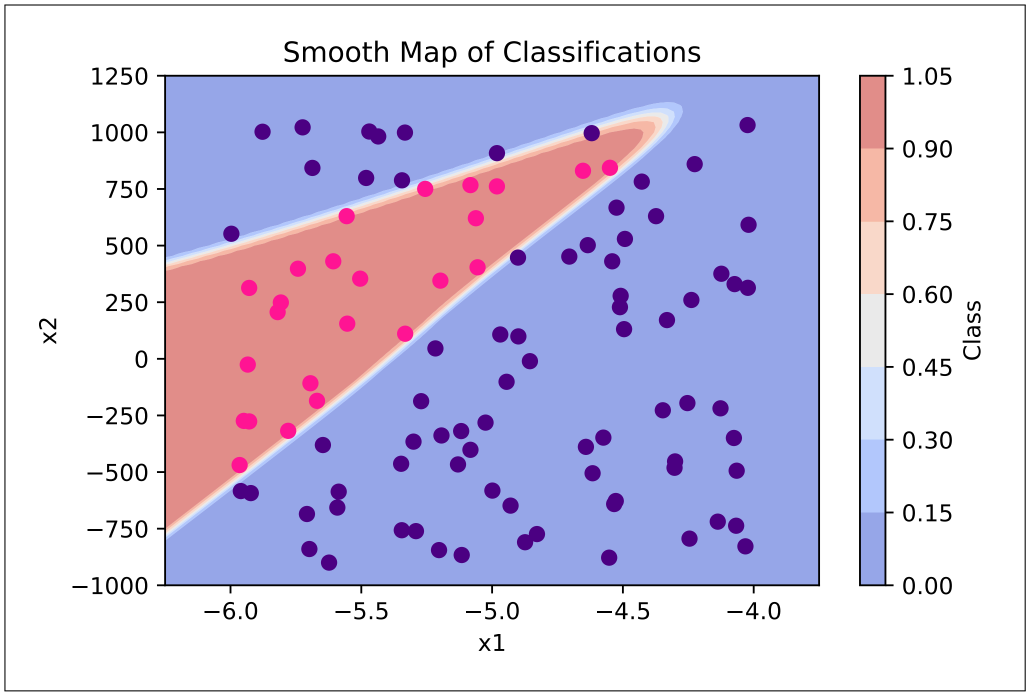
I used a learning rate that was sufficiently small to facilitate a smooth trend in the loss after each epoch, but not so small that it took excessive epochs to reach a steady result. This value was 10^{-3}. I found that the loss reached a relatively stable value after around 100,000 forward and back propagations, so I chose this value for the total number of epochs.

I initialised the all the weights with randum numbers, to ensure the neural network was able to refine a model for the data regardless of the initial values provided.

Below is a plot of the training and test loss as a function of epoch.



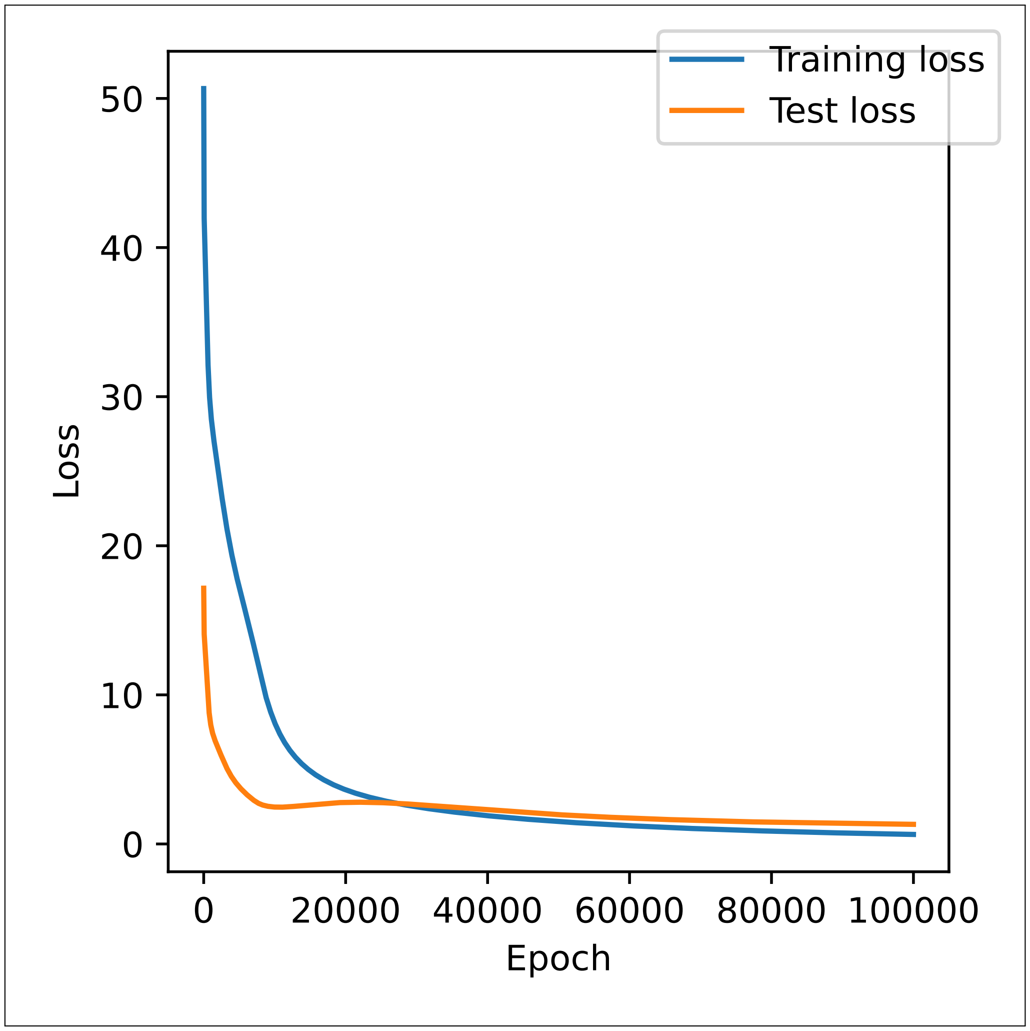
A smooth map of the classifications in x_1 and x_2, for the final model refined by the neural network, is plotted below. Overlaid is the whole dataset - both the training data and the test data.



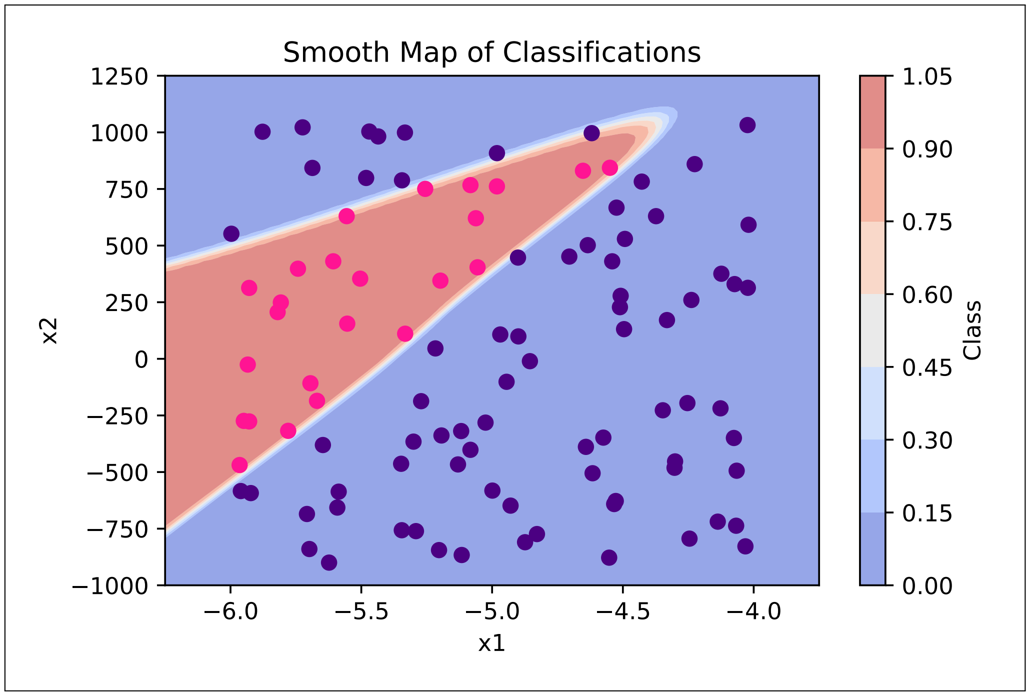
As seen in the figure, the test data nicely fits into the neural network model. The pink data points (classification 1) are all located within the red rea of the map.

3 Question 3

I used PyTorch to make the same plots, shown below.



Training loss
Test loss
Loss
Epoch
0
10
20
30
40
50
0
20000
40000
60000
80000
100000



Smooth Map of Classifications
x1
x2
Class
1250
1000
750
500
250
0
−250
−500
−750
−1000
−6.0
−5.5
−5.0
−4.5
−4.0
1.05
0.90
0.75
0.60
0.45
0.30
0.15
0.00

The main difference is seen in the plots of the losses. My initial losses are much lower than that of the PyTorch model - this may be an artefact of my initial random weight guesses. Further, I have a greater difference in the initial training and test losses, which again may simply be due to differing initialisations between the models. Another difference is that my test loss seem to reach a flatter plateau slightly earlier than with the PyTorch model (at around 40,000).

However, when comparing the net colour-mapped predictions, the two models produce very similar results. Therefore, my manual neural network was rather successful compared to the off-the-shelf framework.

4 Question 4

Activation function: The activation function acts as an analytic threshold for the output of a neuron to be significant. For example, a sigmoid function gives greater weight to values beyond a certain value, allowing for neurons that have a small output to be ”turned off”.

Back propagation Back propagation is a method of applying a neural network where the weights of neuron (and bias) outputs are varied according to the difference between predicted and actual output values. Specifically, by propagating the gradient of the loss function back through the neurons, the weights can be adjusted towards a lower gradient according to a set ”learning rate”.

Learning rate: The learning rate dictates the allowed adjustment to neuron weights during back propagation, while the neural network is training. The higher the learning rate, the greater the adjustments to the weights with each epoch. However, there is also an increased risk of overshooting, so it must be balanced.

Vanishing gradients: Where one gradient in the chain of gradients through the network is close enough to 0, this can result in the multiplication of other terms with a value that is effectively 0. Gradients can therefore ”vanish”, and not allow for the appropriate adjustment of weights. For example, the gradient of the sigmoid function disappears as the values approach both 0 and 1.

Weights and biases: For the neural network to use linear connections to model non-linear relations, multiple linear connections are attributed different weightings. In terms of the specific calculations, each input into each neuron has a weighting parameter than modifies the contribution from that line of input. Back propagation adjusts these weights to achieve the best representation of the relevant model.

Loss: During training, a ’training loss’ function indicates how close the iterations of back propagation are to resulting in the ’true’ values for the neuron weights. Once the training loss experiences little change with each epoch, the neuron can be taken to have been sufficiently trained.

Validation loss is the loss corresponding to a set data that is only subject to the forwards process, such that with each epoch, the ”new” weights are applied to a (blind) dataset. This effectively tests the reliability of the model, and tests that our neural network is not overfitting just the training data. It allows us to estimate the model’s effectiveness on other unseen data.

Finally, testing loss is the single-instance loss when the final trained model is applied to a set of ”unseen” data. This is the ultimate test of whether a model is appropriate to describe an entire process (i.e. whether it can predict the output of any value with relative accuracy).

Convolution kernels: Convolution kernels are used in convolutional neural networks (CNNs). They are filters used to analyse a feature map (e.g. an image) which probe multiple pixels in one application. Each pixel is assigned a ”weight” (like in our neural network example), and the net value of the pixel is the sum of these weights from the original image. As the kernal slides over the map, it reduces the parameters of the system while still capturing patterns in the image within parameter space.

Pooling: Pooling can be used to reduce the spatial size of a featuer map before a convolutional kernel is applied. It is another filter, one that extracts the maximum value of spatially surrounding inputs to the relevant pixel. The result is a spatially smaller map, but that still contains information about what exists in different areas of parameter-space.

Data augmentation: Data augmentation is a process taken to make a trained neural network more robust, by changing aspects of the training data. Data augmentation can include translations of the data, adding noise, or changing single points. The increased variety and obstacles in the data will result in a better performing neural network that is not generally susceptible to discrepencies in future data.

Softmax layers: When a neural network is implemented for classification (and not continuous predictions), it is useful to implement a softmax function in the output. This function effectively converts input values (the effective ”weights” of different categories) into a probability. It does this by finding the exponent of each number, and dividing by the sum of all exponentialvalues. It can be useful where the desired output is a probability of belonging in a certain category.