# Module-5

Antara Paul, Labani Roy

## Activity 1: Perceptron from scratch

For inputs:

$$\mathbf{a}^{(0)} = \mathbf{X} \tag{1}$$

the linear combinations with weights $\mathbf{W}$ and biases $\mathbf{b}$ for each hidden layer is given as:

$$\mathbf{z}^{(i)} = \mathbf{W}^{(i)}\mathbf{a}^{(i)} + \mathbf{b}^{(i)}, \quad \forall i \in \{0, \ldots, L-1\} \tag{2}$$

and then transformed using an activation function. The last layer either has no activation function or a sigmoid function depending on whether we want to do a regression or a classification. For regression:

$$\mathbf{a}^{(i)} = \begin{cases} \tanh(\mathbf{z}^{(i)}), & \text{if } i < L-1 \text{ (hidden layers)} \\ \mathbf{z}^{(i)}, & \text{if } i = L-1 \text{ (output layer)} \end{cases} \tag{3}$$

For classification:

$$\mathbf{a}^{(i)} = \begin{cases} \tanh(\mathbf{z}^{(i)}), & \text{if } i < L-1 \text{ (hidden layers)} \\ \sigma(\mathbf{z}^{(i)}), & \text{if } i = L-1 \text{ (output layer)} \end{cases} \tag{4}$$

where:

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{5}$$

Finally:

$$\mathbf{y}_{\text{pred}} = \mathbf{a}^{(L-1)} \tag{6}$$

Next, for the training of the neural networks, we update the weights and biases for certain number of epochs using:

$$\mathbf{W}^{(i)} \leftarrow \mathbf{W}^{(i)} - \eta \cdot \frac{\partial L}{\partial \mathbf{W}^{(i)}} \tag{7}$$

$$\mathbf{b}^{(i)} \leftarrow \mathbf{b}^{(i)} - \eta \cdot \frac{\partial L}{\partial \mathbf{b}^{(i)}} \tag{8}$$

where $\eta$ is the learning rate and

$$\frac{\partial L}{\partial \mathbf{W}^{(i)}} = \frac{1}{m}\left(\frac{\partial L}{\partial \mathbf{z}^{(i)}}\mathbf{a}^{(i-1)^\top}\right) \tag{9}$$

$$\frac{\partial L}{\partial \mathbf{b}^{(i)}} = \frac{1}{m}\sum \frac{\partial L}{\partial \mathbf{z}^{(i)}} \tag{10}$$

The derivative $\frac{\partial L}{\partial \mathbf{z}^{(i)}}$ in turn depends on the derivative of the tanh or sigmoid function depending upon if it's a regression or a classification problem.

The regression using the test data from the sine plus noise data is given in Fig. 1 and the decision boundary for the classification problem from last module is shown in Fig 2.

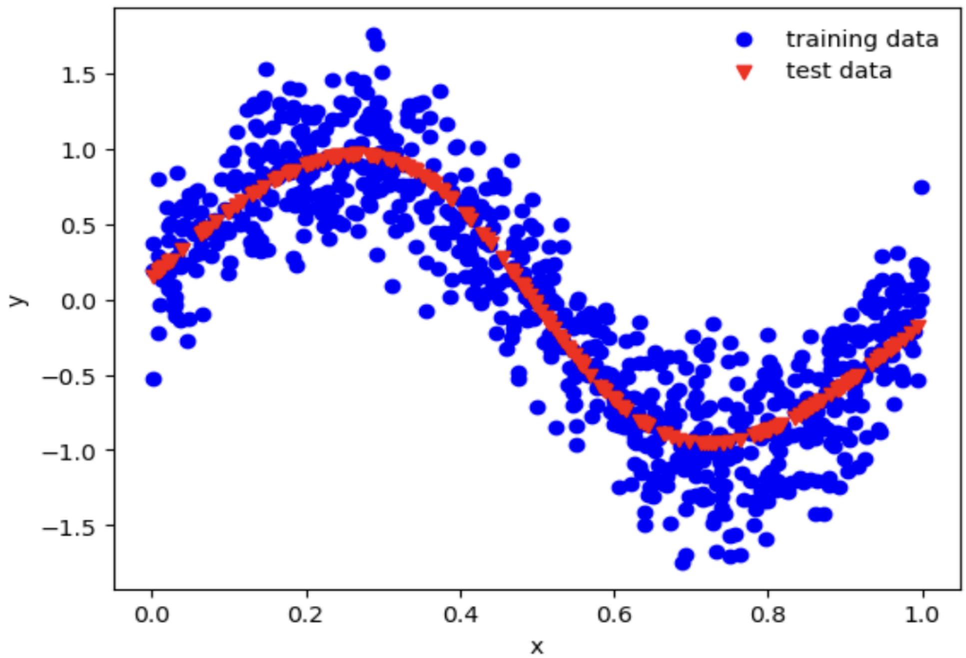


Figure 1

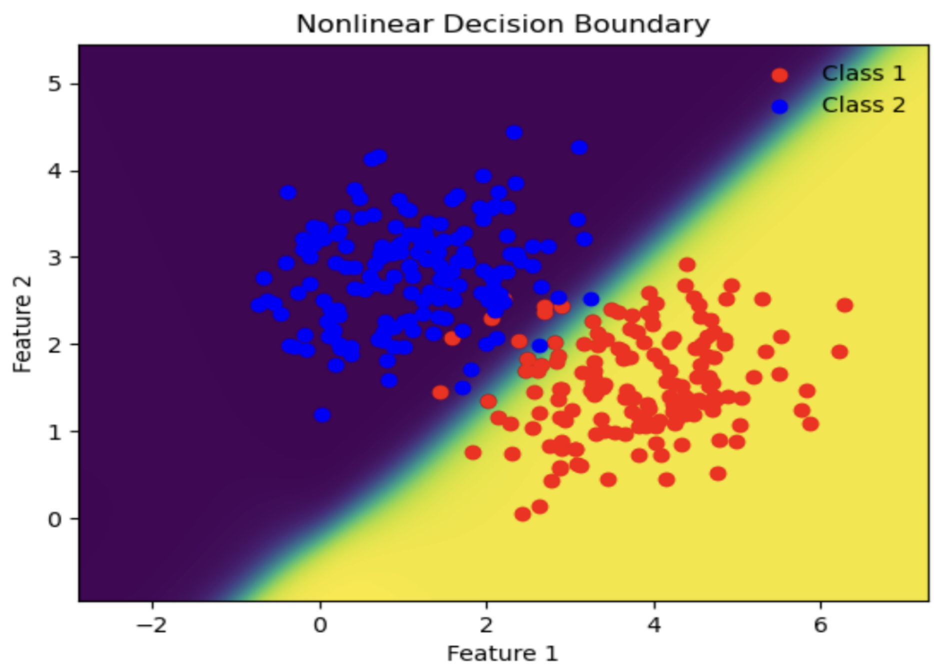


Figure 2

## Activity 2: Varying the parameters

We varied the parameters for different number of data points. We noticed that as the number of nodes in a single hidden layer is increased, the r2 score increases and gets closer to 1. Moreover, for the same number of nodes in each layer and higher number of hidden layers also gives a better r2 score. The r2 score is calculated using `scikit-learn` and for 1000 data points. However, if the number of nodes in each layer or the number of hidden layers is too high, it results in over-fitting. The overfitting is more evident in the case of high number of hidden layers, as shown in Figs. 3 and 4. The plotting was done for 10 data points using the MLP algorithm from scratch.

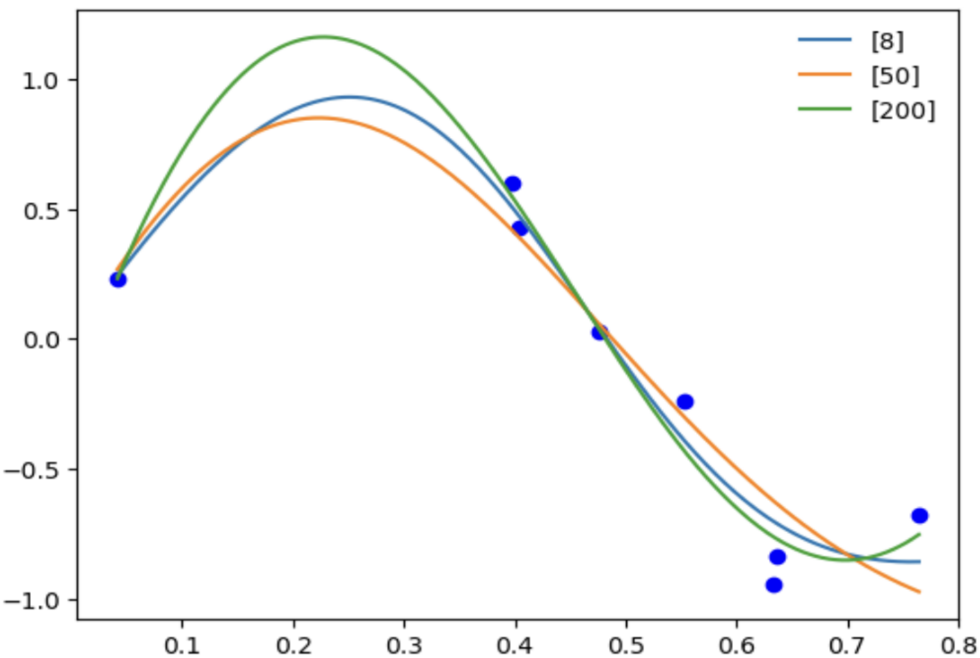


Figure 3

The increase in number of epochs in both cases gave better r2 score and a better fit. However, we could not go beyond 50000 epochs as our computer RAM could not handle it.

## Activity 3: IRIS data classification

From the IRIS data set, two different features of flowers (petal and sepal length) of each flower species (Setosa and Versicolor iris species) are stored in the $X = (x_1, x_2), n \times 2$ matrix, where $n =$ number of samples in each class (class-1: Iris-setosa, class-2: Iris-versicolor). We consider $Y_{data}$ to be $+1$ and $-1$ for class-1 and class-2, respectively.

Using 2-layer Perceptron algorithm, $Y_{predict}$ is determined for each train samples. Then, this trained algorithm is used to classify the test dataset. We got only 2 misclassification while comparing our $Y_{test,predict}$ with $Y_{test,data}$.

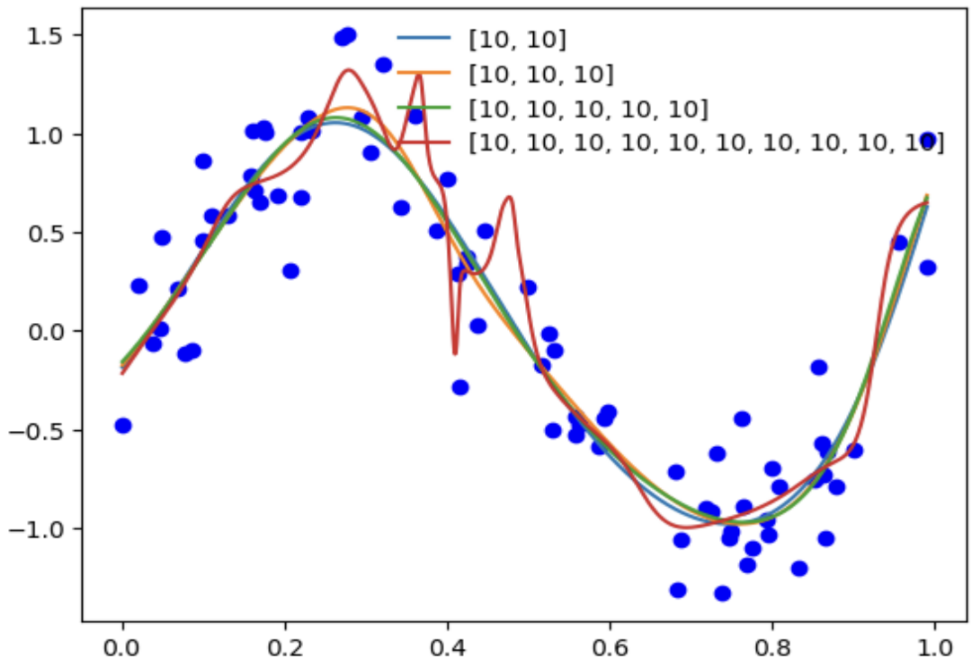


Figure 4

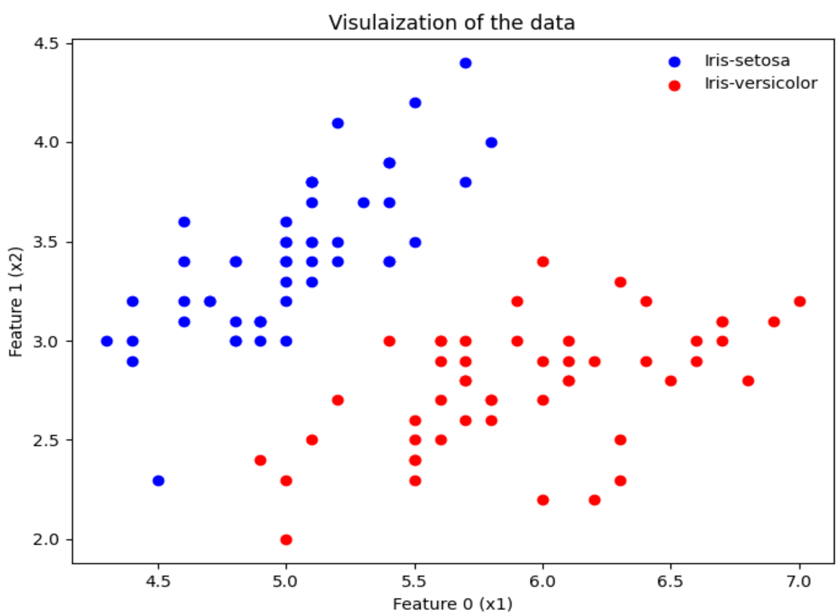


Figure 5