In this lab, we will take a closer look at how non-linear models can handle cases linear models cannot. You will be given a logistic regression model as a baseline and you will implement a simple feed forward network. We will have binary data that is not linearly separable and you will see that logistic regression fails to predict it correctly because it separates the space linearly, whereas the FFN will succeed.

Before you begin

1. Create a new python project and a virtual environment for it in the usual way.
2. Add the starter code in *Lab3.py* and the file *plotting_functions.py* to your project.
3. Make directory *plots* in your project folder and inside *plots* make subdirectories *data* and *loss*.

1 Starter Code

Have a look at Lab3.py: there you have some predefined functions and classes: the *generate_data()* function for generation of not linearly separable data, the *cost_CE()* function to calculate cross-entropy loss, the *LogisticRegression* class you will refer to when implementing the FFN; some parts of the FFN are already written, as well as a part of the *main()*. Please do not change the written parts of the code. However, you may play around with parameters such as size of the dataset, learning rate etc.

2 Tasks

2.1 Weights Initialization

Implement the function *init_weights()* of the class *FeedForwardNetwork*. You must have two matrices of weights: from input layer to hidden layer and from hidden layer to output layer. The model must have two biases: one for each weight matrix. Initialize weights matrices with random weights (not zeros!) and zero biases.

2.2 Forward Pass

Implement forward pass in the *forward()* function of the FFN. Refer to the *LogisticRegression* class and the comments in the code.

2.3 Training Loop

Implement the *train()* function to make the model learn iteratively. In the training loop, on each epoch, you should conduct forward pass and backpropagation and then update weights.

2.4 Predictions

Implement the *predict()* function to predict labels of test data.

2.5 Putting Together

In the *main()* function, a dataset is generated and split into train and test splits and then a logistic regression is trained and used for predictions. Your task here is to do the same with the FFN: train on the train split, predict the labels of the test split and visualize the predictions.

At the first launch, please specify learning rate equal to 0.5 and number of training epochs equal to 2000. After that you may play around with parameters.