# Lab 3

# Activation Functions, MLP, and BP

**Objective:**

This lab aims to:

1. Understand the different activation functions used in NN.
2. Train and test a neural network on a real and complex dataset.
3. Train and test a network for a big dataset of images.

In this lab, the following packages will be used:

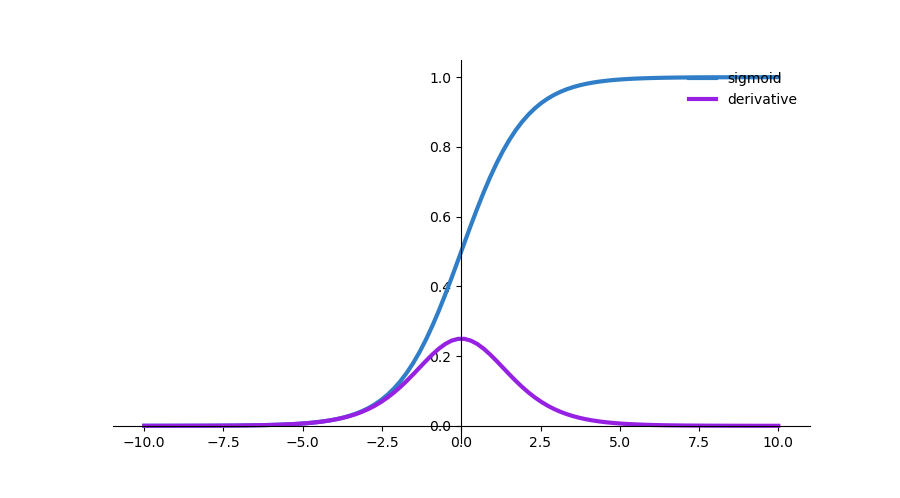
1. Scipy
2. Numpy
3. Matplotlib
4. Pandas
5. Scikit-learn

Be sure that all these packages are installed in your conda environment. Use the file “test\_machine\_learning\_packages.py” to check if all required packages are installed in the conda environment.

# Task – 1: Activation Functions

Different activation functions are used in NNs. This part of the lab is to implement the different activation functions and their derivatives.

* Write Python code to produce the following activation functions and their derivatives.
  + Sigmoid activation function and its derivative.
  + Tanh activation function and its derivative.
  + ReLU activation function and its derivative.

 Figure 1: Sigmoid activation Function and its derivative

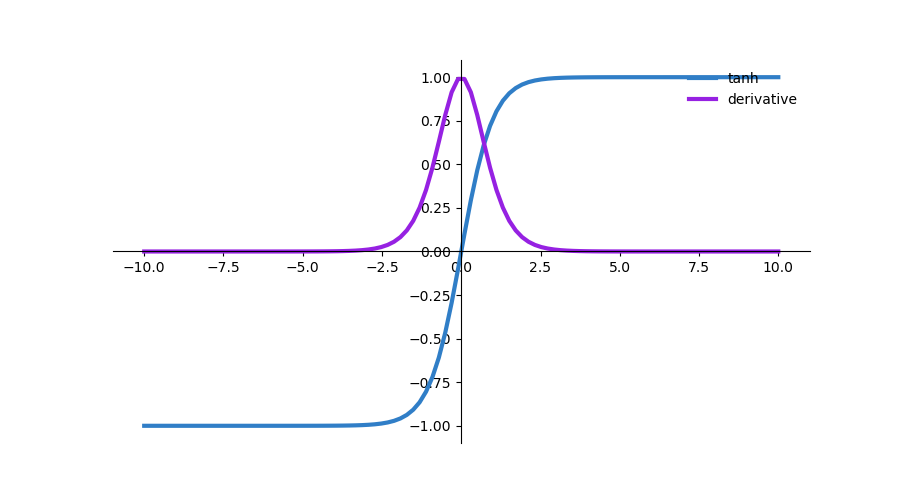


Figure 2: Tanh Activation Function and its derivative

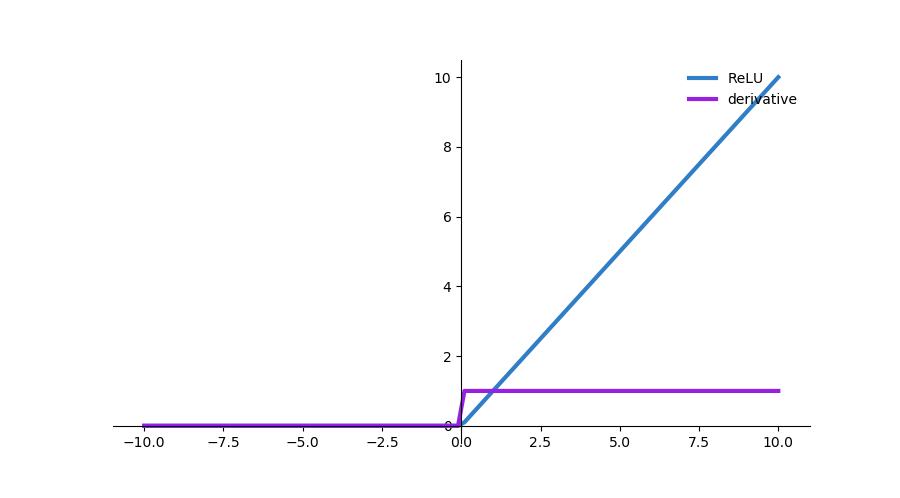


Figure 3: ReLU Activation Function and its derivative

* Define an array X= [0,…,21]and pass it to the sigmoid function to calculate the corresponding values of the function (Y). Plot the results of this operation as in the following.

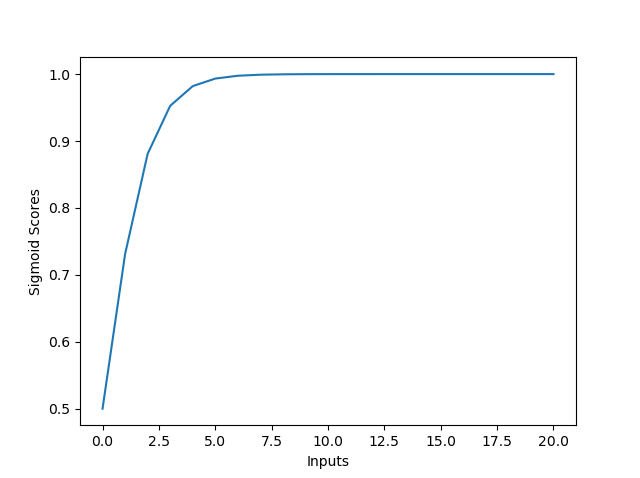


Figure 4: Sigmoid Activation Function and its response to an array.

* Repeat the former task for the softmax function. What is the difference between the results of the softmax and sigmoid plots? Why does this happen?

# Task – 2: Implementing XOR

A simple definition of a neural network is given in the file called (NN\_definition.py).

1. Write a function of the sigmoid activation function and another one for its derivative.
2. Create the XOR truth table and plot the points.
3. Separate the XOR truth table into inputs and outputs and pass them to the “Net” function together with the other parameters.
4. Document the classification parameters.
5. Repeat steps 1-4 to solve the following truth table and document the changes you made on the network to solve this problem and the results.

A B C Output

T T T F

T T F T

T F T T

T F F T

F T T T

F T F T

F F T T

F F F F

After completion of the task, you must be able to answer the following questions:

* Address and discuss the various complications (if any) that you have encountered in the preparation. What was the easy part?
* How many of the examples are correctly classified?
* Do you get different classification results by tuning the parameters? Discuss the influence of parameters on the classification rate.

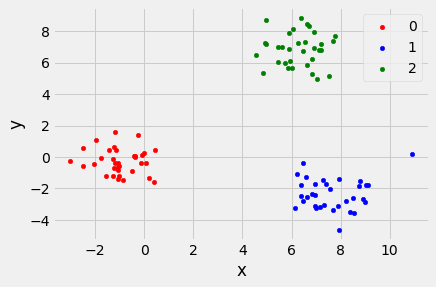
# Task – 3: Dataset Creation and classification

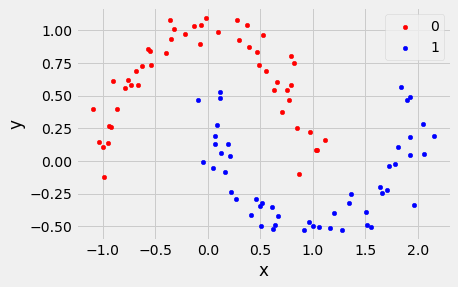
Different datasets can be generated using sklearn.

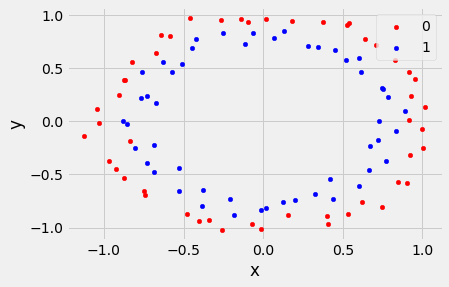
1. Use sklearn.datasets and import make\_blobs
2. Generate a dataset which consists of 100 patterns, 3 classes, standard deviation = 1, and 2 features. Plot this dataset with a different color for each class.
3. Use this dataset to train and test the “Net” used in Task 2
4. Repeat 1-3 by importing “make\_moons” to create a dataset of 100 points and noise = 0.1.
5. Repeat 1-3 by importing “make\_circles” to create a dataset of 100 points and noise =0.1.
6. Report the results of the three different datasets.

After completion of the task, you must be able to answer the following questions:

* Address and discuss the various complications (if any) that you have encountered in the preparation. What was the easy part?
* How many of the examples are correctly classified?
* Do you get different classification results by tuning the parameters? Discuss the influence of parameters on the classification rate.







# Task – 4: Working with Mnist dataset

Mnist dataset ( <http://yann.lecun.com/exdb/mnist/> ) is database of handwritten digits, has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image.

The original black and white (bilevel) images from NIST were size normalized to fit in a 20x20 pixel box while preserving their aspect ratio. The resulting images contain grey levels as a result of the anti-aliasing technique used by the normalization algorithm. The images were centered in a 28x28 image by computing the center of mass of the pixels and translating the image to position this point at the center of the 28x28 field.

Your task:

1. Import from sklearn.datasets, fetch\_openml.
2. Use fetch\_openml() to fetch the mnist dataset, “mnist\_784”, version = 1.
3. Store the feature data and target data in appropriate variables.
4. Plot samples of the training images
5. Cast your target vector as an integer vector.
6. Create training and test sets from the data and target
7. Reshape each element in your training and test inputs so that they have shape, (784, 1).
8. Convert the training results into a 10-dimensional unit vector with a 1.0 in the jth position and zeroes elsewhere. This is used to convert a digit (0...9) into a corresponding desired output from the neural network.
9. Use the python file (Network.py) given to you to build a feedforward network to handle this dataset.
10. Change the parameters of the network by trying different number of neurons and other parameters.
11. Use the trained network to predict different numbers.
12. Show the input number and the predicted one as image. Use the reshape function for this purpose.

**Note That:**

1. **The report should include all the figures produced by the code.**
2. **All codes should be complete and standalone (I do not need to change anything in order to run them).**
3. **The report should include an analysis of the results.**