# Machine Learning CW3

**Objective.** The goal of this coursework is to gain a hand on experience with Multilayer Perceptrons (MLPs) by implementing them from scratch and applying them to a simple classification task.

In the final CW, we will use the outcome of this CW as well as other machine learning algorithms to solve more realistic problems.

*Problem setting.* We will use a simple binary classification problem with two-dimensional inputs: You will be given a set of labeled training data points $X_{\text{tr}} = \{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_N, \mathbf{y}_N)\} \subset \mathbb{R}^2 \times \{0, 1\}$ plus test inputs $X_{\text{te}} = \{\mathbf{x}'_1, \ldots, \mathbf{x}'_M\} \subset \mathbb{R}^2$. Each input in the training set is accompanied by the corresponding class label (either 0 or 1).

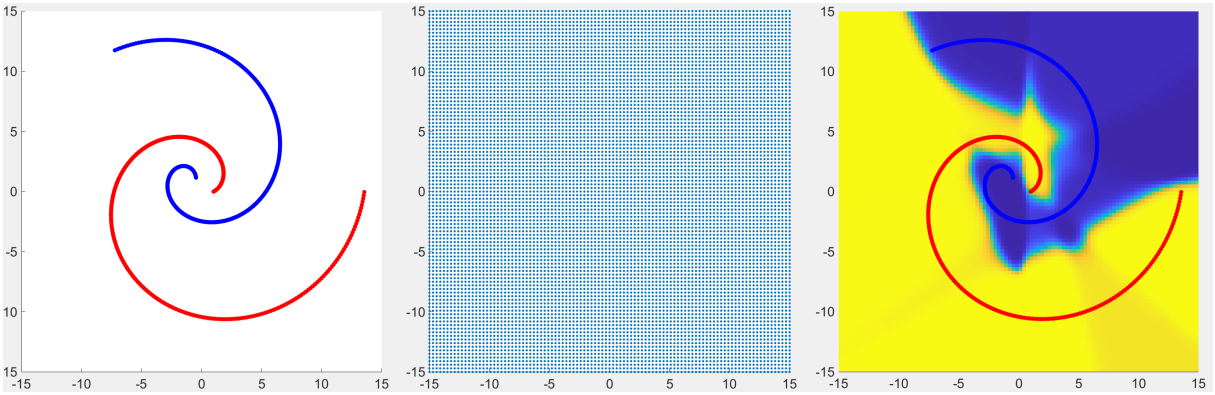

Figure 1: A visualization of our toy classification problem: (left) training set $X_{\text{tr}}$ (class 0 and 1 are respectively highlighted in red and blue); (middle) test set $X_{\text{te}}$ presented as a regular grid sampling of $[-15, 15] \times [-15, 15]$, (right) outputs of the trained MLP on $X_{\text{te}}$ overlaid with the training set $X_{\text{tr}}$.

**Task.** Our task is to train an MLP on $X_{\text{tr}}$, apply the trained MLP to the test set $X_{\text{te}}$, and visualize the corresponding test results. Our data are visualized in Fig. 1: The training set consists of 630 pairs of inputs and the corresponding ground-truth outputs: In Fig. 1(left), the location of each dot represents the corresponding training input ($x^1, x^2$-coordinate values) while the class labels are highlighted by color. The accompanying 'Trn.txt' file contains this training set: For each line, the first two columns represent an input $\mathbf{x} = [x^1, x^2]^\top$ and the third column provides the corresponding class label (0 or 1). The test set consists of 10,200 data instances and they are provided in 'Tst.txt' file where two columns represent a test input per line.

**What to hand on?** Please submit your code and a single-page report:

1. Your code should implement the training and test phases of an MLP. Please use a fully-stochastic gradient descent for training. The submitted code package should include a python script 'mymlp.py' which 1) reads $X_{\text{tr}}$ file from the directly where the script is executed, 2) trains an MLP on $X_{\text{tr}}$; 3) reads $X_{\text{te}}$ from the same directory; and 4) applies the trained MLP to $X_{\text{te}}$. If your code package contains more than one file, please make sure that all four steps mentioned are performed by executing 'mymlp.py': 'mymlp.py' can use other scripts. You can use basic math libraries e.g. NumPy. However, you should implement the training steps of MLP from scratch. In particular, the calculation of the error gradient for back-propagation has to be implemented by yourself. If you use automatic differentiation e.g. provided by TensorFlow, you will get 0 mark for your code.

2. Your report should 1) specify the hyperparameters used in your experiments (e.g. the number of layers, number of neurons in each layer, and learning rate) and 2) visualize the outputs of the trained MLP on the test set. This visualization will look similar to the third column of Fig. 1 while the shapes of the decision boundaries formed by the trained MLPs might vary depending on the choice of the hyperparameters.

Please submit a single zip file that contains your code and a pdf document of your report. Please format the submission as in **'StudentID_Name.zip'**, e.g. 20211234_KwangInKim.zip. If your submission does not meet this file name requirement, the final mark will become 80% of the initial mark.

**Grading.**

1. Code (70/100): We will simply check if your code runs as specified. If your code sucessfully implement the training and testing phases of an MLP and it meets all requirements listed in the previous paragraph, you will earn full marks.

2. Report (30/100): We will check if your report appropriately visualizes the outputs obtained by the trained MLP and if all hyperparameter values are specified.