1 Introduction

In this assignment, you are expected to make practice on applications of neural networks to cope with a data mining task (i.e., classification). By doing so, you will learn underlying phenomenon behind neural networks in solving different complex problem types. In addition, you will analyze the effects of hyperparameters given to construct a network structure.

2 Single-layer Perceptron

Perceptron networks are single-layered networks activated by a Sigmoid (or logistic) function to produce an output. At the end of the learning phase, it yields a linear model. Therefore, it targets problem where the data objects are linearly separable such as 'AND', 'OR', and 'NOT'. However, it performs insufficient performance on nonlinear problems like 'XOR', 'XNOR'. The structure of a single-layered perceptron network with three input neurons is demonstrated in Figure 1. Here, X represents a feature vector in which the first (i.e., $x_0 = 1$) corresponds to the *bias*, while θ represents a weight vector. $h_\theta(X)$, however, is the activation applied output value ranging from 0 to 1.

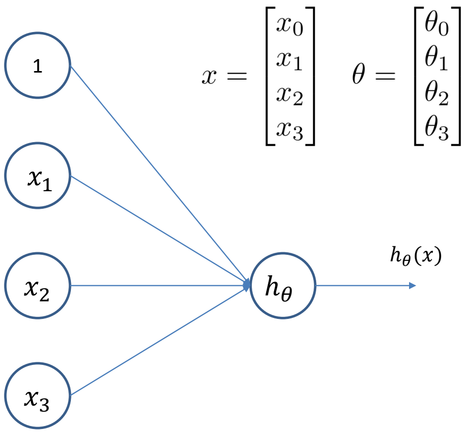

Figure 1: Single-layer perceptron network with three inputs.

2.1 Classification task

The first phase with single-layer perceptron is to use it on classification task and to analyse its performance with varying conditions. Refer to the instruction list below to complete this task.

1. Generate a toy/fictional n-dimensional dataset (i.e., $D \mid D \in \mathbb{R}^n$) having m tuples for binary-class **classification** task.
2. Split D such that randomly selected 70% tuples are used for training while 30% tuples are used for testing.
3. Apply single-layer perceptron network solver to handle D.

Table 1: The effectiveness and efficacy of single-layer perceptron with respect to the varying parameter settings.

#	Parameter setting for *100* iterations: Tuple Size (m)	Dimension Size (n)	Observations*: Training Time (in ms)	Error (cost)
a.	10,000	100		
b.	10,000	1,000		
c.	100,000	100		
d.	250,000	100		
#	**Parameter setting for *500* iterations: Tuple Size (m)**	**Dimension Size (n)**	**Observations*: Training Time (in ms)**	**Error (cost)**
e.	10,000	100		
f.	10,000	1,000		
g.	100,000	100		
h.	250,000	100		

*:**average of ten runs.**

4. For each parameter setting given in Table 1, repeat the previous steps for **ten times**. Then fill each row of the Table 1 with your observations given that parameter setting.

5. Discuss the measurements in your report.

2.2 Visualization of decision boundary

The second step with single-layer perceptron is to apply single-layer perceptron network on solving a typical classification problem to visualize 3-D decision boundary. To accomplish this task follow the instructions below:

1. Generate a binary-class dataset (D) for classification task.

2. There should be at least 500 tuples and three features of which two features are informative to the ground truth vector.

3. Split D such that randomly selected 70% tuples are used for training while 30% tuples

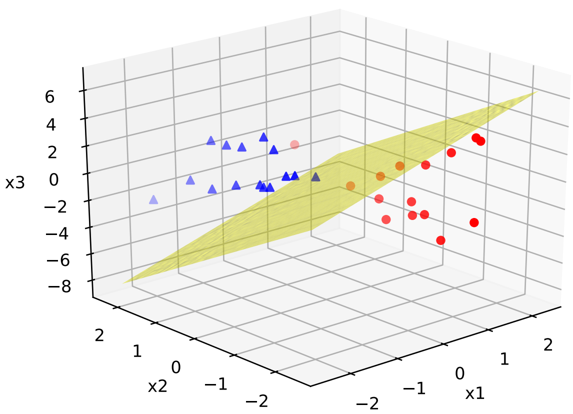

Figure 2: A 3-D decision boundary on binary-class classification problem.

are used for testing.

4. Apply single-layer perceptron network to fit a model on D.
5. Plot testing objects and hypothesis plane on 3-D surface as shown in Figure 2.

3 Multi-layer Perceptron

Multi-layer perceptron (fully connected type neural network) is a stacked version of single-layer perceptron. Unlike to the single-layer perceptron, it is also used to solve nonlinear problems. Activation functions defined in the layers enable multi-layer perceptron networks to yield a nonlinear model. At least three layers take part in this kind of networks. The first is called as *input* layer where the data tuples are given, the next is called as *hidden* layer where progress the information taken from the input layer (or from the previous hidden layer) to the following hidden layer or to the output layer. The last layer, however, is *output* layer that gives the predicted value. A three layered (one hidden layer) neural network is given in Figure 3.

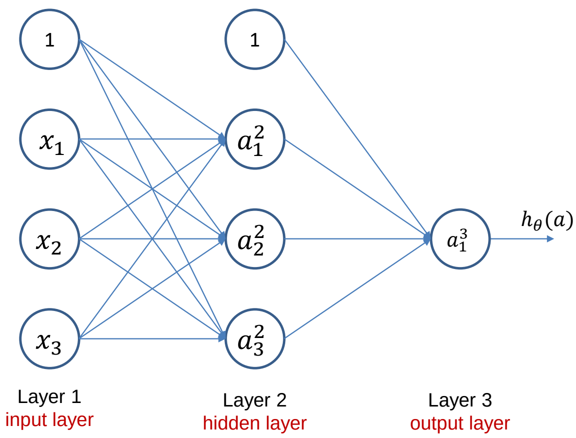

Figure 3: Multi-layer perceptron network with one hidden layer.

3.1 Error convergence with multi-layer perceptron

In this task your are expected to apply multi-layer perceptron to obtain error values throughout the training. To do that, follow the instructions in the following list:

1. Load *digit* dataset (D).
2. Split D such that randomly selected 70% tuples are used for training while 30% tuples are used for testing.
3. Apply multi-layer perceptron network (one hidden layer with 50 neurons) with 100 iterations to handle D.
4. Plot error values as a function of iteration as shown in Figure 4.

5. Discuss your observation.

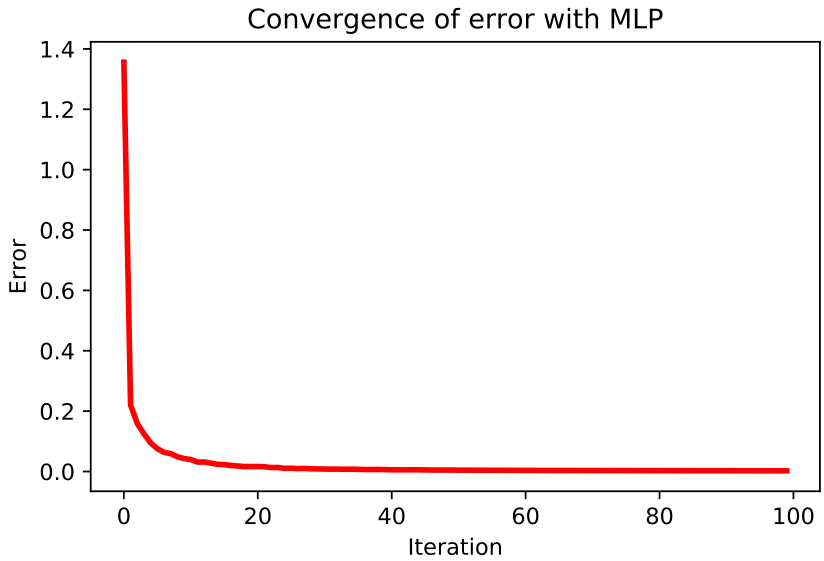

Figure 4: An example convergence plot for error values.

3.2 Effects of multi-layet perceptron structure on train & test scores.

In addition to the convergence analysis with respect to the error, you are also expected to analyse train and test scores (accuracy) as a function of hidden layer size and neurons. To do that, follow the procedure explained below:

1. Load *digit* dataset (D).
2. Split D such that randomly selected 70% tuples are used for training while 30% tuples are used for testing.
3. Apply multi-layer perceptron network with one to H hidden layer size to handle D.
4. The structure of perceptron networks should be as follows:
   - For a network $h \mid 1 \leq h \leq H$ the number of neurons in hidden layers is $2^h, 2^{h-1}, \ldots, 2^1$ from the first to the last hidden layers. A demonstration on this structure is illustrated in Figure 5.
5. Plot *score* (accuracy) values as a function of hidden layer size as shown in Figure 6.

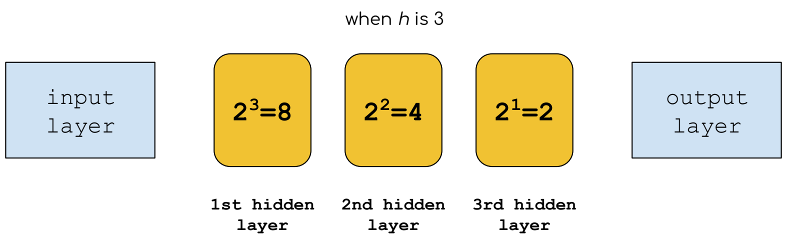

Figure 5: Demonstration for the structure of multi-layer perceptron

Figure 6: An example convergence plot for error values.

Notes

- Your source code should be designed as **easy-to-follow**. **Place comment** in it as much as possible. **Separate each task** through apparent patterns.
- Use LaTeX to prepare your reports. Include the observation tables here to your report. Once again, filled and signed declaration form should be first page of your report. **Reports must not exceed 5 pages in total**.
- **Do not miss** the deadline.
- **Save your work** until the end of this semester.
- The assignment must be **original**, **individual work**. **Duplicate or very similar assignments are both going to be considered as cheating.**
- You can ask your questions via **Piazza** (https://piazza.com/mu.edu.tr/fall2020/ceng3521) and you are supposed to be aware of everything discussed in Piazza.
- You will submit your work on CENG3521 course page at https://dys.mu.edu.tr with the file

hierarchy as below[2]:

→ <student id>.zip
  → Assignment2.py
  → Report2.pdf

[2] do not place any file into a directory. Just compress all the files together.