Iran University of Science & Technology

School of Computer Engineering

Assignment #1

Neural Networks

BY: DR. Naser Mozayani, Fall 2024

Teaching Assistants:

Nafiseh Ahmadi

Due: 1403/08/04

Contents

Notes

1. Submit the answers in a complete PDF file and the code for the questions in the .ipynb format (including the notebook cell outputs) in a compressed file named HW1_StudentID.zip by the specified deadline.
2. A total of 72 hours of delay in submitting the answers is allowed across all projects. After that, for each additional day of delay, 10% of the score will be deducted.
3. If a student submits the project earlier than the deadline and achieves 75% of the score, up to 24 hours will be added to their allowable delay time.
4. The maximum delay for submitting each assignment is 4 days, and after 4 days, submission will not be accepted.
5. It is important to note that the explanation of the code and the obtained results must be included in the PDF file. Code without a report will result in a score deduction.
6. The evaluation of the assignment will be based on the correctness of the solution and the completeness and accuracy of the report.
7. Assignments must be completed individually, and group work on assignments is not allowed.
8. Please allocate sufficient time for the assignment and avoid leaving it until the last days.
9. You can ask your questions in the relevant group.

good luck.

Problem 1

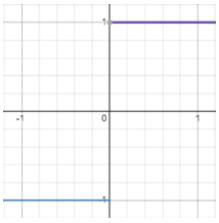
In this question, you need to design an ADALINE neuron that can accurately solve the AND gate. For this problem, the learning rate is set to 0.05, and the bias is considered to be 1. The initial weights are as follows:

- b = 0.1
- w1 = 0.2
- w2 = -0.1

The purpose of this exercise is to explain the step-by-step process of constructing an ADALINE neuron that can correctly learn the AND gate by updating its weights. The training process consists of three iterative steps, and in each step, the weights are updated. Additionally, one of the following activation functions will be used in each step.

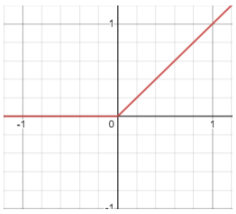
1. Step Function (Hard Limiter):

$$out = \begin{cases} 1 \; if \; y >= 0 \\ -1 \; if \; y <= 0 \end{cases}$$



2. ReLU:

$$out = \max(0, x)$$



At each step, it is necessary to calculate and display the weight changes and explain how the learning rate, initial weights, and the choice of activation function affect the performance of the ADALINE neuron. **(20 points)**

Input table:

X1	X2	Y
0	0	0
1	0	0
0	1	0
1	1	1

Problem 2

In this exercise, we are toying around with the multilayer perceptron architecture.

we fit the following multilayer perceptron on the MNIST dataset:

```
import torch

class PyTorchMLP(torch.nn.Module):
    def __init__(self, num_features, num_classes):
        super().__init__()

        self.all_layers = torch.nn.Sequential(
            # 1st hidden layer
            torch.nn.Linear(num_features, 50),
            torch.nn.ReLU(),
            # 2nd hidden layer
            torch.nn.Linear(50, 25),
            torch.nn.ReLU(),
            # output layer
            torch.nn.Linear(25, num_classes),
        )

    def forward(self, x):
        x = torch.flatten(x, start_dim=1)
        logits = self.all_layers(x)
        return logits
```

The number of parameters for MNIST in this implementation is calculated as follows:

Input size: num_features (for MNIST, this is 784 because images are 28x28 pixels)

Output size: 50 (first hidden layer)

The number of parameters:

- Weights: 784×50=39,200
- Biases: 50

Total parameters for this layer: 39,200+50=39,250

1st Hidden Layer to 2nd Hidden Layer:

- Input size: 50
- Output size: 25 (second hidden layer)

The number of parameters:

- Weights: 50×25=1,250
- Biases: 25

Total parameters for this layer: 1,250+25=1,275

2nd Hidden Layer to Output Layer:

- Input size: 25
- Output size: num_classes (for MNIST, this is 10 because there are 10 digits)

The number of parameters:

- Weights: 25×10=250
- Biases: 10

Total parameters for this layer: 250+10=260

Total Number of Parameters

Adding all the layers together:

39,250+1,275+260=40,785

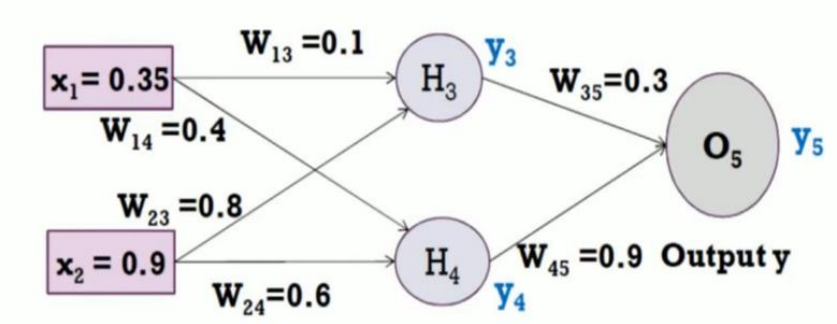
So, this network had 40,785 parameters, and it achieved the following accuracy values:

Train Acc 97.24%
Val Acc 95.64%
Test Acc 96.46%

Can you change the architecture to achieve the same (or better) performance with fewer parameters and only 1 hidden layer? (By providing the reasons and complete details of your model.) **(15 points)**

Problem 3

Suppose that during the training of a neural network, the weights are as follows:



Now the following training data is used to train this network.

X1	X2	y
0.35	0.9	0.5

Where x1 and x2 are the network inputs, and y is the expected output for these inputs.

a. Compute the network's output (y5) for the given inputs.
b. Update the model's weights using the gradient descent algorithm within two epoch.

"Assume the learning rate is set to 1 and the bias of all neurons is set to 1."
(20 point)

Problem 4

In this problem, the fundamentals of Python and PyTorch are introduced first, followed by an implementation of a multi-layer perceptron with PyTorch.

The MNIST dataset, a classic collection of handwritten digits with 60k training examples and 10k test, is used.

a. The following links are to help you review how to work with Python and PyTorch. Python, PyTorch

b. For the implementation, refer to the Notebook **HW1_Q4. ipynb** . Only specific parts need to be completed, which are marked as "TO DO" within the notebook. (Modify only these specified sections.)

The submitted file must include outputs and a detailed explanation of the work you have done; otherwise, no marks will be awarded.
(45 points)