Homework#1: Implementing Fully-connected Neural Network

Due date: 04/02/2025 23:59 PM

Submission Format: Python code (.py file) and report (.pdf file)

Overview

This homework assignment aims to implement a neural network using two popular packages: NumPy(>1.20) and PyTorch(>2.2.1). The implementation should be in Python and the resulting code should be submitted along with a detailed report that explains the implementation.

Task 1 (10 points)

Given a neural network architecture and a pair of inputs with weights, implement neural networks using NumPy and PyTorch. The architecture of the neural networks is as follows:

- Input layer with 3 nodes
- Hidden layer with 4 nodes and ReLU activation function
  For your information, please refer to
  https://en.wikipedia.org/wiki/Rectifier_(neural_networks)
- Output layer with 2 nodes and softmax activation function

Use the following weights for the inputs:

x_1 = [1.0, 2.0, 3.0]

x_2 = [4.0, 5.0, 6.0]

The weights for the neural network are:

w_1 = [[0.1, 0.2, 0.3, 0.4],

[0.5, 0.6, 0.7, 0.8],

[0.9, 1.0, 1.1, 1.2]]

w_2 = [[0.2, 0.1],
[0.4, 0.5],
[0.6, 0.2],
[0.8, 0.7]]

Implement the neural networks using **both NumPy and PyTorch** and print the outputs of the neural networks for the given inputs. We note that we do not use bias terms (i.e., $\boldsymbol{w_0} = [0, \dots, 0]$.)

Note: Please submit your code with detailed report that explains the implementation. The report should be in the python code. The code should be executable.

Task 2 (10 points)

Print out gradients of loss with respect to a specific weight from two neural networks. The neural networks are the ones you implemented in Task 1. The loss function is cross-entropy loss. The target values for the output layer are:

y_1 = [0, 1]
y_2 = [1, 0]

Print the gradients of the loss function with respect to weight w1 for both neural networks, which are from **both NumPy and PyTorch**. There is no learning or update for the task 2.

Note: Please submit your implementation with a README file, the outputs of the gradients for both neural networks, and a detailed report that explains the implementation. The code should be executable. We also note that cross-entropies here should be implemented by your hand.

Task 3 (10 points)

Please repeat the processes from Task 2 and update the $\mathbf{w_1}$ and $\mathbf{w_2}$ **FOR 100 times (i.e., 100 epochs)**. In your report, provide and compare the updated $\mathbf{w_1}$ and $\mathbf{w_2}$ for **both NumPy and PyTorch**.

Use the following learning rate for gradient descent optimization:

```
learning_rate = 0.01
```

Package Requirements

Please use NumPy and PyTorch only for your implementation. The use of other packages is prohibited and implement your code by your hand.

Submission Instructions

Submit your Python source code (.py file) and report (.pdf file) through the iCampus. Your file should be named as `studentID_firstname_lastname_HW#1.py` (e.g., 2020113522_Justin_Bieber_HW#1.py) and `studentID_firstname_lastname_HW#1.pdf`.