Machine Learning

COMP3611

Coursework: Multi-layer Perceptron

Deadline: **13 Dec 2021** by electronic submission via Gradescope.

Implement the following specification in Python and using only NumPy. This coursework is individual and must not be completed in group or by using code snippets from the Internet. The code will be checked for plagiarism.

Introduction

The MNIST database of handwritten digits has a training set of 50,000 examples, validation set of 10,000 examples and a test set of 10,000 examples. The digits have been size-normalized and cantered in a fixed-size image. Each image is 28x28 pixels and labelled into one of 10 categories [0,1,2,3,4,5,6,7,8,9]. We will only use a subset of this dataset for the coursework:

First 10,000 examples of the training set for training and 1,000 examples of the testing set for testing.

Prerequisite

In this Coursework you will design and train a two-layer perceptron to perform image classifications on the MNIST dataset.

In order to run the code necessary for this coursework, you need to install the following packages inside an anaconda environment or you can work on the Google Colab.

```
pip install numpy
```

```
pip install jupyter
```

```
pip install matplotlib
```

This setting is the exact same setting we used for the weekly labs.

Design of a two-layer NN with NumPy

You are going to use only NumPy library to design a two-layer perceptron. The network will have two hidden layers and an output layer. Each hidden layer will be followed by a sigmoid function as its activation function. The output layer will have a softmax function as its activation function since we are working on a multi-class classification task.

Exercises

You will implement exercise 1 and 2 in the provided **'mlp.py'** file and test your implementation in the provided Jupyter Notebook.

Exercise 1

Start by opening the given **'mlp.py'** file and implement the forward phase of the network training procedure. This is similar to the lab implementation where we went through implementing the MLP algorithm with one hidden layer and linear output activation function. However, here, you are asked to do it with two hidden layers and the output layer activation function is softmax. Please implement this inside the 'forwardPass' method of the MLP class in the given **'mlp.py'** Python file

Exercise 2

Implement the backward phase of the network training procedure. Please implement this inside the for loop in the 'train' method of the MLP class in the given **'mlp.py'** Python file. Test your implementation in the notebook. You can train the network with the default parameters setting in the notebook. Once the network is trained for 1000 iterations, evaluate the network by testing its performance on the testing dataset. The testing accuracy will likely be less than %50.

Exercise 3

We can improve the performance of the network by experimenting with different parameter setting. Find the parameter setting that achieves more than 90% testing accuracy. You can change the number of iterations, the learning rate value, and the number of neurons in each hidden layer. You can also experiment with training the network with gradient descent with momentum. The goal is to get the testing accuracy to be more than %90. Once you have found the best parameter setting that achieved the desired accuracy, please run the cell in the notebook that saves the parameters and the network weights in a pickle file.

This exercise gives you a chance to experiment with architecture and parameter optimization, a fundamental step in real-world use of neural networks.

Submission Instructions

You hand in two files and nothing else:

1. The final Python file '**mlp.py**'
2. The pickle file '**best_classifier.pkl**'; that has the weights and the parameter setting of the best network.

Please submit these files separately and don't put them in a folder (this is how Autograder in Gradescope will need to read your submitted files).

Marking scheme (total available: 100)

- Verification of how the forward phase (10/100) and backward phase (20/100) have been implemented. **(30/100)**

- Code runs and can train the network with the default parameters setting in the notebook, where the testing accuracy will be less than %50. **(50/100)**

- Finding the best parameters setting to get testing accuracy to be more than %90. **(20/100)**

    - If you manage to improve the testing accuracy more than %50 but less than %90; we still will give you 5 points out of 20.

- No score for submitted files with different name and different format.

- No score for similar codes.