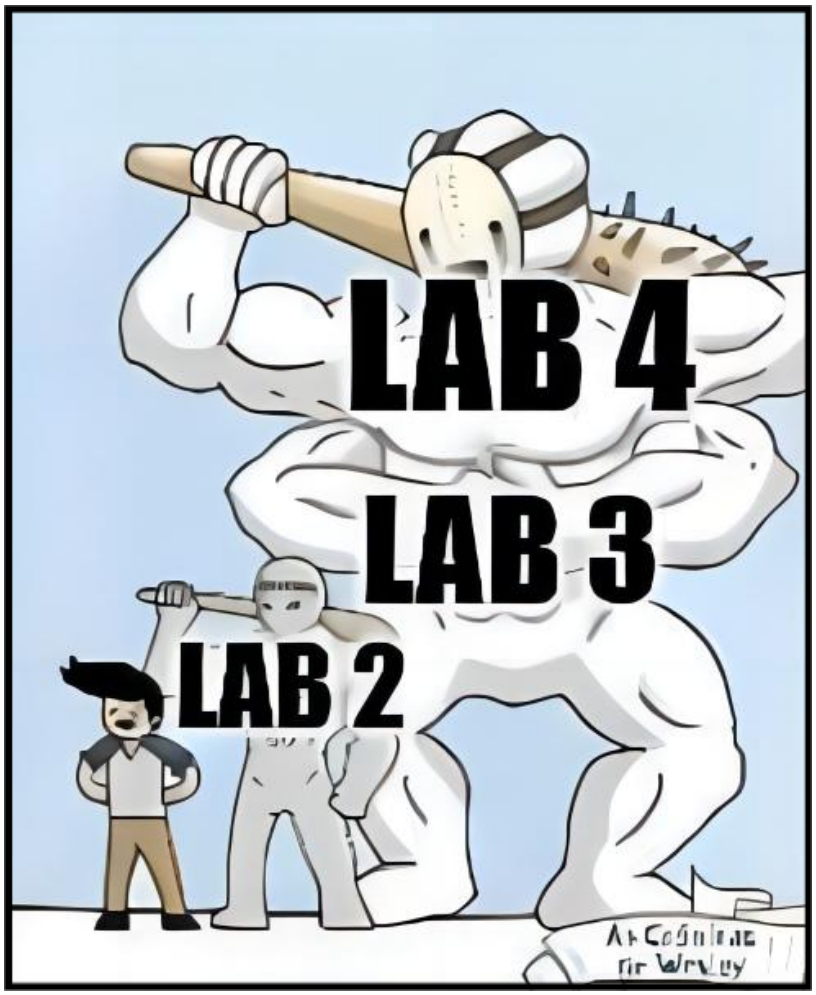
ECE4179 - Lab 3
Multi-Layer Perceptrons (MLP)



Contents

Academic integrity and the Use of Generative AI 3
Late submission policy and grading guidelines 4
Task 1: Approximate the Sine Function 7
Task 2: MLP for classification of the Covertype dataset 8
Task 3: Deep MLP for classification of the Covertype dataset 9

Academic integrity and the Use of Generative AI

Academic integrity

Every lab submission will be screened for any collusion and/or plagiarism. Breaches of academic integrity will be investigated thoroughly and may result in a zero for the assessment and interviews with the plagiarism officers at Monash University.

Lab Instructions and the Use of Generative AI

- You should use Matplotlib to display images and any intermediate results.
- You may use a generative AI tool or coding assistance. We recommend understanding the concepts and coding as much as possible, as you may be performing hand calculations and write code by yourself during the final exam. If you use generative AI tools, please indicate the prompts you used and explain in detail any changes you made to modify the code to complete the assignment.

Late submission policy and grading guidelines

Late submissions

Late lab submission will incur a penalty of 5% of available marks for each day late. That is, with one day delay, we will (sadly) deduct 5% of this lab from the mark you get, *i.e.* if you get 75% and submitted one day late, the final marks you will recieve is 70%. Labs submitted with more than a week delay will not be assessed. Please apply for special consideration for late submission as soon as possible (*e.g.*, documented serious illness). You can do this by applying through Monash University's special consideration application website.

Lab Grading

This lab is worth 5/100 (5 out of 100) marks of the entire unit. Lab attendance is not compulsory. The lab sessions are there for you to get help.

For the .ipynb submission, there are a number of tasks. Each task will have coding components and a discussion component. A task is only considered complete if you can demonstrate a working program and show an understanding of the underlying concepts. Note that later tasks should reuse code from earlier tasks.

You can complete this lab with a partner. You need to register on Moodle under week 5. Only one of you need to submit the lab.

There are 3 tasks in this lab. The tasks are worth 40%, 30% and 30% respectively. Both the coding and discussion tasks are assessed, so please complete all of it! You can enter the discussion answers directly into the notebook.

This lab is about understanding and applying PyTorch-Lightning to simple multi-layer perceptron tasks. By the end of the lab, you will have developed simple MLPs for multi class classification tasks. The following are the four tasks that you should complete.

- Task 1: Approximate the sine function using a MLP by following through the framework for implementing a deep learning model - from creating custom dataset to designing the shallow MLP model, and lastly to train and evaluate model performance.
- Task 2: Applying MLP for classification task. The model used here will remain shallow, and you will be following through the same framework of applying the MLP model to solve the multi-class classification problem.
- Task 3: Improving model performance from task 2's MLP model by adding additional hidden layers.

The learning outcomes for this lab are:

- Familiarising yourself with PyTorch and PyTorch-Lightning
- Understanding implementation of MLPs and network training.
- Analysing and describing accuracy and loss plots.
- Understanding the implications of different MLP results.

Note: Most of the lab does not require you to use the Numpy library. You will be using Pytorch/Pytorch-Lightning in-built methods to create your tensors instead of Numpy. Follow the Pytorch/Pytorch-Lightning videos for more information.

Introduction.

The fundamentals of deep learning models start off with understanding multilayer perceptrons and the training processes required. There are additional hyper-parameters that we can choose and tune, such as the type of optimizers, learning rate, activation functions, and model architecture to name a few. The ability of activation functions to model non-linearities is what allows MLPs to be able to generalise better than traditional machine learning models.

The submission deadline is **30th of August (Friday) 4:30 PM AEST**.

It is recommended that you go through the following videos/documents prior to attending your lab 3:

1. Begin by reading through this document. This document contains all the relevant information for lab 3
2. Follow the lessons on PyTorch/PyTorch-Lightning
3. Watch the introductory video for this lab
4. You may choose to use GoogleColab for this lab so that you can utilise the free GPU provided. [1]

In the lab and in your own time, you will be completing the lab 3 notebook by going through this document and the provided notebooks.

[1] You can choose not to use GoogleColab if you have your own GPU or if you prefer to use your own CPU. It is not as necessary for this lab but it will be useful for the next lab and assignment.

Task 1: Approximate the Sine Function [40%]

This section include show to develop and evaluate a small neural network for function approximation. You will approximate the sine function with a MLP. The following tasks need to be completed:

- 1.1 Data: Create custom dataset and dataloaders
- 1.2 Model: Design a Shallow MLP model
- 1.3 Train & Evaluate: Train and evaluate model's performance
- 1.4 Visualise and Analyse the Experimental Results

The first task in this section aims to setup a dataloader class for the noisy sine data. Task 1.2 requires you to design a shallow linear MLP model with the following structure/hyper-parameters:

Parameter	Value	Description
λ	1e-2 (SGD), 1e-3 (ADAM)	Learning rate
# Hidden layers	1	The layers you will be coding
# Neurons	{64}	Number of neurons in each layer
Activation function	tanh	For each hidden layer
Loss function	MSE	*Use "nn" module*
Optimizer	SGD, ADAM	*Use "torch" module*

Table 1: Hyper parameters for section 1

After constructing the MLP model, you can train the model using the *Trainer* constructor as it has useful built-in methods such as logging results, model checkpointing, training and validation loop, early-stopping. [2]

Ensure you answer the discussion questions at the end of the task as well.

[2]Make sure to check out the lecture videos to understand these concepts

Task 2: MLP for classification of the Covertype dataset [30%]

In this task, you will use your knowledge learned from Task 1 to build another shallow MLP to perform a classification task on a forest cover type dataset. This dataset classifies the cover type into 7 classes by considering 54 different features. More descriptions can be found in the notebook. The tasks need to be completed are:

- 2.1 Data: Load the CoverType dataset from sci-kit learn
- 2.2 Model: Design the Shallow MLP model
- 2.3 Train & Evaluate: Train and evaluate model's performance in different secarios
- 2.4 Visualise the Results
- 2.5 Check the performance in the test dataset

For this Task 2.1, the dataset should be loaded from the existing database that is managed by sci-kit learn. Most instructions are given in the notebook, and please make sure you understand the code. Then in Task 2.2, you will design a shallow MLP which is similar to but slightly different from the one in Task 1. The general structure of the network is shown below:

$$\mathbb{R}^{54} \ni \boldsymbol{x} \rightarrow \text{fc1} : \text{Linear}(54 \times n_{hiddennodes}) \rightarrow \text{ReLU}$$
$$\rightarrow \text{fc2} : \text{Linear}(n_{hiddennodes} \times n_{classess}) = \hat{\boldsymbol{y}} \text{ (predicted class)}$$

After the training is done, you will again do some analysis to the results. And answer the discussion questions in the end.

Task 3: Deep MLP for classification of the Covertype dataset [30%]

Now in this task, you will implement a Deep MLP which has more hidden layers to again do classification on the CoverType dataset. The tasks should be completed are:

- 3.1 Model: Design a Deep MLP model
- 3.2 Train & Evaluate: Train and evaluate model's performance
- 3.3 Visualise the Results

Since the data is already loaded, you can start from design the model, the structure of the model should be similar to the one below:

$$\mathbb{R}^{54} \quad \ni \quad \boldsymbol{x} \quad \rightarrow \text{fc1} : \text{Linear}(54 \times n_1) \rightarrow \text{ReLU} \rightarrow$$
$$\text{... More Hidden Layers ...}$$
$$\rightarrow \text{fc3} : \text{Linear}(n_x \times 10) = \hat{\boldsymbol{y}} \text{ (predicted class)}$$

The training process would be similar to previous tasks, and you will be asked to complete some analysis task.

And again, do not forget to complete the discussion questions at the end of task.

Hopefully this lab has given you insight in training a simple model via the PyTorch-Lightning framework, and provided realistic scenarios in which optimizing hyper-parameters and considering data augmentation is required for any deep learning problem!