# COMP3211:
# Fundamentals of Artificial Intelligence

**Homework Assignment 1**


Release Date: Sep. 30, 2023

# The HKUST Academic Honor Code

Honesty and integrity are central to the academic work of HKUST. Students of the University must observe and uphold the highest standards of academic integrity and honesty in all the work they do throughout their program of study. As members of the University community, students have the responsibility to help maintain the academic reputation of HKUST in its academic endeavors.

# 1 Introduction

In this homework, we will have three questions. First, we will draw a computation graph and work out the gradients. Then, we will implement MLPs on the 2 moon and Iris datasets to train and evaluate. The submission deadline for this homework is **11:59 pm, Oct. 18, 2023**. Please start as early as possible, and feel free to discuss with us if you have any questions about your design and implementation.

This assignment should be solved individually. No collaboration, sharing of solutions, or exchange of models is allowed. Please, do not directly copy existing code from anywhere other than your previous solutions, or the previous master solution. We will check assignments for duplicates. See below for more details about the homework.

# 2 Preliminaries

## 2.1 Environment Setup

Students are required to do the following programming assignments with the anaconda environment, as introduced in tutorial1.ipynb on canvas. Before running the code, dependencies are required to install. You should run

```
pip3 install matplotlib
```

or

```
conda install matplotlib
```

The code is written by the jupyter notebook, you can open the assignment using

```
jupyter notebook assignment1.ipynb
```

## 2.2 Import Packages

We run the first code block to import the packages.

```
import torch
import numpy as np
import torch.nn as nn
import matplotlib.pyplot as plt
import csv
import random
```

NOTE: the anaconda environment should contain the above packages once you follow tutorial1.pptx. If you encounter reported error like "matplotlib not found", just refer to Sec. 2.1 and install the "matplotlib" package.

### 2.3 Support Functions

We provide several support functions that are necessary to run the code. These functions are important to finish Question 2 and Question 3 successfully. Please **DO NOT** modify any of them.

Feel free to read them in detail if you are interested in them. These respective functions are explained as follows:

```
def setup_seed(seed):
    # fix the random seed of the numpy and pytorch engine,
    # make sure the final results are reproducable;
    ...

def AccuarcyCompute(pred: torch.Tensor, label: torch.Tensor):
    # compute the accuracy according to
    # predictions of model and the groundtruth label from dataset;
    ...

def plot_fig(Y: list, title: str, dir: str, X=None, x_label=None):
    # plot the training loss and validation accuracy;
    ...

def load_data(dataset: str):
    # load the training data for training and
    # the validatation for your plotting figure;
    ...
```

### 2.4 Execute the Code Block

You can execute each code block by pressing "shift+enter". If you are not familiar with python or jupyter notebook, you can just execute each code block in order.

## 3 Question 1: Computational Graph

Draw the computation graph of the following function:

$$f(x_0, x_1, x_2, x_3) = e^{x_0^4 - x_1^{-3} + x_2^2 - x_3^{-1} + 1},$$

with $x_0 = 1, x_1 = -1, x_2 = 2, x_3 = -1$. as in the lecture notes (*4-compgraph.pdf*), obtain all the intermediate outputs, final $f$, and gradients with respect to $x_0, x_1, x_2, x_3$. (note: show each output above the edge and the gradient below the edge.) (20 points)

# 4 Question 2: 2-Moon

In this part, we will design and train MLPs to perform classification tasks on the 2-moon dataset, provided in the assignment1.zip. The 2-moon dataset is a 2-dimensional dataset with two labels: 0 and 1. It has 2 input features. The third column in the CSV files contains the labels for the data.

Table 1: Sample of the 2-moon dataset

| Feature 1 | Feature 2 | Label |
|---|---|---|
| -0.083 | 1.126 | 0 |
| -1.126 | 0.348 | 0 |
| ... | ... | ... |

Given the 2-moon dataset, build an MLP with a single hidden layer of 2 hidden units. Fill in the build_1_layer_mlp() function in assignment1.ipynb. The function should take in a parameter nbr_hidden_unit and should build a Multi-layer perceptron (MLP) that has a single hidden layer with nbr_hidden_unit hidden units. Use ReLu as the activation function of MLP. Lastly, return the built MLP.

Next, train the MLP using the train_mlp() function provided. The code also plots the validation accuracy with the number of hidden units as the x-label, and visualize the decision boundary obtained. Repeat the above procedure with 2, 10, 50, 100, 1000, and 10000 hidden units, and describe what you observe when the number of hidden units is varied. You are also provided with part of the code in assignment1.ipynb. (40 points)

## 4.1 Introduction to the provided codes

### 4.1.1 Load the Data

The code block of the following is to load the data for training and validation:

```
setup_seed(3211)

data_list, label_list = load_data('make_moon')
train_data = data_list[0]
train_label = label_list[0]
valid_data = data_list[1]
valid_label = label_list[1]
test_data = data_list[2]
test_label = label_list[2]
...
```

**IMPORTANT:** Please **DO NOT** modify the number 3211 passed to the setup_seed(). Otherwise, we may not be able to reproduce your results.

Once you execute the code block and load the data, you can check the shape of the data by executing the following in a blank block:

```
train_input.shape, train_label.shape
```

If there is no problem in the data loading process, the result should be like:

```
(torch.Size([800, 2]), torch.Size([800]))
```

This means each input has two dimensions, with 800 inputs in total.

### 4.1.2 Your Implementation

You are required to implement a one-layer MLP of dimension nbr_hidden_unit in this code block. You can refer to tutorial3.pptx for examples.

```
def build_1_layer_mlp(nbr_hidden_unit):
    # Your code here
    return mlp
```

This is the **most important** part of Question 2. If you fail to implement it, you will lose all marks of Question 2.

### 4.1.3 Training

The code block of the training process is like:

```
def train_mlp():
    epochs = 30
    loss_list, test_acc = [], []
    lossfunc = torch.nn.CrossEntropyLoss()
    hidden_unit_list = [2, 10, 50, 100, 1000, 10000]
    for hidden_unit in hidden_unit_list:
        mlp = build_1_layer_mlp(hidden_unit)
        ...

train_mlp()
```

It trains the one-layer MLP of different dimensions for the hidden layer, *i.e.*, 2, 10, 50, 100, 1000, 10000. You execute the code block once you have correctly implemented the one-layer MLP in Sec. 4.1.2. You will get the validation accuracies for the different settings, and figures of the decision boundaries. Note that these boundaries are not the exact boundaries that could divide the label.

You will also get the validation accuracy of different dimensions of hidden layers:

# 5 Question 3: Iris

In this question, you will design and train MLPs to perform classification on the Iris dataset, provided in the assignment1.zip. The Iris dataset contains three labels (0, 1, and 2) with 4 input features. The fifth column in the .csv files contains the labels for the data.

Table 2: Sample of the Iris dataset

| Feature 1 | Feature 2 | Feature 3 | Feature 4 | Label |
|---|---|---|---|---|
| 5.0 | 3.3 | 1.4 | 0.2 | 0 |
| 6.7 | 3.3 | 5.7 | 2.5 | 2 |
| ... | ... | ... | ... | ... |

From Q2, you learn the effect of the number of layers on MLP performance. In this question, given the Iris dataset in the assignment1.zip, try to construct MLPs with 2 hidden layers by filling in the build_mlp() function. The first hidden layer has 50 units, and the second hidden layer has 100 units. Train the MLP with the train_mlp() function and use the plot_fig() function to plot your model's training loss curve and validation accuracy curve with the number of training epochs as the x-axis. Print your testing accuracy using the print() function and plot your model's decision boundary using the plot_decision_boundary() function provided. The training, validation, and testing sets are provided, and the code loading the data was given in the assignment1.ipynb. (40 points)

## 5.1 Introduction to the provided codes

### 5.1.1 Load the Data

The code block of the following is to load the data for training and validation:

```
setup_seed(3211)

data_list, label_list = load_data('iris')
train_data = data_list[0]
train_label = label_list[0]
valid_data = data_list[1]
valid_label = label_list[1]
test_data = data_list[2]
test_label = label_list[2]
...
```

**IMPORTANT:** Please **DO NOT** modify the number 3211 passed to the setup_seed(). Otherwise, we may not be able to reproduce your results.

### 5.1.2 Your Implementation

You are required to implement a two-layer MLP in this code block. The first hidden layer has 50 units, while the second hidden layer has 100 units. You can refer to tutorial3.pptx for examples.

```
def build_mlp():
    # Your code here
    return mlp
```

This is the **most important** part of the Question 3. If you fail to implement it, you will lose all marks for Question 3. Please be very careful about the output dimension of the two-layer MLP. The output dimension should equal the number of classes of the Iris dataset.

### 5.1.3 Training

The code block of the training process is like:

```
def train_mlp():
    # Your code here
train_mlp()
```

You need to fill in the train_mlp() yourself this time. You may refer to any code in Question 2. Plotting the training loss, validation accuracy, and the decision boundary of your model. If your model is good enough, you will have a testing accuracy of over 0.90.

## 6 Submission

Please submit a PDF file via canvas, including the answer for question 1 and a completed Python notebook file for questions 2 and 3 (based on the assignment1.ipynb file) to show your work. Name the pdf and .ipynb file in the format **YourStudentID_assignment1.ipynb** (e.g., 12345678_assignment1.ipynb) and upload them to Canvas, note that any code error would lead to 0 points for the respective question. You may submit two times, one for .pdf file, and another for .ipynb file. Required results should be shown clearly.