BME646 and ECE60146: Homework 1

Spring 2024
Due Date: 11:59pm, Jan 15, 2023
TA: Akshita Kamsali (akamsali@purdue.edu)

Turn in typed solutions via BrightSpace. Additional instructions can be found on BrightSpace announcement. The finalized policies regarding the programming assignments can be found on BrightSpace after the first week of classes. Kindly hold off your policy related questions until then.

1 Introduction

The goal of this homework is to improve your understanding of the Python Object-Oriented (OO) code in general, especially with regard to how it is used in PyTorch. This is the only homework you will get on general Python OO programming. Future homework assignments will be specific to using PyTorch classes directly or your own extensions of those classes for creating your DL solutions.

Note that you should use Python 3.x and NOT Python 2.x for this and all future programming assignments. For the Python-related knowledge required in this homework, refer to Prof. Kak's tutorial on OO Python [1].

2 Programming Tasks

1. Create a class named `Sequence` with an instance variable named `array` as shown below:

```
class Sequence(object):
    def __init__(self, array):
        self.array = array
```

   The input parameter `array` is expected to be a list of numbers, *e.g.* `[0, 1, 2]`. This class will serve as the base class for the subclasses later in this assignment.

2. Now, extend your `Sequence` class into a subclass called `Arithmetic`, with its `__init__` method taking in two input parameters: `start` and `step`. These two values will serve as the start and step of the Arithmetic sequence.

3. Further expand your `Arithmetic` class to make its instances *callable*. More specifically, after calling an instance of the `Arithmetic` class with an input parameter `length`, the instance variable `array` should store a Arithmetic sequence of that length and with `start` as your initial value and increments of `step`. In addition, calling the instance should cause the computed sequence to be printed. Shown below is a demonstration of the expected behaviour described so far:

```
AS = Arithmetic(start=1, step=2)
AS(length=5)   # [1, 3, 5, 7, 9]
```

4. Modify your class definitions so that your `Sequence` instance can be used as an *iterator*. For example, when iterating through an instance of `Arithmetic`, the numbers should be returned one-by-one. The snippet below illustrates the expected behavior:

```
AS = Arithmetic(start=1, step=2)
AS(length=5)   # [1, 3, 5, 7, 9]
print(len(AS))  # 5
print([n for n in AS])  # [1, 2, 3, 5, 8]
```

5. Make another subclass of the `Sequence` class named `Geometric`. As the name suggests, the new class is identical to `Arithmetic` except that the array now stores a series. Modify the class definition so that its instance is *callable* and can be used as an *iterator*. What is shown below illustrates the expected behavior:

```
GS = Geometric(start=1, ratio=2)
GS(length=8)   # [1, 2, 4, 8, 16, 32, 64, 128]
print(len(GS))  # 8
print([n for n in GS])  # [1, 2, 4, 8, 16, 32, 64, 128]
```

6. Finally, modify the base class `Sequence` such that two sequence instances of the same length can be compared by the operator `==`. Invoking `(A == B)` should compare element-wise the two arrays and return the number of elements in A that are equal than the corresponding elements in B. If the two arrays are not of the same size, your code should throw a `ValueError` exception. Shown below is an example:

```
AS = Arithmetic(start=1, step=2)
AS(length=5)   # [1, 3, 5, 7, 9]
GS = Geometric(start=1, ratio=2)
```

```
GS(length=5)   # [1, 2, 4, 8, 16]
print(FS == GS)  # 1

GS(length=8)   # [1, 2, 4, 8, 16, 32, 64, 128]

print(FS == GS)  # will raise an error
# Traceback (most recent call last):
#   ...
#   ValueError: Two arrays are not equal in length!
```

3 Submission Instructions

Include a typed report explaining how did you solve the given programming tasks. You may refer to previous homeworks for an outline.

1. Turn in a zipped file, it should include (a) a typed self-contained pdf report with source code and results and (b) source code files (only .py files are accepted). Rename your .zip file as hw1_<First Name><Last Name>.zip and follow the same file naming convention for your pdf report too. Not adhering to the above naming convention will lead to an automatic zero.

2. For this homework, you are encouraged to use `.ipynb` for development and the report. If you use `.ipynb`, please convert it to `.py` and submit that as source code. Do NOT submit .ipynb notebooks.

3. You can resubmit a homework assignment as many times as you want up to the deadline. Each submission will overwrite any previous submission. **If you are submitting late, do it only once on BrightSpace.** Otherwise, we cannot guarantee that your latest submission will be pulled for grading and will not accept related regrade requests.

4. The sample solutions from previous years are for reference only. **Your code and final report must be your own work.**

5. Your pdf must include a description of

   - Reproductions of the outputs for each of the provided snippets above with the given parameters.
   - Correct outputs for each of the provided snippets above with input parameters of your choice.

- Your source code. Make sure that your source code files are adequately commented and cleaned up.

References

[1] Python OO for DL. URL https://engineering.purdue.edu/DeepLearn/pdf-kak/PythonOO.pdf.