Assignment 01

Image and Video Processing with Deep Learning

PUBLISHED: 22 JAN 2025

Submission deadline:
Sunday 26 JAN 2025 16:59:59 (5 PM)

Instructions

The following instructions should be followed strictly. Your assignment is going to be checked and graded using a python script which depends on these naming and format conventions to be able to run your program successfully. So any deviation from the following format conventions will result in 0 points.

File Name format:

asgn1_student_full_name.py
where student_full_name should be the name as it appears in IISER records with spaces replaced by underscore. Note that the entire filename is all small letters (including asgn1 and name)

For example, if my name appears as
“Chaitanya Guttikar” in student records, then the filename should read **asgn1_chaitanya_guttikar.py**

Program format:

Your program should have the following class implemented. Make sure all the methods mentioned below are implemented with exact same names (including capital/small case) and their input and output types are as mentioned here.

We are building a learning pipeline using numpy to
fit the model
y = wx

to the data
[(x_1, y_1) , … , (x_n, y_n)]

(In class, we were writing y = ax instead of y = wx)

```python
import numpy as np

class LinReg:
    def __init__(self, data):

        """
          Data is expected to be in the form
          [(x_1,y_1), (x_2, y_2), ..., (x_n, y_n)]

        """
        <Your code here>

    def get_weights(self) -> np.ndarray:
        """
           This function should return the weights as a numpy array. I will
           call this function to check how your weights/parameters are
           getting updated. In our current case, this will be a 1 dimensional
           numpy array with only one entry. This is to make our code future
           proof.

        """
        <Your code here>

    # model output
    def forward(self, x):
        <Your code here>

    # loss = MSE
    @staticmethod
    def loss(y, y_pred) -> float:
        <Your code here>

    # Gradient of loss with respect to weights.
    @staticmethod
    def gradient(x, y, y_pred) -> float:
        <Your code here>


    def fit(self, learning_rate: float = 0.01, n_iters: int = 20) -> None:
        """
          Feel free to change the default number of iterations and the default
          learning rate to your liking.

        """
        <Your code here>
```

DO NOT USE any packages other than numpy and the built in python packages.

What my program will do:

1. It will have some code that generates *my_data* which will be in the form mentioned above ([(x_1, y_1), …, (x_n, y_n)]. Since this is synthetically generated data, I will already know the best ‘w’ for the model y = wx. Let’s call it *w_best*
2. It will import the LinReg class from your file and instantiate a model *model = LinReg(my_data)*
3. It will run *model.fit()* with your default *learning_rate* and *n_iters* as well as some other values.
4. It will periodically run model.get_weights() to see how the weights are changing.
5. It will check the final value of weights with the best w that I am expecting by doing *weights = model.get_weights(),* and comparing *w_best* with *weights[0]* - Note that *get_weights* method returns a numpy array of weights.
6. It will also run model.forward, model.loss and model.gradient to check if they have been implemented correctly.

If there are any questions, please post them on the google classroom.