DATA7703, Practical 9

2022 Semester 2

1. (Training neural networks for OLS)

(a) Use the `regression_data` function in the lecture slides to generate a dataset of 200 examples $(\mathbf{x}_1 y_1), \ldots, (\mathbf{x}_{200}, y_{200}) \in \mathbf{R}^2 \times \mathbf{R}$. Before you generate the data, first use `torch.manual_seed(1)` to set the random seed for reproducibility.

Use `matplotlib` to plot the dataset. You may find this example of generating 3D plots helpful: https://matplotlib.org/3.1.1/gallery/mplot3d/scatter3d.html. If you put your code in a file, say `P8.py`, and run `python P8.py` to show the plot, then you can rotate the plot to visualize it from any angle you want.

(b) Use `sklearn.linear_model.LinearRegression` to fit an OLS model on the dataset. Print out the model parameters.

(c) Use the three approaches described in lecture slides to train an OLS model on the dataset that you constructed in (a). Compare the models with the model that you obtain in (b).

(d) Try 200 iterations of gradient descent with a learning rate of 0.01, 0.1, 1.0 respectively. Comment on the models obtained.

2. (Convolutional filters) In lecture, we discussed about convolutional filter. We will implement a simple convolutional filter in this question, and apply it to process an image.

(a) Implement a function that takes in a 3 dimensional array, and a 2 dimensional filter matrix, and returns the output of applying the filter to the input array.

(b) Download the file `lenna.png` and run the following code.

```
import matplotlib.pyplot as plt
lenna = plt.imread('lenna.png')
print('Images size:', lenna.shape)
plt.imshow(lenna)
plt.show()
```

(c) Apply the convolutional filter with weight matrix $\begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix}$ to `lenna`, and display the output.

(d) A RGB image, like `lenna` can be represented as an $H \times W \times 3$ array. Thus the image can be seen as 3 $H \times W$ arrays or channels stacked on top of each other, or $H \times W$ RGB values. Convert the output in (c) to an 2D array by converting a pixel to 1 if its RGB values are all larger than the average RGB values, and 0 otherwise. Display the 2D array as a grayscale image by adding the `cmap="gray"` option to `plt.imshow`. Describe what the convolutional filter in (c) does.