Exercise 3 - **Assignment 1**

Deep Learning Lab
Due: Sunday 23 October 2022, 10:00 pm (time in Lugano)

October 3, 2022

Submission Instruction

You should deliver the following by the deadline:

- **Report**: a single *pdf* file that clearly and concisely provides evidence that you have accomplished each of the tasks listed below. The report should not contain source code (not even snippets). Instead, if absolutely necessary, briefly mention which functions were used to accomplish a task. All figures should have a caption, and there should be a text that refers to it (see the Latex documentation if you are not familiar with this). Please also make sure that the section numbering in your report exactly follows that of the assignment.

- **Source code**: a single Python script that can be used to accomplish each of the tasks listed above. The source code will be read superficially and checked for plagiarism. Importantly, if a task is accomplished in the code, but not documented in the report, it will be considered missing. Therefore, please make sure to report everything in the report. Note: Jupyter notebook files are not accepted.

Please carefully read the instructions above to prepare your submission. Failure to stick to these rules may result in reduction of points. As stressed in class, it is strictly forbidden to exchange your code with others or copy texts/code from the internet. The score of 0 will be given to all students involved in such cases.

Note for those who have started working on the **'preview' version**: we introduced a slight change. We added the 0-th order coefficient to $\mathbf{w} = [\mathbf{0}, -5, 2, 1, 0.05]$ and modified the definition of $\mathbf{x}$ accordingly. The solution based on the 'preview version' will also be accepted without any penalty, but if you have not started solving the problem yet, please use this updated version.

1 Polynominal Regression [96/100 points]

Let $z \in \mathbb{R}$. Consider the polynomial p given by

$$p(z) = 0.05z^4 + z^3 + 2z^2 - 5z = \sum_{i=0}^{4} \mathbf{w}_i z^i,$$

where $\mathbf{w} = [w_0, w_1, w_2, w_3, w_4]^\intercal = [0, -5, 2, 1, 0.05]^\intercal \in \mathbb{R}^5$. This polynomial can also be expressed as a *dot-product* between two vectors: $p(z) = \mathbf{w}^\intercal \mathbf{x}$ by defining $\mathbf{x} = [1, z, z^2, z^3, z^4]^\intercal \in \mathbb{R}^5$.

Consider also an i.i.d. dataset $\mathcal{D} = \{(z_i, y_i)\}_{i=1}^{N}$, where $y_i = p(z_i) + \epsilon_i$, and each ϵ_i is drawn from a normal distribution with mean zero and standard deviation $\sigma = 0.5$.

Now if we assume that the vector $\mathbf{w}$ is unknown, **linear regression** (see slides Sec. 2.1) could estimate it given the dataset $\mathcal{D}$ by using the dot-product form above and by representing the original dataset $\mathcal{D}$ as $\mathcal{D}' = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$, where $\mathbf{x}_i = [1, z_i, z_i^2, z_i^3, z_i^4]^\intercal \in \mathbb{R}^5$.

1. (4 pts) **Use** the following helper code below to visualize the polynomial above (you have to report a figure). *Do not forget the 0-th degree coefficient in the argument 'coeffs.'*

Listing 1: Helper function to visualize a polynomial.

```
import numpy as np
import matplotlib.pyplot as plt

def plot_polynomial(coeffs, z_range, color='b'):
    # coeffs: tuple or list containing coefficients of
    #   a polynomial in the increasing order of degrees
    # z_range: tuple or list of two elements
    #   representing the range of z (min, max)
    z = np.linspace(z_range[0], z_range[1], 100)
    y = np.polynomial.polynomial.polyval(z, coeffs)
    plt.plot(z, y, color)
```

2. (10 pts) **Complete** the code below that allows you to generate the data $\mathcal{D}'$ described above.

Listing 2: Polynomial regression dataset generation (incomplete).

```
def create_dataset(w, z_range, sample_size, sigma, seed=42):
    """Incomplete documentation."""
    random_state = np.random.RandomState(seed)
    z = random_state.uniform(z_range[0], z_range[1], (sample_size))
    x = np.zeros((sample_size, w.shape[0]))
    for i in range(sample_size):
        # for j in range ... TODO complete
        #     x[i, j] = ...  TODO complete
    y = x.dot(w)
    if sigma > 0:
        y += random_state.normal(0.0, sigma, sample_size)
    return x, y
```

3. (10 pts) Use the completed code and the following parameters to generate training and validation data points:
    - Each x_i should be in the interval $[-3, 3]$.
    - $\mathbf{w} = [0, -5, 2, 1, 0.05]^\intercal$.
    - Use $\sigma = 0.5$.
    - Use a sample of size 500 created with a seed of 0 for training.
    - Use sample of size 500 created with a seed of 1 for validation.
4. (6 pts) Visualize the generated training data points (i.e., the '(z, y)' pairs) in a 2D scatterplot. Do the same for the validation set (report two separate figures).
5. (4 pts) Search for the documentation of `torch.nn.Linear`. Notice the flag `bias`. Explain what that flag does using equations. Should that flag set to `True` or `False` for this problem? Explain.
6. (20 pts) Adapt the code of linear regression in the 1D case presented in the lecture (at the end of Sec. 2.1) to perform polynomial regression using the generated training dataset $\mathcal{D}'$.
7. (20 pts) Find and report a suitable **learning rate** and **number of iterations** for gradient descent. Report the **initial (random) values** and the **estimate** of $\mathbf{w}$ you obtained after training. You can consider your hyper-parameters to be good if your training and validation losses are below 0.6. *Hint: when your learning rate is too high, you may get 'nan'. If it is too low, you may need more than 2000 steps to achieve* 0.6.

8. (10 pts) Plot the training and validation losses as a function of the gradient descent iterations.
9. (2 pts) Visualize the polynomial defined by the estimate of $\mathbf{w}$ you obtained above, and comment.
10. (10 pts) Report and explain what happens when the training dataset is reduced to 10 observations while keeping the number of validation data points **unchanged**. (you can use the hyper-parameters you found above; no need to tune them again).
11. (**Bonus**, 10 pts) Plot the evolution of each coefficient of $\mathbf{w}$ as a function of the gradient descent iterations.

2 Questions [4/100 points]

Provide a **brief and concise** answer (**max. 3 sentences**) to the following questions **in your own words** and add them to your report. You can find more information in the relevant subsections of chapter 5,8 of the Deep Learning Book for example.

1. (2 pts) What does it mean if your model is overfitting?
2. (2 pts) How can you mitigate overfitting? (one example solution is enough)

NB: We are expecting concise answers. Long answers including reformulations ("in other words...") or long examples will not be graded.