Assignment 1

Deadline: 10.22.2023 - 11.59pm

By the deadline, you are required to submit the following:

Report. Prepare a single PDF file that presents clear and concise evidence of your completion of each of the listed tasks. **Use the overleaf template available on iCorsi**.

Source code. Create a single zipped Python script **using the template available on iCorsi**, performing each of the specified tasks. If a task is accomplished in the code but not documented in the report, it will be considered incomplete. Therefore, make sure to thoroughly report all your work in the submitted report. **Jupyter notebook files will not be accepted**.

General hints. The question marked with * is more challenging, and we recommend possibly leaving them as the last questions to solve. To obtain the maximum grade, not only should all exercises be completed correctly, but the plots must also be clear, have a legend, and have labels on the axes and the text should be written in clear English. For saving plots in high quality, consider using the command `matplotlib.pyplot.savefig`. Clarity of plots and text will account in total for 5 points.

Submission rules: As already discussed in class, we will stick to the following rules.

- Submit ONE pdf file containing your report and ONE (zipped) .py file containing the code. Use the templates and name your files NAME_SURNAME.pdf and NAME_SURNAME.py. We will read and correct only these files. Other files present in the folder will not be read. If such files are missing, they will be evaluated with a score of 0.
- Code either not written in Python or not using PyTorch receives a grade of 0.
- Submission between 00.00 am - 00.10 am of Monday 23rd will be accepted with no reduction.
- Submission between Monday 23rd, 00.11 am - 11.59 pm will be accepted with -30% on your score

- If plagiarism is suspected, TAs and I will thoroughly investigate the situation, and we will summon the student for a face-to-face clarification regarding certain answers they provided. In case of plagiarism, a score reduction will be applied to all the people involved, depending on their level of involvement.
- If extensive usage of AI tools is detected, we will summon the student for a face-to-face clarification regarding certain answers they provided. If the answers are not adequately supported with in-person answers, we will proceed to apply a penalty to the evaluation, ranging from 10% to 100%.

Polynomial regression (with gradient descent) [95 points]

Let $z \in \mathbf{R}$ and consider the polynomial

$$p(z) = \frac{z^4}{100} - z^3 + z^2 - 10z = \sum_{i=1}^{4} z_i w_i \tag{1}$$

where $\boldsymbol{w} = [w_1, w_2, w_3, w_4]^T = [-10, 1, -1, \frac{1}{100}]^T$. This polynomial can be also expressed as the dot product of two vectors, namely

$$p(z) = \boldsymbol{w}^T \boldsymbol{x} \quad \boldsymbol{x} = [z, z^2, z^3, z^4]^T \tag{2}$$

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

Now, assuming that the vector $\boldsymbol{w}$ is unknown, linear regression could estimate it using the dot-product form presented in Equation 2. To achieve this we can move to another dataset

$$\mathcal{D}' := \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N \qquad \boldsymbol{x}^i = [z_i, z_i^2, z_i^3, z_i^4]^T$$

The task of this assignment is to perform polynomial regression using gradient descent with PyTorch, even if a closed-form solution exists.

1. (5 pts) Define a function

   ```
   plot_polynomial(coeffs, z_range, color='b')
   ```

   Where `coeffs` is a `np.array` containing $[w_0, w_1, w_2, w_3, w_4]^T$ (w_0 in this case is equal to 0) `z_range` is the interval $[z_{\text{min}}, z_{\text{max}}]$ of the z variable; `color` represent a color. Report and comment on the plot.

2. (10 pts) Write a function

   ```
   create_dataset(coeffs, z_range, sample_size, sigma, seed=42)
   ```

that generates the dataset $\mathcal{D}'$. Here `coeffs` is a `np.array` containing $[w_0, w_1, w_2, w_3, w_4]^T$, `z_range` is the interval $[z_{\text{min}}, z_{\text{max}}]$ of the z variable; `sample_size` is the dimension of the sample; `sigma` is the standard deviation of the normal distrbution from which ε_i are sampled; `seed` is the seed for the random procedure.

3. (5 pts) Use the code of the previous point to generate data with the following parameters

   - Each z_i should be in the interval $[-3, 3]$
   - $\sigma = 0.5$
   - Use a sample size of 500 for training data and a seed of 0
   - Use a sample size of 500 for evaluation data and a seed of 1

4. (5 pts) Define a function

   ```
   visualize_data(X, y, coeffs, z_range, title="")
   ```

   that plot the polynomial $p(z)$ and the generated data, (train and evaluation), where `X, y` are as returned from the function `create_dataset`, `coeffs` are the coefficient of the polynomial, `z_range` is the interval $[z_{\text{min}}, z_{\text{max}}]$ of the z variable and `title` may be helpful to distinguish between the training and the evaluation plots. Report and comment on the plots.

5. (20 pts) Perform polynomial regression on $\mathcal{D}$ using linear regression on $\mathcal{D}'$ and reports some comments on the training procedure. Consider your training procedure good when the training loss is less than 0.5. You don't need more than 1000 iterations. In particular, explain:

   - How you preprocessed the data.
   - Which learning rate do you use, and why. What happens if the learning rate is too small; what happens if the learning rate is too high, and why.
   - If `bias` should be set as True of False in `torch.nn.Linear` and why.

6. (10 pts) Plot the training and evaluation loss as functions of the iterations and report them in the same plot.

7. (5 pts) Plot the polynomial you got with your estimated coefficient as well as the original one in the same plot.

8. (10 pts) Plot the value of the parameters at each iteration.

9. (15 pts) Re-train the model with the following parameters:

   - Each z_i should be in the interval $[-3, 3]$
   - $\sigma = 0.5$
   - Use a sample size of 10 for training data and a seed of 0

- Use a sample size of 500 for evaluation data and a seed of 1
- Keep the learning rate you chose at the previous point.

Report: A plot with both the training and evaluation loss as functions of the iterations in the same plot and the polynomial you got with your estimated coefficient as well as the original one in the same plot. Comment on what is going on. **Note:** Do not overwrite the code, keep the original training and this new one as two separate parts.

10. (5 pts*) This part is less trivial and should be considered as a separate exercise. Suppose you have to learn the function

$$f(x) = 5\sin(x) + 3$$

and you know that your data are x^i belong to the interval $[-a, a]$. Which performances would you expect using linear regression in the following cases?

(a) $a = 0.01$

(b) $a = 5$

Note: We want to see some code that generates noisy data from f, do the linear regression and we want to see the loss value on it. If you know the mathematics behind it, you can use your knowledge to answer and validate the data, but as this is a Lab exam, we want to see empirical evidence.

Questions [5 points]

Please answer these questions in max 10 lines. In Lecture 3, we have seen an explicit step of the parameter update for one-dimensional linear regression, namely

$$w_{t+1} = w_t - \alpha \frac{\partial \mathcal{L}}{\partial w}(w_t, b_t) \tag{3}$$

$$b_{t+1} = b_t - \alpha \frac{\partial \mathcal{L}}{\partial b}(w_t, b_t) \tag{4}$$

Hint: Do we have any information about the gradient of the loss function with respect to the parameter at the very beginning of the training loop?

(a) Do you think that could be beneficial to choose two different values of α (learning rate) for Equation (3) and (4)?

(b) Explain the difference between choosing two different learning rates *in principle* and what is instead done by the adaptive methods - e.g, Adagrad. If you don't know the method, you can study one of the following resources:

- Section 8.5.1 of Deep Learning Book
- Section 12.7 of Dive Into Deep Learning