Machine Learning per il Calcolo Scientifico

Fifth laboratory exercises

General guide

For each laboratory, an incomplete Python notebook will be provided with exercises (steps) that must be completed in the given order (some of the exercises will be needed in future laboratories). In step zero, all the Python packages that are needed in order to complete the notebook are listed. This PDF includes the text for the exercises and the expected outcomes for each step. While following the notebook is recommended, you are also welcome to attempt the exercises without using it.

Step Zero

Here are the required Python (`https://www.python.org/`) packages for this laboratory:

- PyTorch (`https://pytorch.org/`)
- Numpy (`https://numpy.org/`)
- Matplotlib (`https://matplotlib.org/`)

Step one: Load the dataset

In this laboratory we consider the Allen-Cahn equation:

$$u_t = \Delta u - \varepsilon^2 u(u^2 - 1), \quad u \in \mathbb{R} \times \mathbb{R}_{>0}.$$

The operator that we wish to learn is

$$\mathcal{G} : u(\cdot, t = 0) \mapsto u(\cdot, t = 1).$$

a. Download the data (see the files AC_data_input.npy and AC_data_output.npy) and load it in the notebook.

b. Plot the dimensions of each tensor to ensure that the dataset has been created correctly and take a look plotting one input-output pair plotting it.

c. Create a DataLoader, dividing the dataset in batches of 10 elements.

Step two: Define the FNO model

a. Define the spectral convolution block, this correspond to do the Fourier transform of the input, multiply with a coefficients tensor and then do the inverse Fourier transform of the result. For the Fourier transform and inverse Fourier transform you can use the functions 'torch.fft.rfft' and 'torch.fft.irfft' respectively.

b. Define the Fourier Neural Operator, this correspond to apply the spectral convolution block multiple times with the lifitng operator at the beginning and the projection operator at the end.

c. Introduce a FNO with 3 layers, tanh activation function, hidden dimension (width) equal to 64 and 16 Fourier modes.

Step three: Train the FNO

a. Define a function to train the FNO using the Adam optimizer with a learning rate of 0.001 for 50 epochs and halving the learning rate every 50 epochs and use the relative L^2 norm as loss function (for the loss use the function LprelLoss provided in the utilities.py file). During the training process save the value of the loss function.

b. Make a figure of the loss function and check that it decreases.

c. Plot an example of the approximation with the corresponding true solution and check the results obtained.

d. You can run the training process multiple times (without restarting the notebook) to augment the number of epochs of training and check both about the convergence of the loss function and the resulting approximation plots.