2AMM10 2021-2022 Assignment 2: Particle trajectory modeling with machine learning

1 Introduction

Experiments and computer simulations are instrumental in the physical sciences and in engineering. Typically, a large number of (simulation) experiments are necessary to gain insight in how a set of parameters affect the experiment's outcome. For example, a mechanical engineer may perform a large amount of computer simulations to study how different materials respond to stress, and a drug developer may perform a large amount of lab experiments before finding a chemical substance that leads to the desired response. The main limitation of such approaches is that experimental capacity is inherently limited, and that accurate simulations are often computationally intensive. This opens the possibility for machine learning to accelerate the experimental process.

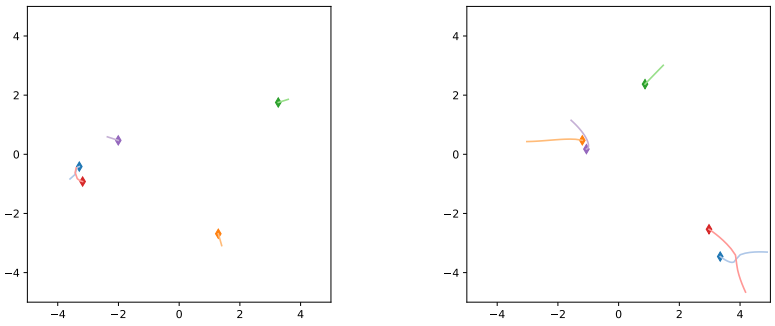
In this assignment, we use deep learning methods to investigate the dynamics of charged particles evolving in a plane over time. The movement of individual particles is determined by its relative location and charge w.r.t. other objects in the plane. The assignment is divided into two parts. In part 1, we study the dynamics of a system of $n = 5$ particles interacting with each other, aiming to approximate long-term dynamics with a deep learning approach. In part 2, we study the case of $n = 1$ particle interacting with 3 charges that are present at fixed locations in the plane, aiming to predict these charges from the particle's path. Additionally, we aim to learn how to continue this particle's path. Both parts are described in more detail in their respective sections.

2 Part 1

In part 1, we aim to model how a system of n particles $p_1, p_2, \ldots, p_n$ moving in the plane evolves over time. At time t, the system is described by the position $\mathbf{x}_i^t \in \mathbb{R}^2$ and velocity $\mathbf{v}_i^t \in \mathbb{R}^2$ of each particle p_i. Each particle has an associated charge $c_i \in \{-1, 1\}$ that remains constant over time. All initial positions $\mathbf{x}_i^0$ and velocities $\mathbf{v}_i^0$ are given, and at any time t, the evolution of the system is governed solely by the forces of all pairs of particles acting upon each other. For any two particles, the force between them depends on their charges and relative locations.

2.1 Data

The data is generated by randomly initializing preliminary positions and velocities of $n = 5$ particles, as well as their charges. The system is then simulated for three seconds as a 'warm up' phase; $t = 0$ reflects the time directly after finishing the warm up. Then, the system is simulated for another 1.5 seconds, after which the simulation run terminates. Example simulations are depicted in Figure 1.



(a) From $t = 0$ to $t = 0.5$

(b) From $t = 0$ to $t = 1.5$

Figure 1: Two simulations of five charged particles interacting with eachother. The diamond indicates the starting position, whereas the line displays the trajectory.

The simulation procedure itself consists of discretizing time into points with interval $\Delta t > 0$. Smaller Δt leads to more accurate simulations, but longer computation time; the dataset for this assignment is generated using $\Delta t = 0.001$. At each step, the forces of the particles acting upon each other are calculated. The acceleration of each particle is proportional to the sum of the forces acting upon it. Then, the particles' velocities are updated using the calculated accelerations $\mathbf{a}_i^t \in \mathbb{R}^2$: $\mathbf{v}_i^t = \mathbf{v}_i^{t-\Delta t} + \mathbf{a}_i^t \cdot \Delta t$. Finally, the positions are updated: $\mathbf{x}_i^t = \mathbf{x}_i^{t-\Delta t} + \mathbf{v}_i^t \cdot \Delta t$.

There are three partitions of the data for this task: a training set, consisting of 10000 simulations, a validation set consisting of 2000 simulations, and a test set consisting of 2000 simulations. A function for loading the data is provided with the assignment.

2.2 Assignment Tasks

The goal of part 1 is to investigate if machine learning can be used for the task of predicting the positions of the n particles at a time t from the **initial** positions and velocities ($t = 0$). The motivation of this investigation is twofold: on the one hand, exploring whether or not machine learning is capable to predict the future state of a complex dynamical system is interesting in itself from a research perspective; on the other hand, a more practical motivation is that reliably predicting the particles' future positions would enable 'shortcutting' the many sequential computational steps that are necessary for a physics-based simulation, hopefully reducing computation times at the cost of low prediction error. A more detailed description of the tasks and concrete deliverables can be found in Section 4.

3 Part 2

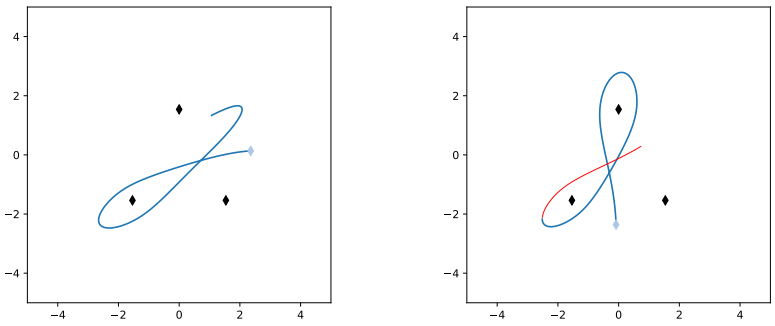
In part 2, we study the dynamics of a single particle p_1 moving around in a plane with three charges c_2, c_3, c_4 present at fixed positions. One can consider this a special case of a particle system with $n = 4$ as described in part 1, but with the position of particles p_2, p_3, p_4 fixed. One difference is that we fix the charge of particle p_1 to 1, and charges c_2, c_3, c_4 are now sampled uniformly from the *interval* $[-1, 0]$, i.e. they are no longer sampled from set $\{-1, 1\}$.

Part 2 is split into two subtasks: **(2.1)** Predicting the values of charges c_2, c_3, c_4 by looking at p_1's trajectory, and **(2.2)** predicting a continuation of p_1's trajectory by looking at its initial trajectory.

3.1 Data

The datasets are generated by shooting a particle into the plane such that it can interact with the three fixed charges. The three charges are placed in a triangle configuration around the center of the plane. p_1 is shot in from a random position on the side of the plane, with an initial velocity pointing towards the approximate center of the plane.

We generate simulations of varying length. All simulations start at $t = 0$ and end at $t = 10 \pm 1$. For task **(2.1)**, we provide these simulations alongside the values of charges c_2, c_3, c_4, such that we can predict the charges from the simulation. For task **(2.2)**, we continue a subset of simulations for an additional 4 ± 2 seconds, i.e. ending the entire simulation at $t = 14 \pm 3$. We provide these two pieces of the simulations separately, such that one forms the initial path to be provided, and the other the path we wish to predict. Example simulations are depicted in Figure 2.



(a) A simulation as used to infer the charge values, as used in task (2.1).

(b) A simulation and its extended trajectory (in red), as used in task (2.2).

Figure 2: Two simulations of a particle interacting with three fixed charges. The blue diamond indicates p_1, whereas the black diamonds indicate fixed charges c_2, c_3, c_4.

The simulations are computed following the same procedure as task 1. We provide the simulations sampled with $\Delta t = 0.1$, i.e., 100 ± 10 steps for the first part and an additional 40 ± 20 steps for the second part. We provide only positions $\mathbf{x}_1^t \in \mathbb{R}$ of particle p_1.

We consider the problem setting of having only a limited amount of data to train, consequently, you should consider models with inductive biases well aligned with the data. For each subtask we provide three partitions of the data. For task **(2.1)**, 800 simulations for training, 100 for validation and 100 for testing. We provide simulations of 100 ± 10 steps alongside charges c_2, c_3, c_4. For **(2.2)**, 150 simulations for training, 100 for validation and 100 for testing. We provide simulations of 100 ± 10 steps alongside their continuation of 40 ± 20 steps. A function for loading the data is provided with the assignment.

3.2 Assignment Tasks

The goal of part 2 is to investigate whether we can infer information about a system with one particle moving in a field with charges, using deep learning methods. We do this in two subtasks:

(2.1) Using the positions of positively charged particle p_1 during a simulation, predict the values of negative charges c_2, c_3, c_4.

(2.2) Using the positions of positively charged particle p_1 during a simulation up to $t = 10 \pm 1$, continue its trajectory for an additional 4 ± 2 seconds.

The first subtask explores whether we can train a model to infer certain properties from a system by using its simulations and these desired properties. The second subtask explores

whether we can train a model to learn the dynamics of a given simulation, and continue it accurately.

For this data, note that it is likely not possible to get near-perfect performance. Some simulation paths might not cover enough of the system to contain full information about the environment. You should take this into account for your evaluation and discuss the results in this context.

4 Deliverables

The code for parts 1 and 2 should be handed in separately. Skeleton files that load the provided data for each task are provided.

For the report, you should hand in a single report for all tasks. Each (sub)task should get its own section, that is, one for **(1)**, **(2.1)** and **(2.2)**. Note that you are allowed to refer to content of another section, i.e., you do not have to re-explain data or methodologies if you share them between different (sub)tasks.

Each task section should contain the following subsections:

- **Problem formulation.** Develop a machine learning problem formulation for the task. What kind of machine learning task fits the problem description? How can you measure whether or not the model adequately performs this task?
- **Model formulation.** Develop a model that fits the task. Here, your model should be formulated on a high level using mathematical notation and accompanying explanations. Motivate your model by explaining how relevant characteristics and symmetries of the data are accommodated for by your design choices.
- **Implementation and training.** Implement your model in PyTorch, and describe the implementation details. Implement and describe the procedure(s) you used to train the model. Your report should concisely describe all necessary information for someone to reproduce your implementation(s) and training procedure(s) from scratch. This includes, but is not limited to, the number and type of layers in your neural network(s), activation functions, loss function, optimization algorithm, etc.
- **Experiments and results.** Conduct suitable experiments and obtain results to evaluate the performance of your model. To assess the quality of your approach in different settings, you should do the following (for the corresponding task):

  (1)
  - Evaluate on varying training set sizes and for varying prediction horizons. The amount of training samples should vary from 10^2 to 10^4, and the prediction horizons are $t = 0.5$, $t = 1$ and $t = 1.5$. For each training set size/time horizon combination, compare to a simple linear baseline where $\mathbf{x}_i^t = \mathbf{x}_i^0 + \mathbf{v}_i^0 \cdot t$.
  - Come up with and execute one additional experiment that provides an interesting insight in your method.

  (2.1)
  - Evaluate the performance of your method. Think carefully about the metrics you use for evaluation with respect to the problem statement. On top of quantitative evaluations, you should investigate some cases where your model fails, and describe why you think this is the case.

  (2.2)
  - Evaluate the performance of your method. Besides quantitative performance, investigate a few individual cases; at least one where the model does well, alongside one where it does not. Also, compare your model to a simple linear baseline, i.e., extrapolating in the direction between the last 2 timesteps of the input simulation ($\mathbf{x}_i^t - \mathbf{x}_i^{t-\Delta t}$).

  Carefully describe the experiment setup and report and interpret your results (on the train, validation and test sets).

- **Conclusion.** Summarize your approach and results, and concisely discuss limitations and some directions for future research.

A skeleton file for the report is provided, although you are not obligated to use this format. *Do* make sure that all the (sub)sections in this skeleton are present in your submitted report.