Chapter 1

Machine Learning Analysis

1.1 Methods

The objective of this part is to build a machine learning-based tool to infer the interaction potential starting from the history of positions and velocities of an ensemble of Active Brownian Particles. As explained in section **??**, one could use as input some global attribute of the system, such as the radial distribution function, but, as showed by **bag_interaction_2021**, this approach does not work well in out-of-equilibrium cases where an active velocity is present.

1.1.1 The Graph Neural Network

Here, we tried to replicate the approach used by **ruiz-garcia_discovering_2024**, where a vector of positions and orientations of a set of ABPs is given as input to a Graph Neural Network which tries to predict resulting velocities. Our GNN is structured as explained in section **??**, namely with a node and a message function. Both of them have 4 layers, with 300 hidden nodes; each hidden layer has a ReLU (Rectified Linear Unit) activation function, while the output layer of each Network is a simple linear layer without any activation. As for Figure **??**, the input layer of message function is $2n_f$ dimensional, where n_f is the number of single particle features, namely x, y, θ to have information about positions and orientations of a pair of interacting agents. Messages are aggregated using sum as an aggregation function (`'add'` in PyG jargon) to respect the physics of the problem. Node function's input dimension is $n_f + n_m$ where n_m is the output dimension of the message function; this is done because node function has the purpose of taking the aggregated message from all the senders (particle inside a threshold distance Γ) along with the single features of the receiver and trying to predict receiver's velocity (acceleration in Newtonian dynamics).

Here it is important to note that the history is not relevant: the network just takes one instant at a time and predicts instantaneous velocities starting from positions and orientations, without knowing what happens before or after. Instantaneous velocities are computed dividing the difference of consecutive positions by the integration time interval, then they get checked for big jumps, that happen when periodic boundary condition correction take place.

The problem of message dimension n_m is tackled in section **??**. Here, we stuck to a 2D message dimension, as is reported in the reference paper [**ruiz-garcia_discovering_2024**].

1.1.2 Simulations

To understand the role of the potential's functional form in prediction results, we tried to train and test the network on two different simulations, one done using a spring potential with $k_s = 0.1\,\mathrm{N\,m^{-1}}$ and $x_0 = 12\,\mu\mathrm{m}$ and the other using a LJ potential with $\sigma = 4\,\mu\mathrm{m}$ and $\epsilon = 0.01\,\mathrm{pJ}$. These potential have the difference in range: without limiting our interaction threshold radius, elastic force not only has effect at long distance but its absolute value increases, while LJ is a short range potential. We expect the network to perform worse on LJ since it has less relevant interactions to learn from.

These two potentials required different integration steps: $10^{-3}\,\mathrm{s}$ for LJ and $5 \times 10^{-2}\,\mathrm{s}$ for spring potential. Both simulations were ran with a velocity of $15\,\mu\mathrm{m\,s^{-1}}$, in a $100\,\mu\mathrm{m} \times 100\,\mu\mathrm{m}$ box with 100 particles of $2\,\mu\mathrm{m}$ in radius.

1.2 Training and Testing

We let the two systems evolve for a total simulated time of 1000 s, then instantaneous velocities were computed and finally a sample of 1000 equally spaced snapshots was taken from each simulation. We divided these snapshots in 750 for training and 250 for testing. The network was trained and tested on each simulation's data separately to have preliminary results.

After training, a minimization process is used to find the best linear transformation that maps message components into the known forces.

1.2.1 Spring Potential

1.2.2 Lennard Jones Potential

1.3 Conclusions