Discussions about the project during "UCA School on Complex systems 2021"

L. Berti, Z. El Kiyati, S. Reidel; L. Giraldi, C. Prud'homme

October 21, 2021

1 19 October

1.1 Three sphere swimmer far from walls

Possible actions

- Shrink 1 arm
- Elongate 1 arm

Possible States

- $(L_1, L_2) \in \{A, B\}^2$ with A, B fixed values

Reward

- $\delta_t = X^{CM}_{t+1} - X^{CM}_t$ every step
- $R = \sum_t \gamma^t \delta_t$ cumulative reward

Procedure for reinforcement learning that was followed:

- Start with state s_t and action a_t (chosen via Q-learning)
- Simulate passing between s_t and s_{t+1} as specified by a_t in unit time
- Update the Q-function with the rule $Q^\pi(s_t, a_t) \leftarrow Q^\pi(s_t, a_t) + \alpha(r_t + \gamma \max_a Q^\pi(s_{t+1}, a) - Q^\pi(s_t, a_t))$.

Precise values we looked at (during M2 project):

- R=1 radius of the spheres
- B=10 maximal length of the arms
- A=6 minimal length of the arms
- $\Delta = 0.16R$ Displacement after one swimming period (L=Long Arm, S=Short Arm; Sequence during swimming period: LL, SL, SS, LS, LL)

1.2 Three sphere swimmer next to a wall

Experimentally, swimmers have high affinity to the walls, they stick to it. Then, depending on the angle, they can be deflected.

We discussed about different aspects to describe our problem:

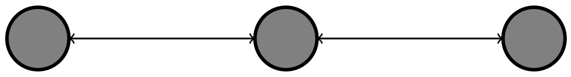

Figure 1: Three sphere swimmer, far from walls.

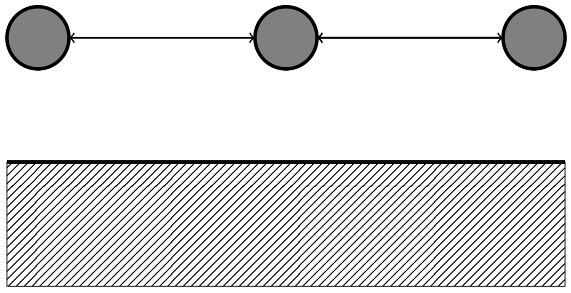

Figure 2: Three sphere swimmer, close to a wall.

Reward: Swim in the x direction and, at the same time, try to maintain the distance H from the wall

$$r_t = \frac{(\vec{X}^{t+1}_{CM} - \vec{X}^{t}_{CM}) \cdot \hat{e}_x}{|y_{CM} - H|}$$

Actions: Discrete action space, where you can choose how much you can shrink your arm

$$A = \{2, 4, 6, 8\} \quad \text{Supposing that the initial length is 10}$$

The way to deal with continuous actions spaces is to put a probability distribution over the space, learn it and use it to choose the future states. A possible algorithm is Normalised Advantage Function (NAF). However, it is sensibly more difficult than "just" continuous state spaces.

States: (in other words, the states describing the swimmer)

- Mass centres of the spheres $(X^1_{CM}, X^2_{CM}, X^3_{CM})$

NB: the orientation of the swimmer can be computed using the centres of mass, as well as the distance from the wall (and in principle also the lengths of the arms). However, when looking at function approximation, inputting also these other variables might help learning.

Question: *Does the problem have the Markov property (i.e. $\mathbb{P}[s_{t+1}|s_0, a_0, s_1, a_1, \dots, s_t, a_t,] = \mathbb{P}[s_{t+1}|s_t, a_t]$)? Depending on the state that describes the problem, it might or might not have it. A way to check this is to test the independence of the future with respect to the past via a test of independence. The test is the following:*

- *Take a collection of samples of your state $s_t = (X^1_{CM}, X^2_{CM}, X^3_{CM})$ in a discrete set $\{a, b, c, \dots\}$*
- *Compute the evolution of your collection of samples*
- *Realise a test of independence via a χ^2 test $\forall x \in \{a, b, c, \dots\}$ of the past and future, i.e. (s_{i-1}, s_{i+1}) and verify you always have a positive result. If not, the process does not have the Markov property as the future is not independent from the past.*

Discretization of the problem? If we suppose the Markov property, we could think in a first approximation to discretise the state space and attempt a Q-learning algorithm (i.e. matrix representation of the Q-value function). The idea is to build a grid over the xy plane and use the midpoint of each square as the approximation every time the continuous state variable happens to fall in the square.

However, since the state corresponds to the three centres of mass of the sphere, each in $\mathbb{R}^2$, the size of the discretized state space rapidly explodes if we want to make a good discretization of the 2d region in which the swimmer evolves.

Continuous state problem. In this case, we allow the state variables (centres of mass of the three spheres) to vary continuously in spaces of the form $[A_i, B_i] \times [C_i, D_i]$, for $i = 1, 2, 3$.

1. The first approach we will look at is a ***linear function approximation of*** Q of the form $Q(s,a) \approx \hat{Q}(s,a;w) =< x(s,a), w >$ where $\hat{Q}$ is the approximation of Q, w are the weights of the linear estimator, $x(s,a)$ are feature vectors (i.e. combinations of state and action that best describe our problem).

   In this first approach, we will look at an episodic treatment of our problem: we fix a "finish line" at $X = 0.5$, for example, and we reward the swimmer based on the velocity with which this finishing line is attained. The best policy we will look for will be the one allowing to get there faster. We ask

   - $|X_{t_F} - x_{CM,t_0}| > 0.5$
   - a the same time, fix a maximal time for which one episode is allowed to run, for example $t \leq 10T$ where T is the cycle period of the swimming stroke.

   The communication with the fluid solver is always performed by reading the result file (to get the centres of mass) and by modifying the input files (to prescribe the new action) until the final total time is reached, and this for each episode.

2. The second approach we will look at is the ***non-linear function approximation*** of Q using neural networks. During our investigation, we will try with Deep Q-Network (inspired by paper [1]) and with Deep Q-network with Long-Short Term Memory layers (inspired by paper [3]).

2 20 October

2.1 Linear continuous model

The first step now is to construct the *feature vector* $x(s,a)$, function fo the state and actions that should encode some sort of interaction informations at the input of the model. In order to do so, we chose a tiling code approach (see [2, Section 9.5]) that uses a finite collection of tiling square patterns to locate regions in the continuous space where the update of the Q function should be "extended".

A tiling code algorithm is proposed in Python by R. Sutton in its website[1]. It outputs a vector v of $N_{tilings}$ components, whose value v_i is the global index of the tile among the whole collection of tiles, and at the same tile belongs to tiling i.

Once the feature vector has been constructed via tiling code, we apply an episodic semi-gradient[2] SARSA method to approximate the Q function with a linear model $\hat{Q}$ [2, Sec 10.1] as reported in 1. In the algorithm, $\hat{Q}(s,a;w) =< x(s,a), w >$ is the linear approximation of the Q function, hence $\nabla_w \hat{Q} = x(s,a)$.

[1] `http://incompleteideas.net/tiles/tiles3.html`

[2] "Semi-gradient" because there is not a gradient term with respect to weights of $\hat{Q}(S', A'; w)$, and a not a complete gradient of the $(R + \hat{Q}(S', A'; w) - \hat{Q}(S, A; w))^2$ loss function is computed.

Algorithm 1 Episodic semi-gradient SARSA approximation of $\hat{Q}$

```
Require: $\alpha > 0, \epsilon > 0, w \in \mathbb{R}^d$, $\hat{Q}$ parametrization (linear for us)
  for $i$ in $N_{episodes}$ do
    Choose $S, A$ initial state and action of the episode (use $\epsilon$-greedy)
    for $j$ in $N_{step}$ do
      Take action $A$ and observe $S', R$
      if $S'$ is terminal then
        $w \leftarrow w + \alpha[R - \hat{Q}(S, A; w)]\nabla_w \hat{Q}(S, A; w)$
        Go to next episode
      end if
      Choose $A'$ as a function of $\hat{Q}(S', \cdot; w)$ via $\epsilon$-greedy scheme
      $w \leftarrow w + \alpha[R + \gamma\hat{Q}(S', A'; w) - \hat{Q}(S, A; w)]\nabla_w \hat{Q}(S, A; w)$
      $S \leftarrow S'$
      $A \leftarrow A'$
    end for
  end for
Ensure: $w$
```

References

[1] G. Novati, S. Verma, D. Alexeev, D. Rossinelli, W. M. van Rees, and P. Koumoutsakos, *Synchronisation through learning for two self-propelled swimmers*, Bioinspiration & Biomimetics, 12 (2017), p. 036001.

[2] R. Sutton and A. Barto, *Reinforcement Learning, second edition: An Introduction*, Adaptive Computation and Machine Learning series, MIT Press, 2018.

[3] S. Verma, G. Novati, and P. Koumoutsakos, *Efficient collective swimming by harnessing vortices through deep reinforcement learning*, arXiv:1802.02674 [physics], (2018). arXiv: 1802.02674.