A note for green hand in Q - learning[1]

Reproduction letter: Flow Navigation by Smart Microswimmers via RL

这份文档总结了部份入门强化学习的基本概念，但它并不是一份教材。教材往往内容丰富且知识体系完整，而这可能也常常对应着更加冗长的内容、有意义但并不实用的算法和超出需要的数学严谨。对于具备一定数学基础，却并不曾真正接触过强化学习，又渴望避开繁琐的细节，快速理解其核心数学定义并上手体验强化学习在科学中应用的同学，我希望这份笔记可以帮到你。

正如题目所写，这是一份通往理解Q-learning算法的学习笔记，虽然这里只介绍了一个算法，但学习这份笔记所带来的理解将会对理解其他强化学习算法十分有帮助。笔记第一部份会向你介绍强化学习中你会用到的的一些基本定义和定理，理解时序差分学习（一类强化学习算法）中很常用的一种算法——Q-learning。第二部份会向你展示一篇论文的复现作为应用的例子，查看它的代码并在这份代码基础上玩一玩会对你理解算法和消除陌生感有很大帮助。

注意，这份笔记建立的时候，我就和现在的你一样是新人，因而笔记中一定存在着大大小小的问题，如果你在阅读这份笔记过程中感到困惑或难以理解，请相信你自己，可以以此为线索对不理解的地方单独查找和学习，也可以先行阅读其他笔记或教材而将此笔记当作一份cheatsheet；也欢迎任何人指出问题或帮助改进和完善这一笔记，以帮助之后的人。

最后，祝你学得愉快，尽管可能会遇到困难，请永远不要放弃。

This document summarizes some basic concepts of introductory reinforcement learning, but it is not a textbook. Textbooks are often rich in content with a complete knowledge system, which also often corresponds to longer content, meaningful but impractical algorithms, and mathematical rigor beyond necessity. For those with a certain mathematical foundation, who have not really encountered reinforcement learning, and wish to avoid cumbersome details to quickly grasp its core mathematical definitions and experience its application in science, I hope this note can help you.

As the title suggests, this is a learning note leading to an understanding of the Q-learning algorithm. Although only one algorithm is introduced here, the understanding gained from studying this note will be very helpful in understanding other reinforcement learning algorithms. The first part of the note will introduce you to some basic definitions and theorems used in reinforcement learning, and understand Q-learning, a commonly used algorithm in temporal difference learning (a type of reinforcement learning algorithm). The second part will show you the reproduction of a review letter as an example of application, examining its code and playing around with it based on this code will greatly help you understand the algorithm and eliminate unfamiliarity.

Please note, when I compiled these notes, I was a novice just like you are now. There are definitely issues, big and small, in the notes. If you feel confused or have difficulty understanding while reading these notes, trust yourself. You can use this as a clue to look up and learn about things you don't understand independently, or you can read other notes or textbooks first and treat these notes as a cheatsheet. I also welcome anyone to point out issues or help improve and perfect these notes, to assist those who come after.

Finally, I wish you a pleasant learning experience. Despite the potential difficulties, never give up.

Below are the textbooks I referred to during my study (highly recommended, almost all of the content of the notes comes from them) and the paper reproduced in the second part, thanks to the authors of these works. I list them here for easy reference and learning.

- "Mathematical Foundations of Reinforcement Learning." by Shiyu Zhao
- "Flow Navigation by Smart Microswimmers via Reinforcement Learning", C. Simona and G. Kristian and C. Antonio and B. Luca, PhysRevLett.118.158004 (2017).

[1] Compiled by Yono (Github: feizhanxia)

I. Key points for understanding Q - Learning

1. Markov Decision Process (MDP)

What is MDP?

`Markov decision process is Markov decision process.`

Process

Sets :

State space $\mathcal{S}$

Action space $\mathcal{A}(s)$, $s \in \mathcal{S}$

Reward set $\mathcal{R}(s,a)$, $s \in \mathcal{S}$, $a \in \mathcal{A}$

Model:

State transition probability $p(s' \mid s,a)$, $\Sigma_{s' \in \mathcal{S}} p(s' \mid s,a) = 1$

Reward probability $p(r \mid s,a)$, $\Sigma_{r \in \mathcal{R}(s,a)} p(r \mid s,a) = 1$

Decision

Policy:

$$\pi(a \mid s), \sum_{a \in \mathcal{A}(s)} \pi(a \mid s) = 1$$

Markov

Markov property:

$$p(s_{t+1} \mid s_t, a_t, s_{t-1}, a_{t-1}, \ldots, s_0, a_0) = p(s_{t+1} \mid s_t, a_t)$$

$$p(r_{t+1} \mid s_t, a_t, s_{t-1}, a_{t-1}, \ldots, s_0, a_0) = p(r_{t+1} \mid s_t, a_t)$$

2. Sate value and action value

One step:

$$S_t \xrightarrow{A_t} S_{t+1}, R_{t+1}\,.$$

Random variables S_t, $S_{t+1} \in \mathcal{S}$, $A_t \in \mathcal{A}(S_t)$, and $R_{r+1} \in \mathcal{R}(S_t, A_t)$

A state-action-reward trajectory:

$$S_t \xrightarrow{A_t} S_{t+1}, R_{t+1} \xrightarrow{A_{t+1}} S_{t+2}, R_{t+2} \xrightarrow{A_{t+2}} S_{t+3}, R_{t+3} \ldots$$

Define a random variable **discounted return**:

$$G_t \equiv R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots,\ \gamma \in (0,1)$$

γ is the discount rate, which determines the importance of future rewards.

State value of a state s is expected value of discounted return at state s :

$$v_\pi(s) \equiv \mathbb{E}\left[G_t \mid S_t = s\right]$$

Action value is expected value of G_t at state s taking action a:

$$q_\pi(s,a) \equiv \mathbb{E}\left[G_t \mid S_t = s, A_t = a\right]$$

We can see that,

$$v_\pi(s) = \sum_{a\in\mathcal{A}} \pi\left(a \mid s\right) q_\pi(s,a) \ .$$

3. Bellman Equation

Bellman equation in terms of state values

$$\begin{aligned} v_\pi(s) &= \mathbb{E}\left[R_{t+1} \mid S_t = s\right] + \gamma \mathbb{E}\left[G_{t+1} \mid S_t = s\right] \\ &= \underbrace{\sum_{a\in\mathcal{A}} \pi\left(a \mid s\right) \sum_{r\in\mathcal{R}} p(r \mid s,a) r}_{\text{mean of immediate rewards}} + \underbrace{\gamma \sum_{a\in\mathcal{A}} \pi\left(a \mid s\right) \sum_{s'\in\mathcal{S}} p(s' \mid s,a) v_\pi(s')}_{\text{mean of future rewards}} \\ &= \sum_{a\in\mathcal{A}} \pi\left(a \mid s\right) \left[\sum_{r\in\mathcal{R}} p(r \mid s,a) r + \gamma \sum_{s'\in\mathcal{S}} p(s' \mid s,a) v_\pi(s') \right] \end{aligned}$$

Bellman equation in terms of action values

$$q_\pi(s,a) = \sum_{r\in\mathcal{R}} p(r \mid s,a) r + \gamma \sum_{s'\in\mathcal{S}} p(s' \mid s,a) \sum_{a'\in\mathcal{A}} \pi\left(a' \mid s'\right) q_\pi(s',a')$$

4. Bellman optimality equation (BOE)

Definition. Optimal policy and optimal state value.

A policy π^* is **optimal** if $v_{\pi^*}(s) \geq v_\pi(s)$ for all $s \in \mathcal{S}$ and for any other policy π. The state values of π^* are the **optimal state values**.

The element-wise expression of **Bellman Optimality Equation.**

$$\begin{aligned} v(s) &= \max_{\pi(s)\in\Pi(s)} \sum_{a\in\mathcal{A}} \pi\left(a \mid s\right) \left(\sum_{r\in\mathcal{R}} p(r \mid s,a) r + \gamma \sum_{s'\in\mathcal{S}} p(s' \mid s,a) v(s') \right) \\ &= \max_{\pi(s)\in\Pi(s)} \sum_{a\in\mathcal{A}} \pi\left(a \mid s\right) q(s,a) \end{aligned}$$

Matrix-vector form of the BOE

$$v = \max_{\pi\in\Pi} \left(r_\pi + \gamma P_\pi v\right)$$

with,

$$v \in \mathbb{R}^{|\mathcal{S}|}, \quad \left[r_\pi\right]_s \equiv \sum_{a\in\mathcal{A}} \pi\left(a \mid s\right) \sum_{r\in\mathcal{R}} p(r \mid s,a) r, \quad \left[P_\pi\right]_{s,s'} \equiv p(s' \mid s) = \sum_{a\in\mathcal{A}} \pi\left(a \mid s\right) p(s' \mid s,a)$$

Denote right-hand side of BOE

$$f(v) \equiv \max_{\pi\in\Pi} \left(r_\pi + \gamma P_\pi v\right) \ ,$$

Then the BOE can be express in

$$v = f(v)$$

Contraction mapping theorem[2]

For any equation that has the form $x = f(x)$ where x and $f(x)$ are real vectors, if $f : X \to X$ is a contraction mapping,

$$\text{i.e. } \forall x_1, x_2 \in X,\ \exists \gamma \in (0,1),\ \text{s.t.} \quad \left\| f(x_1) - f(x_2) \right\| \leq \left\| x_1 - x_2 \right\|$$

then the following properties hold;

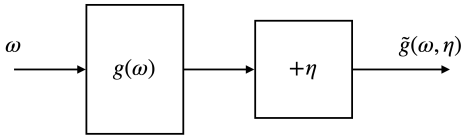
- *Existence: There exists a fixed point x^* satisfying $f(x^*) = x^*$.*
- *Uniqueness: The fixed point x^* is unique.*
- *Algorithm: Consider the iterative process: $x_{k+1} = f(x_k)$ where $k = 0,1,2,\ldots$*

Then, $x_k \to x^*$ as $k \to \infty$ for any initial guess x_0. Moreover the convergence rate is exponentially fast.

Contraction property of $f(v)$ in BOE

The function $f(v)$ on the right-hand side of the BOE is a contraction mapping. In particular, for any $v_1, v_2 \in \mathbb{R}^{|\mathcal{S}|}$, it holds that $\left\| f(v_1) - f(v_2) \right\|_\infty \leq \left\| v_1 - v_2 \right\|_\infty$ where $\gamma \in (0,1)$ is the discount rate, and $\| \cdot \|_\infty$ is the maximum norm. (*Proof is omitted again.*)

*This contraction property with the contraction mapping theorem make clear the **existence** and **uniqueness** of BOE's solution. And obviously the contraction mapping theorem suggest a **iterative algorithm** for solving v* for BOE, and it has a specific name called **value iteration** (one algorithm in dynamic programming), whose implementation we don't show here.*



A black-box system, only input ω and noisy output $\tilde{g}(\omega, \eta)$ known

5. Robbins-Monro algorithm

The Robbins-Monro (RM) algorithm is a famous stochastic iterative algorithm for root-finding. It can be shown that the famous stochastic gradient descent algorithm is a special form of the RM algorithm.

Aim is to find the root of equation

$$g(\omega) = 0.$$

However, the expression of the function g is unknown (e.g. NN with structure and parameters unknown), and we can only obtain a noisy observation of $g(\omega)$:

[2] Proof is omitted here. The key for proof is to show that the sequence $\{x_k\}$ is Cauchy.

$$\omega_{k+1} = \omega_k - a_k \tilde{g}(\omega_k, \eta_k), \quad k = 1,2,3,\ldots$$

Robbins-Monro theorem.

In the Robbins-Monro algorithm, if

(a) $g(\omega)$ monotonically increasing with finite gradient: $0 < c_1 \leq \nabla_\omega g(\omega) \leq c_2$ for all ω;

(b) a_k should converge to zero but not too fast: $\Sigma_{k=1}^{\infty} a_k = \infty$ and $\Sigma_{k=1}^{\infty} a_k^2 < \infty$;

(c) $\mathbb{E}[\eta_k \mid \mathcal{H}_k] = 0$ and $\mathbb{E}[\eta_k^2 \mid \mathcal{H}_k] < \infty$ where $\mathcal{H}_k = \{\omega_k, \omega_{k-1}, \ldots\}$;

then ω_k almost surely converges to the root ω^* satisfying $g(\omega^*) = 0$.

We don't give proof here. RM theorem will be of use in derivation of Q-learning algorithm.

6. TD learning of optimal action values: Q-learning

The Q-learning algorithm is

$$q_{t+1}(s_t, a_t) = q_t(s_t, a_t) - \alpha_t(s_t, a_t)\left[q_t(s_t, a_t) - \left(r_{t+1} + \gamma \max_{a \in \mathcal{A}(s_{t+1})} q_t(s_{t+1}, a)\right)\right],$$

$$q_{t+1}(s, a) = q_t(s, a), \quad \text{for all } (s, a) \neq (s_t, a_t)$$

where $t = 0,1,2,\ldots$ Here, $q_t(s_t, a_t)$ is the estimate of the optimal action value of (s_t, a_t) and $\alpha_t(s_t, a_t)$ is the learning rate for (s_t, a_t).

Derivation of Q-learning algorithm

We aim to solve the Bellman optimality equation, which means to estimate optimal action values and find optimal policies directly.

BOE

$$v(s) = \max_{\pi} \sum_{a \in \mathcal{A}(s)} \pi\left(a \mid s\right)\left[\sum_r p(r \mid s, a)r + \gamma \sum_{s'} p(s' \mid s, a)v(s')\right]$$

which is,

$$v(s) = \max_{a \in \mathcal{A}(s)}\left[\sum_r p(r \mid s, a)r + \gamma \sum_{s'} p(s' \mid s, a)v(s')\right]$$

Then the state value of optimal policy can be denoted as:

$$v(s) = \max_{a \in \mathcal{A}(s)} q(s, a)$$

Then the BOE can be rewritten as

$$\max_{a \in \mathcal{A}(s)} q(s, a) = \max_{a \in \mathcal{A}(s)}\left[\sum_r p(r \mid s, a)r + \gamma \sum_{s'} p(s' \mid s, a) \max_{a \in \mathcal{A}(s')} q(s', a)\right]$$

Thus, what we need to solve is exactly the BOE expressed in terms of action values:

$$q(s, a) = \sum_r p(r \mid s, a)r + \gamma \sum_{s'} p(s' \mid s, a) \max_{a \in \mathcal{A}(s')} q(s', a)$$

$$\text{i.e.} \quad q(s, a) = \mathbb{E}\left[R_{t+1} + \gamma \max_a q\left(S_{t+1}, a\right) \mid S_t = s, A_t = a\right]$$

For state s_t we define a function as

$$g\left(q(s_t,a_t)\right) \equiv q(s_t,a_t) - \mathbb{E}\left[R_{t+1} + \gamma \max_a q\left(S_{t+1},a\right) \mid S_t = s, A_t = a\right]$$

Then the BOE is equivalent to

$g\left(q(s_t,a_t)\right) = 0$

We can obtain noisy observation

$$\begin{aligned}
\tilde{g}\left(q(s_t,a_t)\right) &= q(s_t,a_t) - \left[r_{t+1} + \gamma \max_a q(s_{t+1},a)\right] \\
&= \underbrace{\left(q(s_t,a_t) - \mathbb{E}\left[R_{t+1} + \gamma \max_a q\left(S_{t+1},a\right) \mid S_t = s, A_t = a\right]\right)}_{g(q(s_t,a_t))} \\
&\quad + \underbrace{\left(\mathbb{E}\left[R_{t+1} + \gamma \max_a q\left(S_{t+1},a\right) \mid S_t = s, A_t = a\right] - \left[r_{t+1} + \gamma \max_a q(s_{t+1},a)\right]\right)}_{\eta}
\end{aligned}$$

Therefore, the Robbins-Monro algorithm for solving $g\left(q(s_t,a_t)\right) = 0$ is

$$\begin{aligned}
q_{t+1}(s_t,a_t) &= q_t(s_t,a_t) - \alpha_t(s_t,a_t)\tilde{g}\left(q(s_t,a_t)\right) \\
&= q_t(s_t,a_t) - \alpha_t(s_t,a_t)\left[q_t(s_t,a_t) - \left(r_{t+1} + \gamma \max_{a \in \mathcal{A}(s_{t+1})} q_t(s_{t+1},a)\right)\right]
\end{aligned}$$

Here we derived the Q-learning algorithm.

II. Reproduction of the letter

This research considers a model of active particles that engage in self-propelled locomotion with constant speed v_s while being carried by the underlying flow.

The particles can only sense partial information of the environment, knowledge about the direction of local vorticity. And the particles always have knowledge of its own direction.

The particles only have partial control to its direction: the direction of the swimming velocity is determined by competition between a torque to align the particle with its preferred direction and rotation induced by the vorticity of underling flow.

The goal of the particles is to maximize displacement in the upward direction in the long run.

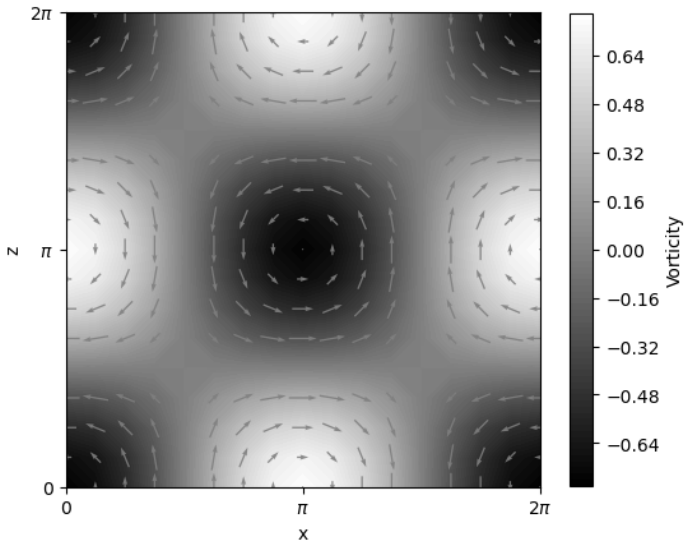
A naive policy is always trying to swim upwards. In the numerical experiments following we will train reinforcement learning agents to be smart particles with a smart policy that has different choices of preferred swimming direction in different areas of the flow field. The algorithm we adopt here is Q-learning.

Then we compare trained smart particles with naive particles to show the RL algorithm allows particles to learn effective strategies in complex flows, which illustrates the potential of RL algorithms to solve difficult navigation problems.

1. Plot the fluid environment

Taylor-Green Flow:

$$\boldsymbol{u} = \left(\frac{u_0}{2}\right)\left[-\cos x \sin z, \sin x \cos z, 0\right] = \nabla \times \boldsymbol{\psi}, \quad \text{where } \boldsymbol{\psi} \equiv \left(-\frac{u_0}{2}\cos x \cos z\right)\hat{\boldsymbol{y}}$$



Vorticity:

$$\boldsymbol{\omega} = \nabla \times \boldsymbol{u} = \left(u_0 \cos x \cos z\right)\hat{\boldsymbol{y}}$$

TGF can be considered a model for convection in two dimensions, with research before revealing the existence of different regimes for gyrotactic particles in it.

2. Dynamical model

The position of the particle is determined by superposition of the fluid velocity $\boldsymbol{u}$ and the swimming velocity $v_s\boldsymbol{p}$ (constant speed v_s and changing direction $\boldsymbol{p}$),

$$\dot{\boldsymbol{x}} = \boldsymbol{u} + v_s\boldsymbol{p} + \sqrt{2D_0}\boldsymbol{\eta} = u_0\left(\hat{u} + \Phi\boldsymbol{p}\right) + \sqrt{2D_0}\boldsymbol{\eta}$$

where $\hat{u} = \left(\frac{1}{2}\right)\left[-\cos x \sin z, \sin x \cos z\right]$, $u_0\Phi = v_s$. Here, D_0 is the transitional diffusivity and $\boldsymbol{\eta}$ is Gaussian white noise.

The direction of the swimming velocity $\boldsymbol{p}$ is determined by competition between preferred direction $\boldsymbol{k}_a$ and local vorticity,

$$\dot{\boldsymbol{p}} = \frac{1}{2B}\left[\boldsymbol{k}_a - \left(\boldsymbol{k}_a \cdot \boldsymbol{p}\right)\cdot \boldsymbol{p}\right] + \frac{1}{2}\boldsymbol{\omega} \times \boldsymbol{p} + \sqrt{2D_R}\boldsymbol{\xi} = \frac{1}{2B}\left(\left[\boldsymbol{k}_a - \left(\boldsymbol{k}_a \cdot \boldsymbol{p}\right)\cdot \boldsymbol{p}\right] + \Psi\hat{\omega} \times \boldsymbol{p}\right) + \sqrt{2D_R}\boldsymbol{\xi}$$

where $\hat{\omega} = \cos x \cos z\, \hat{\mathbf{y}}$, $\Psi = B\omega_0$. Here, D_R is the rotational diffusivity and $\boldsymbol{\xi}$ is Gaussian white noise.

Naive particles obeys its naive policy with only one single $\boldsymbol{k}_a$ that always pointing upwards, while smart particles are trained to have a more complex strategy to choose $\boldsymbol{k}_a$ according to the environmental cues.

3. Identify agent in RL framework

State space $\mathcal{S}$

The environment and the dynamics are set continuous but the agent can perceive only a course representation of their current state (swimming direction and flow). The state space is discrete:

$$\mathcal{S} = \mathcal{S}_\omega \times \mathcal{S}_p \text{ where } \mathcal{S}_\omega = \{\omega_-, \omega_0, \omega_+\} \text{ and } \mathcal{S}_p = \{\uparrow, \downarrow, \rightarrow, \leftarrow\}$$

and in detail we define the course-gained states:

$$\mathcal{S}_\omega : \begin{cases} \omega_- \in [-1, -0.33) \\ \omega_0 \in [-0.33, 0.33] \\ \omega_+ \in (0.33, 1] \end{cases} \qquad \mathcal{S}_p : \begin{cases} \uparrow & \pi/4 < \theta(p) < 3\pi/4 \\ \leftarrow & 3\pi/4 < \theta(p) < 5\pi/4 \\ \downarrow & 5\pi/4 < \theta(p) < 7\pi/4 \\ \rightarrow & 7\pi/4 < \theta(p) < 9\pi/4 \end{cases} \text{ where } \theta(p) = \arctan(p_z/p_x)$$

Action space $\mathcal{A}$

The set of actions comprises four preferred swimming directions,

$$\boldsymbol{k}_a \in \mathcal{A} = \{\uparrow, \downarrow, \rightarrow, \leftarrow\}$$

Reward set $\mathcal{R}$

The reward is given by the increase in altitude:

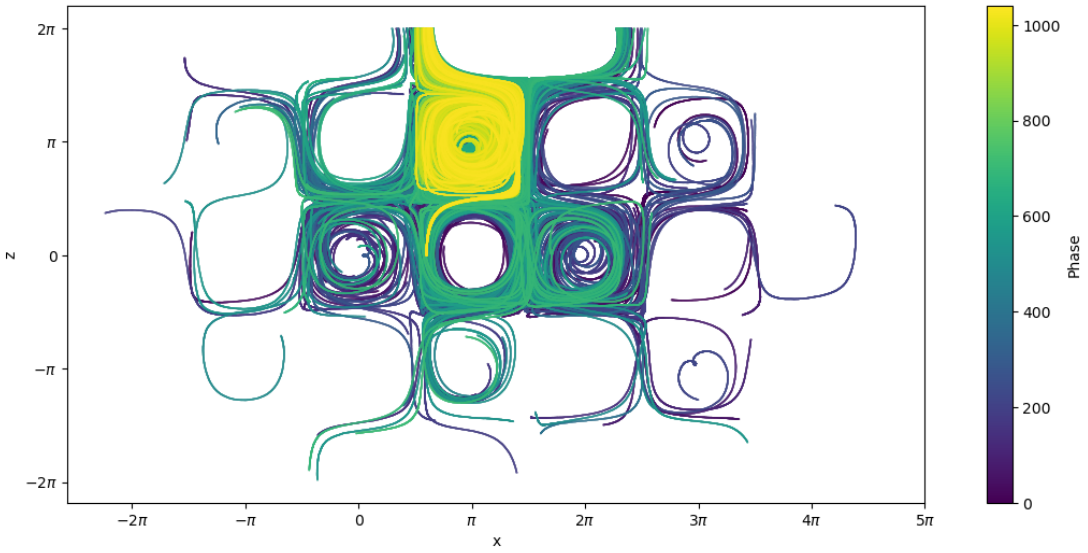
$$\text{for } s_n \rightarrow s_{n+1} \text{ the reward } r_{n+1} = z(s_{n+1}) - z(s_n)$$

where $z(s_n)$ and $z(s_{n+1})$ are z coordinates of the particle in state s_n and s_{n+1}.

4. Training and visualization

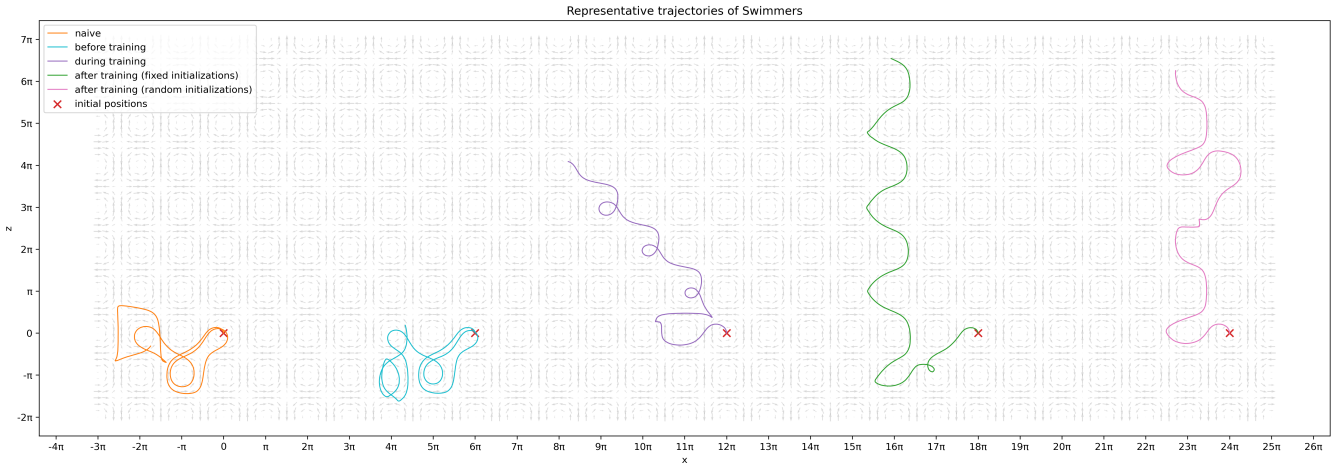
Particle motion is characterized by two dimensionless parameters Φ and Ψ. The parameters chose here are $\Phi = 0.3$, $\Psi = 1$, $B = 1$, $u_0 = 1$ and time step $\mathrm{d}t = 0.01$. The amplitudes of transitional noise and rotational noise are fixed to $D_0 = 0.001$, $D_R = 0.0001$.

A figure of trajectories illustrating training process



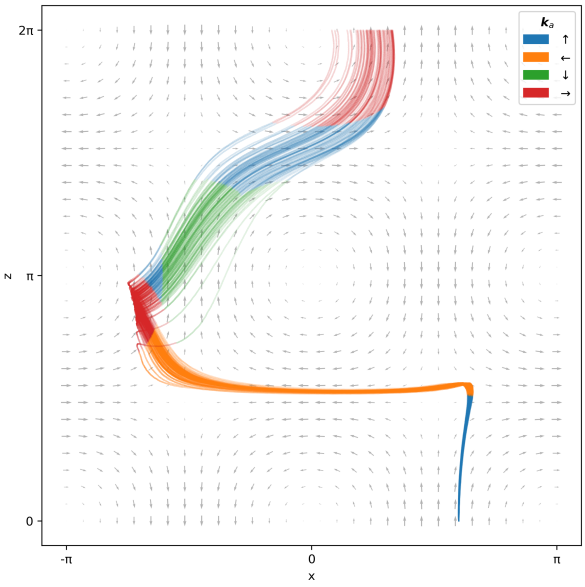
Fixed start position, $\Phi = 0.1$, $\Psi = 0.3$, max_episode_length = 5000, 1000 episodes. The colors of trajectories represent the episode number the agent was in. The figure shows that the agent became less exploratory with an upward trend in the training process. This training process is not ideal, as it can be seen from the trajectories that the agent has not experienced enough states (covering the entire space). However, its training trajectories is representative of training under fixed initial states, and they can give us a scenario of the process of convergence.

Representative Trajectories of Swimmers

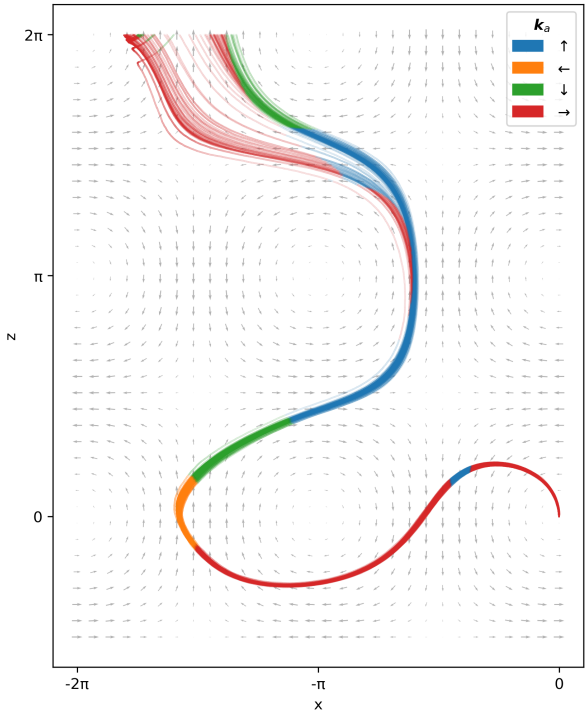


A set of representative trajectories at different stages in the learning process for smart particles compared with typical trajectories for naive particles confined in the flow. Learning is done with discount factor $\gamma = 0.996$, length of an episode $N_s = 10000$ steps, learning rate α decaying from 0.1 and exploration rate ϵ decaying from 0.2. The agent with random initializations is trained for 3000 episodes in one session, and the agent with fixed initializations is trained for 2000 episodes in one session.

Plotting the optimal actions for the smart particle that trained to exploit the flow to move upwards, we observe that the strategy contains some seemingly ineffective choices in the short run, such as pointing downwards and the benefit of these actions can be appreciated only over a long horizon.



An example of the optimal actions for the smart particle that succeeded to escape the confinement. (Initially $x = 0.6\pi$, $z = 0$, $\theta_p = \pi/2$)



An example of the optimal actions for the smart particle that succeeded to escape the confinement. (Initially $x = 0$, $z = 0$, $\theta_p = \pi/2$)