HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Reinforcement learning für intelligente Brownsche Dynamik

Haonan Xu
2583088

Hao Sun
2705866

24-05-2023

1 Introduction

Machine learning is a subset of artificial intelligence that involves the development of algorithms and statistical models that enable machines to learn from data and improve their performance over time without being explicitly programmed.

Reinforcement learning is a field of machine learning that emphasizes how to take actions based on the environment to achieve maximum benefits. In machine learning problems, the environment is abstracted as a Markov decision process. Reinforcement learning algorithms can learn the optimal strategy by trying different methods and receiving feedback from the environment. This algorithm does not rely on the precise mathematical model of the Markov decision process. Therefore, it is more suitable for complex and large-scale problems and is not limited by traditional dynamic programming methods.

Q-learning is a method of reinforcement learning. Its main approach is to continuously update the Q-table based on the environment and reward function during the learning process. By using the Q-table, the algorithm can find the optimal action in the current state to maximize the expected reward and achieve the best outcome throughout the entire process.

2 Results and Analysis

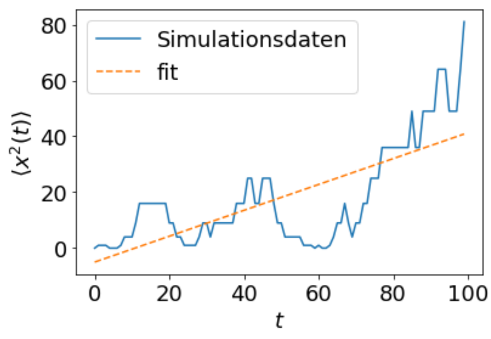
In this experiment, we followed the instructions in the Einleitung and learned the Q-learning algorithm step by step. In each task, we obtained meaningful images or data through Python programming for analysis.

The objective of this experiment is to explore the problem of finding the optimal path in a one-dimensional environment with a certain probability of random walk and interference.

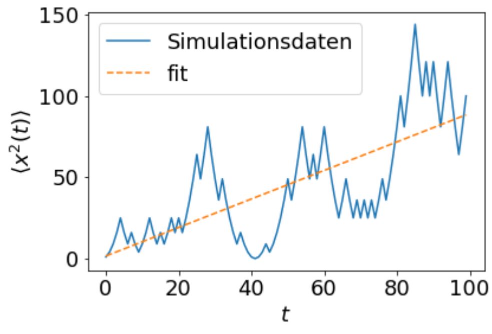
We will use Python in the following five tasks to understand each part that needs to be explored in this experiment.

2.1 Aufgabe5.1

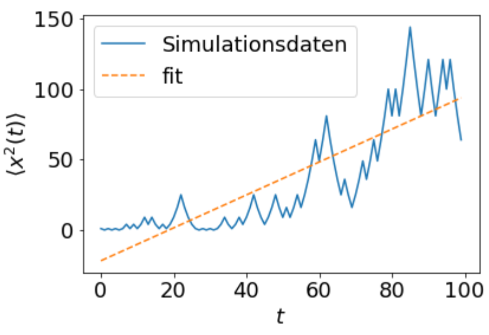
Task 5.1 mainly addresses the part of random walk. We try to find the relationship between the diffusion coefficient and the mean square displacement in a random walk. Usually, the step size of random walk in unit time is different. However, in this experiment, we use the same step size of a random walk, but with a certain probability of random walk behavior, to achieve the same situation. The probability of a random walk depends on the diffusion constant D.



((a)) D=0.2



((b)) D=0.5



((c)) D=0.9

Figure 1: MSD

$$D := \frac{a^2}{2\tau} \tag{1}$$

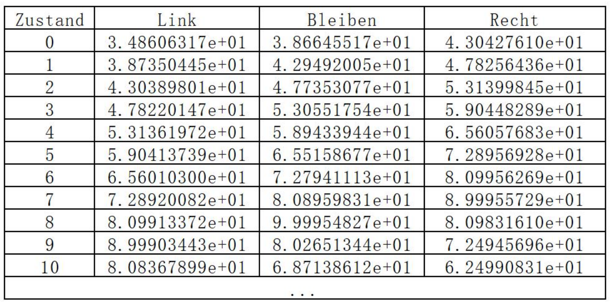
$$MSD = \frac{1}{N}\sum_{N}^{i}\left|x_i(t_0+t) - x_{i(t_0)}\right|^2 \tag{2}$$

Based on formulas (1) and (2), we can infer that the slope of $\langle x^2(t)\rangle$ and the diffusion coefficient has a double relationship. However, in the experiment, since the behavior of the random walk is random, it is not always possible to produce a good image with a sample size of 20,000. Therefore, we ran the simulation multiple times to find three sets of images with different diffusion coefficients. The diffusion coefficient of Figure 1(a) is 0.2, with a slope of 0.462. The diffusion coefficient of Figure 1(b) is 0.5, with a slope of 0.896. The diffusion coefficient of Figure 1(c) is 0.9, with a slope of 0.839. We believe that the deviation between the simulated slope and the diffusion coefficient is that the sample size cannot satisfy the law of large numbers.

2.2 Aufgabe5.2

Task 5.2 mainly constructed the environment model of Q-learning, which is similar to giving the agent a fixed initial position and a target position in a circular track (because the boundaries at the ends of a one-dimensional straight line are continuous), and using accumulated rewards to teach the agent how to find the target. The learning process is the process of updating the Q table, and by constantly updating the Q table, the agent can make it more accurate.

In the experiment, we used an initial position of 55 and a target position of 8. The purpose was to verify that the boundary conditions were working in the code. Theoretically, in the output gif, the agent should move continuously to the right until it reaches the target position and stops. After simulating with Python, we found that because of the code that includes random steps, in the last simulation, the agent also has a certain probability of moving to the left, which we consider normal. The purpose is to give the agent the possibility to choose a non-optimal action under the current learning, which can provide a new possibility and discover a better strategy. If the subsequent learning is not considered, the last Q-table can be used as a reference for behavior, and the random step program can be canceled, which is consistent with the theoretical continuous movement to the right.



Zustand	Link	Bleiben	Recht
0	3.48606317e+01	3.86645517e+01	4.30427610e+01
1	3.87350445e+01	4.29492005e+01	4.78256436e+01
2	4.30389801e+01	4.77353077e+01	5.31399845e+01
3	4.78220147e+01	5.30551754e+01	5.90448289e+01
4	5.31361972e+01	5.89433944e+01	6.56057683e+01
5	5.90413739e+01	6.55158677e+01	7.28956928e+01
6	6.56010300e+01	7.27941113e+01	8.09956269e+01
7	7.28920082e+01	8.08959831e+01	8.99955729e+01
8	8.09913372e+01	9.99954827e+01	8.09831610e+01
9	8.99903443e+01	8.02651344e+01	7.24945696e+01
10	8.08367899e+01	6.87138612e+01	6.24990831e+01
...			

Figure 2: Q-Table

From the table in image 2, we can see that the maximum Q-value for the action of moving to the right is on the positions from 0 to 7 to the left of the target position. The maximum Q-value for staying still at the target position 8 is obtained. For the positions from 8 to 10 to the right of the target position, the maximum Q-value for the action of moving to the left is obtained. This indicates that the Q-table is effective.

2.3 Aufgabe5.3

2.3.1 Aufgabe5.3.1

Aufgabe5.3.1.1 Welche Form hat Q in diesem Modell?

In this model, the Q is a matrix with 4 rows (for the 4 states) and 2 columns (for the 2 possible actions in each state), i.e

$$\left|\begin{array}{lcc} & action1 & action2 \\ state1 & Q(0,0) & Q(0,1) \\ state2 & Q(1,0) & Q(1,1) \\ state3 & Q(2,0) & Q(2,1) \\ state4 & Q(3,0) & Q(3,1) \end{array}\right|$$

Aufgabe5.3.1.2 Zu Beginn des Lernalgorithmus ist $\epsilon \approx 1$. Für diese Aufgabe ist der genaue Wert unerheblich. Berechnen Sie Q von Hand nach dem Ende der erste Epoche mit einer beliebigen Aktionsfolge in Abhängigkeit von R, γ, α.

The formula for Q is:

$$\mathcal{Q}_{ij}^{\text{neu}} = \mathcal{Q}_{ij}^{\text{alt}} + \alpha \left(\mathcal{R} + \gamma \max_k (\mathcal{Q}_{i'k}) - \mathcal{Q}_{ij}^{\text{alt}} \right) \quad (3)$$

We assume that the first according to the sequence of actions is $\leftarrow\rightarrow\rightarrow\rightarrow\rightarrow$.The value of Q in a epoche depends on R, γ, α.

1.$Q(0,0) = Q(0,0) + \alpha\left(\mathcal{R} + \gamma\left(Q(0,0)\right) - Q(0,0)\right)$

2.$Q(0,0) = Q(0,0) + \alpha\left(\mathcal{R} + \gamma\left(Q(0,1)\right) - Q(0,0)\right)$

3.$Q(0,1) = Q(0,1) + \alpha\left(\mathcal{R} + \gamma\left(Q(1,1)\right) - Q(0,1)\right)$

4.$Q(1,1) = Q(1,1) + \alpha\left(\mathcal{R} + \gamma\left(Q(2,1)\right) - Q(1,1)\right)$

5.$Q(2,1) = Q(2,1) + \alpha\left(\mathcal{R} + \gamma\left(Q(3,1)\right) - Q(2,1)\right)$

Aufgabe5.3.1.3 Berechnen Sie Q von Hand nach der zweiten Epoche nach der Aktionsfolge $\rightarrow\leftarrow\rightarrow\rightarrow\rightarrow$ in Abhängigkeit von R,γ,α.

Similar to the previous question:

1.$Q(0,1) = Q(0,1) + \alpha\left(\mathcal{R} + \gamma\left(Q(1,0)\right) - Q(0,1)\right)$

2.$Q(1,0) = Q(1,0) + \alpha\left(\mathcal{R} + \gamma\left(Q(0,1)\right) - Q(1,0)\right)$

3.$Q(1,0) = Q(1,0) + \alpha\left(\mathcal{R} + \gamma\left(Q(1,1)\right) - Q(1,0)\right)$

4.$Q(1,1) = Q(1,1) + \alpha\left(\mathcal{R} + \gamma\left(Q(2,1)\right) - Q(1,1)\right)$

5.$Q(2,1) = Q(2,1) + \alpha\left(\mathcal{R} + \gamma\left(Q(3,1)\right) - Q(2,1)\right)$

Aufgabe5.3.1.4 Berechnen Sie Q von Hand während der dritten Epoche nach der einer Aktion $\rightarrow$ in Abhängigkeit von R, γ,ϵ.
During the third epoch after an action $\rightarrow$,the Q function is:

$$Q(0,1) = Q(0,1) + \alpha\left(\mathcal{R} + \gamma \max\left(Q(1,x)\right) - Q(0,1)\right)$$

Here we choose random actions with probability ϵ.It means that we choose Q(1,0) or Q(1,1) with probability ϵ.

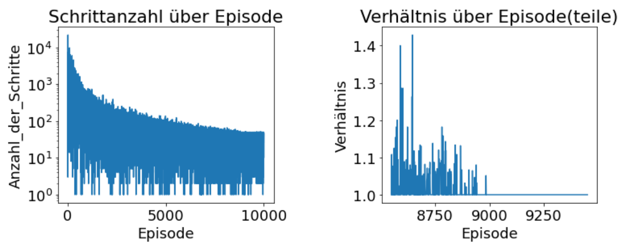
Aufgabe5.3.1.5 Wie erklären Sie, dass der Agent nach dem Lernprozess schon in Zustand 0 von dem Ziel in Zustand 3 gelernt hat? Welche Aktion wird der Agent für diesen Zustand gelernt haben?
The agent has learned that state 3 is the goal and that action 1 (go right) brings the agent closer to the goal. This is because the Q table contains the highest Q value for action 1 in state 0. The agent has learned that performing action 1 can move from state 0 to state 3.

Aufgabe5.3.1.6 Kann der Agent in diesem Modell lernen in einer Aktion nach links zu laufen?
The agent cannot learn to walk left in this model, because there is no reward for this step.Therefore the Q value cannot be increased

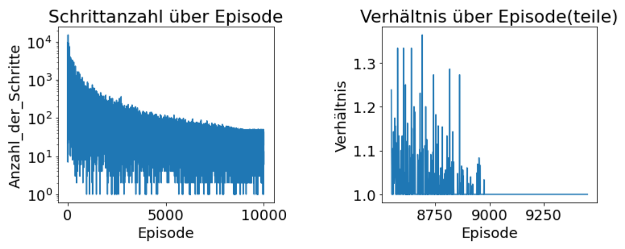
2.3.2 Aufgabe5.3.2

In task 5.3.2, we attempted to find the impact of the learning rate α on the Q-learning process and its outcomes. Through equation (3), we know that α affects the weight of the original Q-value and the new Q-value when updating Q-values. For a relatively high learning rate α , the learning process is faster but unstable, and the result may not converge. For a low learning rate α , the learning speed is slow, and it consumes more computation, but the result is more likely to converge. We conducted simulations using Python for three different learning rates α .

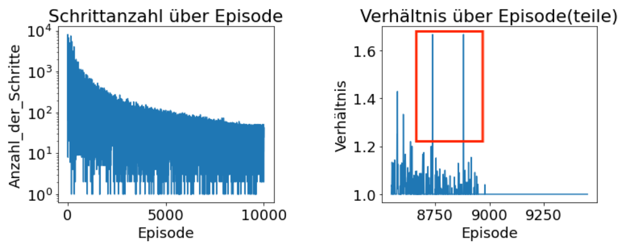
By comparison, we found that when the Learning rate approaches 1, the number of steps required to complete a process in the



((a)) α=0.000001



((b)) α=0.25



((c)) α=0.999999

Figure 3: Learning rate α

period close to ϵ equals one is more than 0.4 times the minimum required steps. We believe that this is due to the interference with the update of the Q table, but it may also be because too many random walks were selected instead of following the Q table during the process due to randomness. After the period where ϵ equals one, its performance is very stable and requires the minimum number of steps. For the overall learning process, the required steps decrease exponentially, and the learning speed increases exponentially. The learning rate is positively correlated with the learning speed, which can be seen through the comparison of three images.

2.4 Aufgabe5.4

In task 5.4, we studied the case with interference. We introduced a condition where there is a certain probability of preventing the agent from moving to the right, or forcing it to move to the left in a certain interval. We observed the behavior of the model under these conditions. In the task, we first adjusted the code to determine that the transition point is roughly between 0.70 and 0.85, and divided this interval into 40 parts, resulting in a generated image. We found that the behavior in the in-

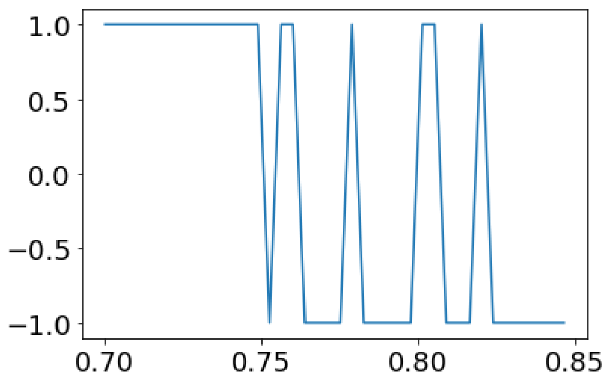


Figure 4: Looking for $P_{(0)}$

terval from 0.74 to 0.83 was oscillating rather than being in a stable state. Therefore, we believe that there exists a point $P_{(0)}$ within the interval from 0.74 to 0.83.

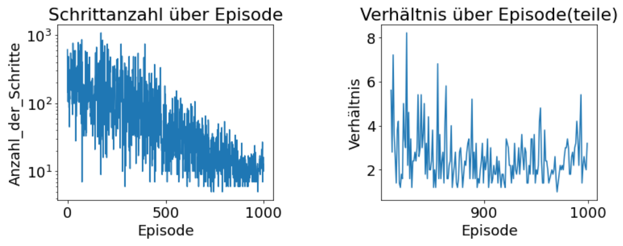


Figure 5: α=0.999999 with barrier zone

We changed the value of Learning rate and adjusted the obstacle blocking probability to 0.7. We observed a highly unstable convergence result in the simulation, as shown in the figure(5). Even when ϵ value is 1, the required number of steps still appears highly unstable, with oscillations ranging from one to five times the minimum required steps. There are two reasons for this situation: first, Learning rate close to 1 is highly susceptible to environmental interference factors. Second, we added additional environmental interference factors, namely our interference interval, which magnified the effect of interference. This is consistent with our expectations and indirectly verifies part of Task 5.3, where a high Learning rate is highly susceptible to interference and is not easy to converge.

3 Discussion

In this experiment, we attempted to solve a one-dimensional pathfinding problem. We went from the basic random walk to the pathfinding problem and then added interference to understand the basic operations of Q-learning. We learned about the four important parameters in Q-learning, namely R, alpha, gamma, and epsilon. R represents the reward and provides the basis for quantifying the Q-table. Alpha can change the efficiency of learning, gamma affects the focus on recent rewards versus long-term rewards, and epsilon can affect the proportion of random steps taken to explore new behaviors.

Regarding the learning rate alpha, it is a particularly important parameter in Q-learning. We found that in this experiment, changing the learning rate can not only affect the speed of learning but also the stability of learning. The relationship between the two is relative, a faster learning rate means lower stability. A lower learning rate means that more computing power is required, but it has higher stability, is easier to converge, and is less affected by the environment. For real-world problems, the impact of the environment may be uncertain and cannot be quantified. This means that in future practical applications, it is worth continuing to research and think about how to provide better learning rates based on the corresponding environment

In general, our research findings indicate that the learning rate strategy has a significant impact on Q-learning and needs to be adjusted appropriately. In addition, when facing complex environments, we also need to pay attention to the adjustment and optimization of the model to achieve better results.