4890-1066 GCL情報理工学特別講義VII(強化学習)

Homework assignment

April，2025

Lecturer: Takayuki Osa

|  |
| --- |
| Student ID: 37-236969 Name: Yi Wang |
| Execute the program “DQN.ipynb” (For example, you can use Google colaboratory) and answer the following questions.   * Please submit as a PDF file. * The filename should be “(student\_id)\_(Given\_name)\_(Family\_name)\_DQN.pdf”   e.g., “482222\_Takayuki\_Osa\_DQN.pdf”.  Deadline: May 20, 2025 |
| Q1. Find the part for computing the maximum of the Q-value given by    Please copy and paste the corresponding line of the codes.  Answer:  target\_Q\_max, \_ = torch.max(target\_Q, dim=1, keepdim=True)  Q2. Find the part for computing the target value given by    Please copy and paste the correspondine line of the codes.  Answer:  target\_Q = reward + not\_done \* self.discount \* target\_Q\_max  Q3. Find the part for computing the following loss function given by    Please copy and paste the corresponding line of the codes.  Answer:  critic\_loss = torch.nn.functional.mse\_loss(current\_Q\_chosen, target\_Q)  Q4. Run the program using the different hyperparameters, e.g., using a different number of units in each layer or different learning rates. Test **at least three** different hyperpraemters. Describe the hyperparameters you tested and show the learning curve for each setting. Discuss how the performance was influenced by the hyperparameters.  Answer:  The setting is Python 3.11, NVIDIA GeForce RTX 3090, and Ubuntu 22.04.  Besides, I did some changes. The code is shown in:  <https://github.com/yiwangyw/rl_course>  First, I set the device and move the model and tensors to device.  And my gym and torch version only support the datatype of original int so I add some codes like “int(action\_test)”.  In the end, I fix some random seeds for comparation of performance for different hyperparameters, but I am not sure if it is okay.  In train function, I add “env.reset(seed= trail\_seed + epi\_cnt)” for environment setting and do the same thing for test environment “env\_test.reset(seed= trail\_seed + test\_n)”.  For each trial, I use different but certain seeds “trail\_seed = int(args['random\_seed']) + ite”, i.e., 1234 + trail number.  Expl-eps 0.1 **discount 0.99** tau 0.005 start timesteps 1e4    Expl-eps 0.1 **discount 0.95** tau 0.005 start timesteps 1e4    Expl-eps 0.1 **discount 0.90** tau 0.005 start timesteps 1e4    Expl-eps 0.1 **discount 0.70** tau 0.005 start timesteps 1e4    Expl-eps 0.1 **discount 0.50** tau 0.005 start timesteps 1e4    As the discount parameter progressively decreases from 0.99 to 0.95, 0.9, 0.7, and 0.5, the algorithm’s performance gradually deteriorates and exhibits increased instability.  When the discount factor is relatively high likes 0.99 or 0.95, the model prioritizes long-term rewards, demonstrating higher final returns and more stable training curves.  However, as the discount factor diminishes, e.g., to 0.7 and 0.5, the model becomes increasingly myopic, focusing primarily on immediate and short-term rewards. It results in a significant decline in the policy’s long-term performance, accompanied by greater fluctuations in the training curve and difficulty in achieving stable convergence to high-return levels.  Therefore, higher discount factors facilitate the acquisition of long-term optimal policies, whereas lower discount factors may render the algorithm excessively short-sighted, thereby compromising its overall performance.  **Expl-eps 0.01** discount 0.99 tau 0.005 start timesteps 1e4    **Expl-eps 0.05** discount 0.99 tau 0.005 start timesteps 1e4    **Expl-eps 0.1** discount 0.99 tau 0.005 start timesteps 1e4    Expl-eps 0.5 discount 0.99 tau 0.005 start timesteps 1e4    Expl-eps 0.8 discount 0.99 tau 0.005 start timesteps 1e4    Expl-eps 1.0 discount 0.99 tau 0.005 start timesteps 1e4    As the exploration epsilon parameter progressively increases from 0.01 to 0.05, 0.1, 0.5, 0.8, and 1.0, the algorithm’s stability exhibits a marked decline.  When Expl-eps maintains low values, the algorithm predominantly employs a deterministic greedy policy with minimal stochastic exploration, enabling rapid and stable convergence to higher reward levels while demonstrating robust policy performance. As Expl-eps increases to moderate levels likes 0.1, the algorithm continues to converge to satisfactory rewards, albeit with marginally reduced stability and the emergence of reward fluctuations. When Expl-eps is further elevated to higher values, the proportion of random exploration increases substantially, resulting in severe algorithmic volatility. It demonstrates that the exploration rate significantly influences the stability of algorithmic performance.  Expl-eps 0.1 discount 0.99 tau 0.005 **start timesteps 5e3**    Expl-eps 0.1 discount 0.99 tau 0.005 **start timesteps 1e4**    +  Expl-eps 0.1 discount 0.99 tau 0.005 **start timesteps 5e4**    As the start timesteps parameter progressively increases from 5e3 to 1e4 and subsequently to 5e4, the algorithm’s performance exhibits significant deterioration and increasing instability. When start timesteps maintains a relatively small value, the algorithm rapidly transitions into an effective learning phase, achieving high rewards expeditiously with stable convergence. When start timesteps increases to 1e4, the algorithm retains its capacity for effective learning, albeit with a slight delay, resulting in final rewards and stability.  However, when start timesteps is further elevated to the substantial value of 5e4, the algorithm becomes entirely incapable of effective learning, remaining perpetually in the random exploration phase, which occurs because larger start timesteps values keep longer random policies during the initial phase and severely impeding effective policy learning. Consequently, it is important to set the random exploration steps appropriately according to the specific requirements of the training task. |