When should agents explore?

Zabolotnyi Artem [1]

Abstract

Nowadays, reinforcement learning is a significant direction of machine learning and deep learning. The core idea is to use an agent that operates in the environment and has the primary goal of maximizing some reward. Training such an algorithm could be difficult due to the massive amount of possible actions. An agent should explore a variety of actions and, at the same time, choose more relevant ones. Trade between such actions is an exploration vs exploitation problem, one of the most important in RL. In this project, we focus on applying different strategies of choosing when to explore and when to exploit. The main goal of this project is to compare different strategies of exploration and exploitation and compare them using Atari games.

1. Problem statement

Reinforcement learning helps to solve problems when we have a complex environment and cannot collect and label all possible samples from it. Without knowing an optimal action policy, we can put an agent into this environment and define the reward function. Agent must discover the environment and actions to maximize the reward for the current action and the other situation and all possibles of action sequences. The agent will understand the environment after many different trials. Some could be non-optimal but help the agent get necessary environmental information. Agent develops some strategy for what to do in a different situation. Based on this experienced agent could find an optimal path through all possible actions to maximize reward.

One of the core challenges of reinforcement learning is to make the algorithm flexible in terms of trade between exploration and exploitation. (Kaelbling et al., 1996) To get a maximum reward, it must use actions that it tried before and has a positive influence. However, sometimes, there are actions that agents do not take into account in the past but using them potentially gets better rewards than others. Exploitation - is an action choosing using previous experience. When exploration is probably not the best action at this time, which probably leads to a better global solution, it is a very hard and still not solved problem of an optimal algorithm of the proper way of combining exploration and exploitation.

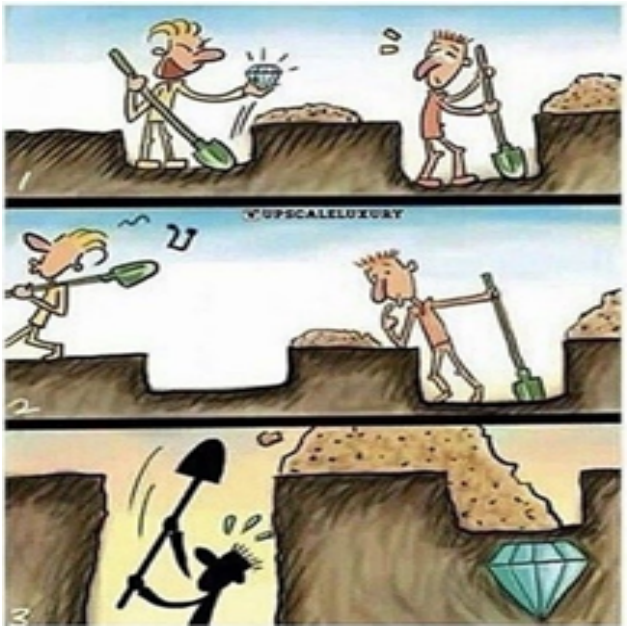

Figure 1. Exploration and exploitation. Make decision using only current reward could be not optimal in global case.

2. Strategies of exploration and exploitation

The main goal of RL is to maximize reward. To obtain good results algorithm should use diverse experiences to explore the different situations and sometimes use a policy which optimizes reward. Obtaining the best algorithm could be decomposed into two parts: granularity and switching mechanism.

[*]Equal contribution [1]Skolkovo Institute of Science and Technology, Moscow, Russia. Correspondence to: Zabolotnyi Artem <zaaabik@gmail.com>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

2.1. Granularity

Switching between modes could be done in several ways: **Step-level** - In each step of agent decides to follow policy or does not optimal step. The canonical example of such a method is ϵ-greedy. The core idea is to move within policy, but with the small probability, ϵ makes different actions.

Experiment-level The extreme case when behaviour during all processes of training agent is exploration without any policy. Learned policy using only for evaluation.

Episode-level exploration uses one strategy during the whole episode (training games versus tournament matches in a sport).

Intra-episodic method falls between step and episode. Exploration should contain several steps but should not be during the whole episode.

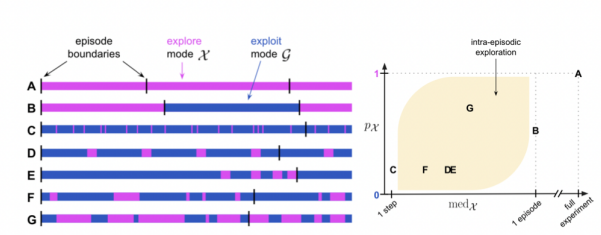

Figure 2. Illustration of different types of temporal structure for two-mode exploration. Left: Each line A-G depicts an excerpt of an experiment (black lines show episode boundaries, the experiment continues on the right), with the colour denoting the active mode (blue is exploit, magenta is explore). A is of experiment-level granularity, B episode-level, C step-level, and D-G are of intra-episodic exploration granularity. Right: The same examples, mapped onto a characteristic plot of summary statistics: overall exploratory proportion pX versus typical length of an exploratory period medX. The yellow shaded area highlights the intra-episodic part of space studied in this paper (some points are not realizable, e.g., when $pX \approx 1$, then medX must be large). C, D, E, F share the same $pX \approx 0.2$, while interleaving exploration modes in different ways. D and E share the same medX value, and differ only on whether exploration periods are spread out or happen toward the end of the episode.

2.2. Switching methods

The way to decide when to start to explore and when to stop could be done in several ways:

Blind switching - the easiest switching method does not use any information about the current state, and actions are done. It could be implemented with a counter (every 100 steps, start to explore with ten steps) or probabilistic when we have a probability of starting exploration mode for the parameterized amount of steps.

Informed switching following method uses the information of the agent to switch strategies. The agent returns two scalars; the first is *trigger* could be interpreted as uncertainty. When the trigger value is high, the agent will switch to explore mode. One of the ways to measure such value is 'value promise discrepancy'.

$$D_{promise}(t-k,t) := \|V(s_{t-k}) - \sum_{i=0}^{k-1} \gamma^i R_{t-i} - \gamma^k V(s_t)\| \tag{1}$$

where V(s) estimates value at sate s, R - reward value and γ discount factor.

2.3. Combining approaches

To improve the exploration process and prevent hyperparameter tuning author propose to add modification.

Bandit adaptation makes the algorithm more flexible using two parameters, the first, how often we can enter into exploration mode, and the second, how quickly we exit from it. Such parameters could be duration, probability, or target rate. Also, this optimization could be done by a meta-controller which maximizes episodic return.

Homeostasis value range of $D_promise$ could change over time due to training which improves accuracy. If we use the same threshold for all time, it will be not optimal. For this reason, an adaptive threshold algorithm was used. The core idea is to track the signal for switching during the training and adapt it to a specific average *target rate*. Such a trick helps to make the signal independent of the scale of a trigger signal.

2.4. Intra-episodic exploration

Such technique of combination methods is discussed in 2. There are several things we can change:

- Explore mode
- Explore duration
- Blind or informed trigger
- Exploit duration

Combining different choices in each category, we obtain a new algorithm. We will compare performance between each other and several baselines.

3. Q-learning

The core idea of this approach is to approximate the Q function in some step t. The function describes the final

reward if the agent makes action a in a state s and then follows the current policy.

$$Q_{t+1}(s,a) = E[R_{t+1} + \lambda max_{a'} Q_k(S_{t+1}, a')] \quad (2)$$

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(terminal\text{-}state, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Repeat (for each step of episode):
    Choose A from S using policy derived from Q (e.g., ϵ-greedy)
    Take action A, observe R, S'
    $Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma \max_a Q(S',a) - Q(S,A)\right]$
    $S \leftarrow S'$;
  until S is terminal

Figure 3. Q-learning: An off-policy TD control algorithm.

The loss function for deep model will look:

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s') \sim \mathrm{U}(D)} \left[\left(r + \gamma \max_{a'} Q(s',a';\theta_i^-) - Q(s,a;\theta_i) \right)^2 \right]$$

Figure 4. Q-learning: Loss function.

4. Related works

One of the most essential $\epsilon - greedy$ algorithms adopted for creating chunks of exploration actions where length is sampled from a distribution with heavy tails ((Dabney et al., 2020)). The paper (Bagot et al., 2020) proposes intrinsic reward pursuit invoked by the agent. Authors propose to learn such reward function through exploration options, i.e. additional temporally-extended actions to call separate policies.

5. Data

For our experiments, we use Atari Learning Environment (Bellemare et al., 2013) - one of the most significant benchmarks for the study exploration. Benchmark contains a large number of old atari games adapted for RL agents. For our experiments, we choose several games:

Pong - one of the most famous 2D game. The game is simplified version of tennis, where 2 players could move paddies up or down on the screen to hit the ball back and forth. Opponent player get points where you fail to return the ball back.

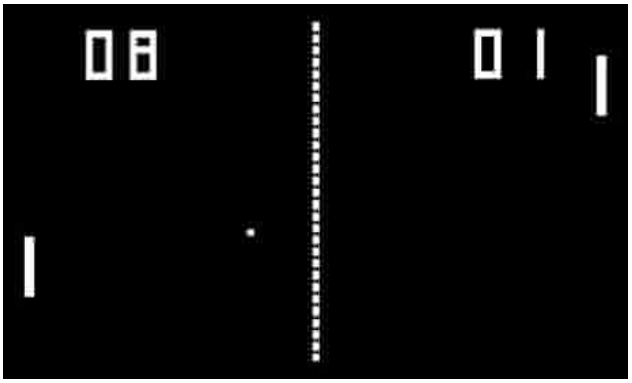

Figure 5. Pong gameplay

Breakout - classic game published by Atari in 1976. There are blocks in the upper part of the screen. The player should move the paddle to hit the ball against the bricks and eliminate them. The player has three lives, which decrease by one if the paddle misses the ball's rebound. The highest available score is 896. There are two possible actions: move left or move right.

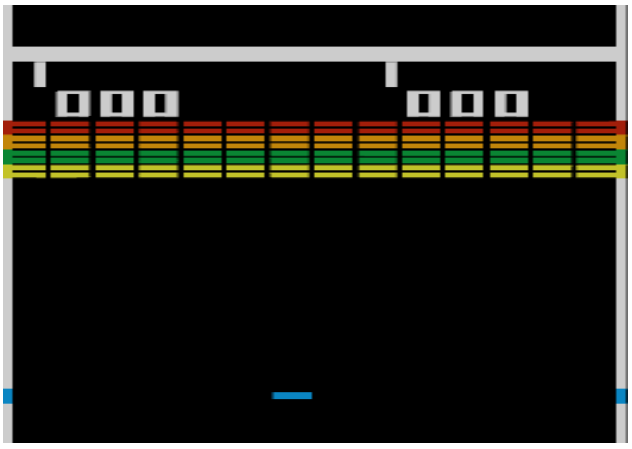

Figure 6. Breakout gameplay

Beamrider - another very popular Atari game. You should fly on the starship equipped gun with a limited amount of shots. Enemy ships appeared in front of the players, and you can destroy them or dodge them. There are nine possible actions like move left/right, shot etc.

Figure 7. Beamrider gameplay

6. Experiments

The project's primary goal is to compare different strategies for exploring and exploiting. For experiments, we use the Atari RL environment, agent-based on DQN model (Mnih et al., 2015).

For this project, we use four methods for blind switching strategies.

1. Pure explore mode ($p_x = 1 = \epsilon$)
2. Pure exploit mode ($p_x = 0 = \epsilon$)
3. $\epsilon - greedy$ strategy with ($p_x = 0.01 = \epsilon$)
4. $\epsilon - greedy$ strategy with exponential decay

,where p_x probability of exploration action in each step.

Parameters: For all experiments we use the same parameters of environments.

1. Skip 3 frames
2. Stack 4 neighborhood frames to obtain single input
3. Resize input in 84x84

For exponent decay $\epsilon - greed$ we use start value one and multiplier 0.999985 with min possible value 0.02

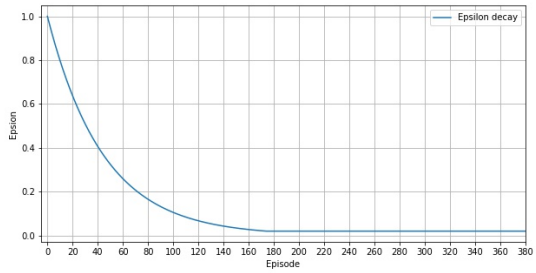

Figure 8. Epsilon value during training $\epsilon - greed$ decay model

6.1. Pong

The first experiment is related to the pong game as one of the simplest. The model can solve this problem. Game end if one of the players gets 21 points. Thus the worst result is -21, and the win of our algorithm is getting 21. As we can find in figure , our model is very close to obtaining a maximum reward in-game.

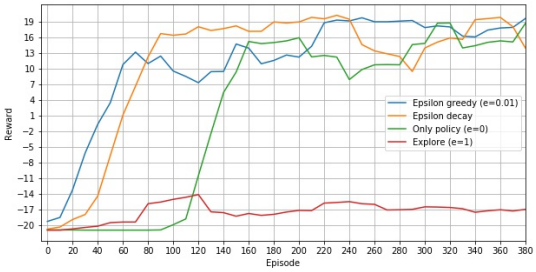

Figure 9. Pong. Mean reward value for different strategy

From figure 9, we can find that only explore mode does not work at all for the pong game. The reason for this phenomenon is not explored. Only the strategy mode shows lower performance because the highest reward is smaller than different methods, and to obtain the best quality model, spend more time. Epsilon greedy and epsilon greedy decay shows the closest results, while decay methods obtain the best reward in a shorter time.

6.2. Beamrider

Probably is the most challenging game in our experiments because of the nine possible actions and the huge variate of the actions in it. The min score equals 0, while the random score is near 360 points. Figure 10 shows mean rewards between 5 runs during the training of the model. A faster training process gets better results for only policy strategy,

while all other methods get close to each other in terms of training speed and final result.

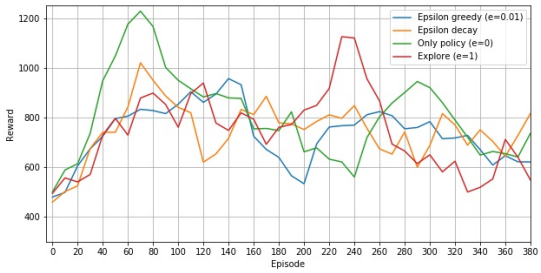

Figure 10. Beamrider. Mean reward value for different strategy

6.3. Breakout

The most unexpected results we observe for the breakout game. The game should be not so hard for humans because there are only two meaningful actions. The maximum available score here is 864, and the lower score is 0. Training algorithms take a massive amount of time for all strategies and the reward after one day of training is pretty low. All strategies have a similar dynamic during training and maximum obtained reward .

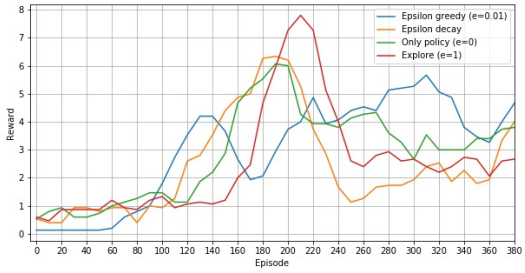

Figure 11. Breakout. Mean reward value for different strategy

After looking for a training procedure more precisely, we observe the situation when could the model be stuck. After players miss the ball, they need to press the 'Fire' button to continue play, while the model predicts different actions and can not continue the game. Due to this problem, the results collected from this game are not influential for our project.

7. Further work

This project compares several approaches for blind switching explore/exploit mode. This method works for simple games like pong. At the same time, this situation could dramatically change if we look into games with a more complicated environment and with a wide variety of actions. We focus our research in two directions. The first is to work with the breakout game and causes of learning stuck. Second, try informed switch methods on the same games.

8. Conclusion

In this project, we are doing research in the direction of exploration/exploitation methods in reinforcement learning using the Atari games environment. Making a model even for such a not complicated task took a lot of debugging time and also not less training time. For *pong* game, we find that only explore strategy does not work at all in case, while $e - greedy$ strategies show the best result. Also, the critical notion is that not for all games we get a satisfactory performance. *Breakout* game becomes hard for our model stuck in a situation where a player should press the special button after losing the life. Agent in *Beamrider* game shows the best results when we use the only-policy strategy of exploitation, and all other methods get approximate equal results.

References

Bagot, L., Mets, K., and Latré, S. Learning intrinsically motivated options to stimulate policy exploration. 2020.

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

Dabney, W., Ostrovski, G., and Barreto, A. Temporally-extended ϵ-greedy exploration. *arXiv preprint arXiv:2006.01782*, 2020.

Kaelbling, L. P., Littman, M. L., and Moore, A. W. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *nature*, 518(7540): 529–533, 2015.