MSc in Intelligent CIS
ECE 805 - Machine Learning
Tutorial 6 - Reinforcement Learning
08 May 2020

Kleanthis Malialis
Marie Skłodowska Curie Widening Fellow
KIOS Research and Innovation Center of Excellence
University of Cyprus
malialis.kleanthis@ucy.ac.cy

Contents

1 Tabular solution methods

1.1 Exploration / exploitation dilemma

Try a new restaurant (explore) or go to your favourite restaurant (exploit).

- Too much exploitation is not desirable since it is more likely for an agent to stuck in a local maximum.
- Too much exploration is also not desirable since learning becomes difficult.

Greedy strategy: An agent selects actions only by exploiting its knowledge.

ϵ-greedy strategy: An agent behaves greedily most of the time but with a probability *epsilon* it selects an action randomly. This strategy can theoretically sample over time every action ensuring convergence to Q^*.

Decayed ϵ-greedy strategy: It is advised to reduce ϵ over time, therefore, exploration is initially encouraged but as the agent becomes more and more experienced exploitation is then encouraged (necessary condition for SARSA convergence).

Boltzmann strategy: another popular exploration strategy.

1.2 Initial conditions

Consider a reward function that generates rewards in the range $[a, b]$.

- optimistic initialisation $\rightarrow$ set the Q-values to b
- pessimistic initialisation $\rightarrow$ set the Q-values to a
- hybrid $\rightarrow$ set the Q-values randomly in $[a, b]$

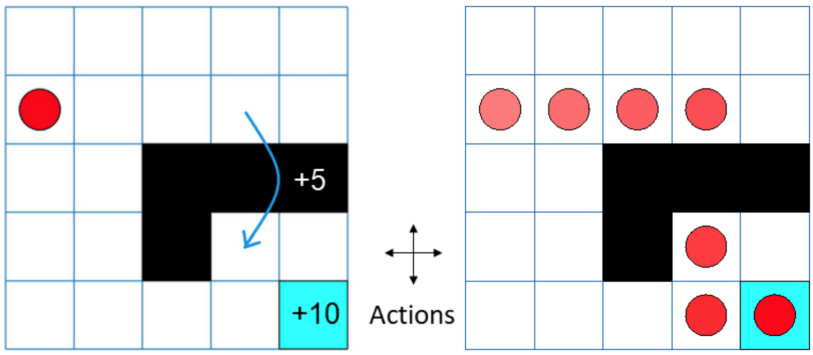
Optimistic initialisation allows all actions in all states to be tried out before convergence to the optimal policy occurs (slower). With pessimistic initialisation, an agent's behaviour can only converge to the optimal policy if the exploration-exploitation strategy identifies it (faster).

1.3 Q-learning (off-policy)

Algorithm 1 Q-learning

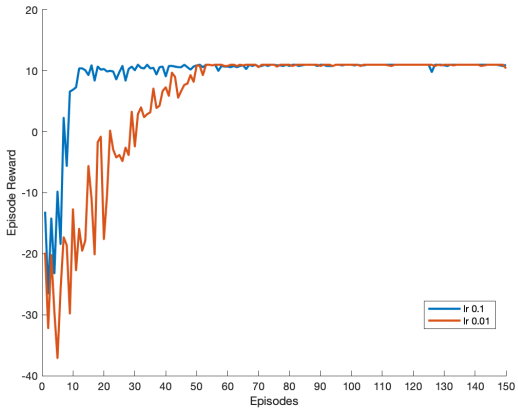
$\alpha \in (0, 1]$: learning rate
$\epsilon > 0$: exploration rate (small)
$\forall s \in S, a \in A(s)$ initialise $Q(s, a)$ arbitrarily except $Q(terminal, \cdot) = 0$
for each episode **do**
    initialise S
    repeat for each step of episode
        Choose A from S using policy derived from Q (e.g. ϵ-greedy)
        Take action A, observe R, S'
        $Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma \max_a Q(S', a) - Q(S, A)]$
        $S \leftarrow S'$
    until S is terminal

The relevant MATLAB code is found in the **qlearning.m** file. The problem domain is shown in Fig. 1 and the results are shown in Fig. 2

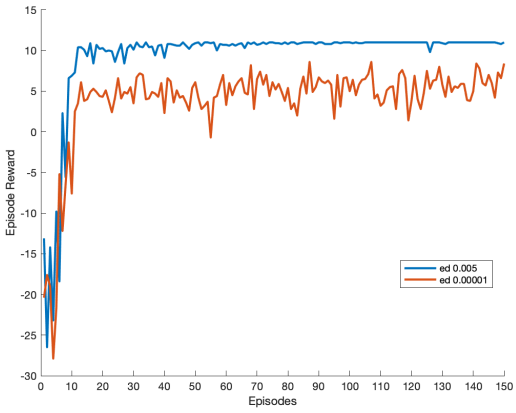


(a) environment (b) optimal policy

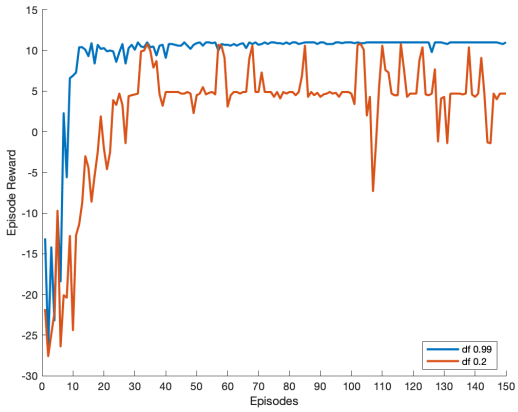
Figure 1: Grid world



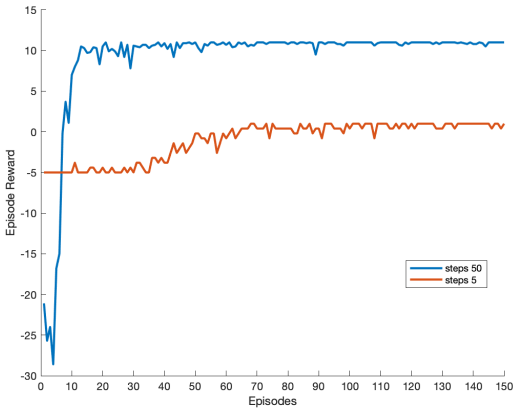
(a) learning rate



(b) exploration decay factor



(c) discount factor



(d) steps per episode

Figure 2: Role of hyper-parameters (Q-learning)

Algorithm 2 SARSA

$\alpha \in (0, 1]$: learning rate
$\epsilon > 0$: exploration rate (small)
$\forall s \in S, a \in A(s)$ initialise $Q(s, a)$ arbitrarily except $Q(terminal, \cdot) = 0$
for each episode **do**
    initialise S
    Choose A from S using policy derived from Q (e.g. ϵ-greedy)
    repeat for each step of episode
        Take action A, observe R, S'
        Choose A' from S' using policy derived from Q (e.g. ϵ-greedy)
        $Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma Q(S', A') - Q(S, A)]$
        $S \leftarrow S'$, $A \leftarrow A'$
    until S is terminal

1.4 SARSA (on-policy)

The relevant MATLAB code is found in the **sarsa.m** file. The problem domain is shown in Fig. 1 and the results are shown in Fig. 3.

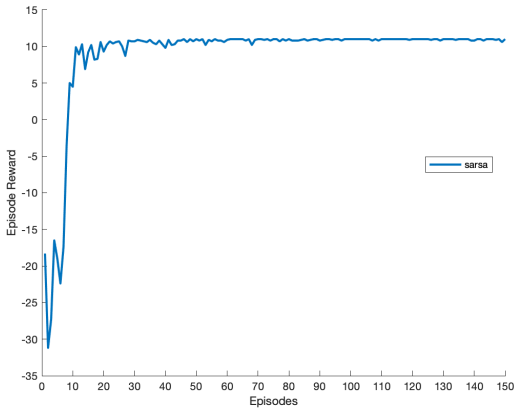


Figure 3: SARSA

1.5 Convergence criteria

Theoretical convergence:

- The environment's states satisfy the Markov property.
- An agent visits all state-action pairs infinitely often.
- The exploration rate reduces to zero (for SARSA).
- The learning rate reduces to zero.
- Bounded rewards.

2 Approximate solution methods

RL algorithms experience difficulties when scaling-up to complex real-world applications that involve large and continuous state and action spaces. Therefore, learning becomes infeasible due to memory constraints and computational time and a tabular implementation becomes very inefficient. Function approximation techniques approximate the search space.

2.1 Deep Q-Networks (DQN)

DQN: Human-level control through deep reinforcement learning (paper, video, blog)

Neural network architecture

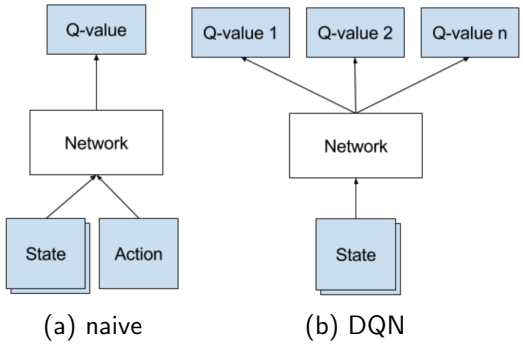


Figure 4: DQN's neural network architecture

Loss:

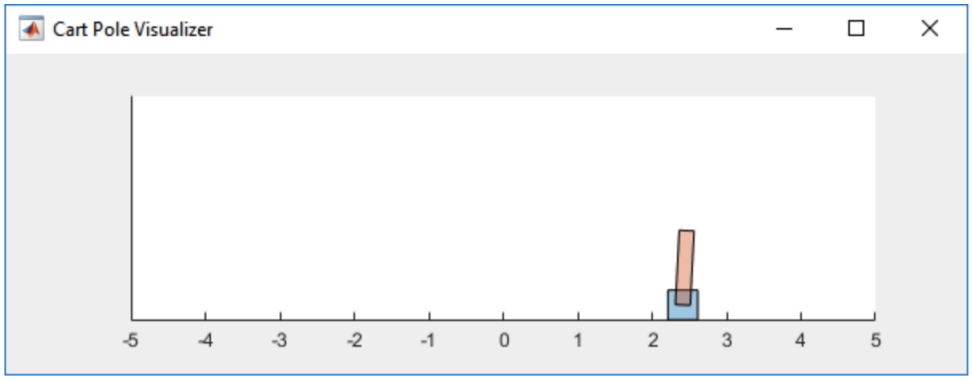
$$L = \frac{1}{2}[\underbrace{r + \gamma \max_a Q(s', a)}_{\text{target}} - \overbrace{Q(s, a)}^{\text{prediction}}]^2$$

Experience replay
During training all the experiences $< s, a, r, s' >$ are stored in a memory. When training the network, random samples (batches) from the replay memory are used instead of the most recent transition.

Importance: typically, successive states are highly similar, therefore, there is a high risk that the network will forget about what it's like to be in state it hasn't seen in a while. Experience replay breaks the similarity of subsequent training samples that may drive the network into a local minimum.

Target network
To address the "moving target" issue, we use a separate *target network* to estimate the target. The target network has the same architecture as the function approximator but with frozen parameters. The parameters from the *prediction network* are copied to the target network at every C iterations.

Simulations
The relevant MATLAB code is found in the **dqn.m** file. The problem domain is shown in Fig. 5 and the results are shown in Fig. 6.



For this environment:

- The upward balanced pendulum position is `0` radians, and the downward hanging position is `pi` radians
- The pendulum starts upright with an initial angle of +/- 0.05 radians
- The force action signal from the agent to the environment is from -10 to 10 N
- The observations from the environment are the position and velocity of the cart, the pendulum angle, and its derivative
- The episode terminates if the pole is more than 12 degrees from vertical, or the cart moves more than 2.4 m from the original position
- A reward of +1 is provided for every time step that the pole remains upright. A penalty of -5 is applied when the pendulum falls.

Figure 5: Cart-Pole

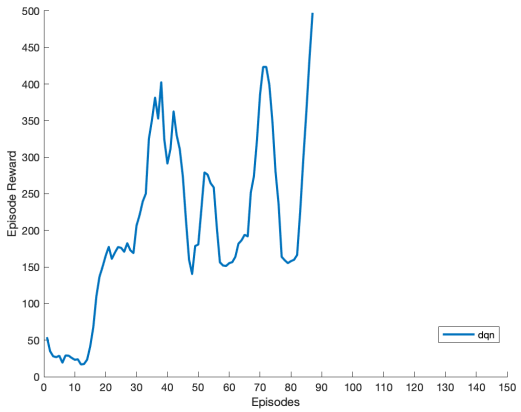


Figure 6: DQN

3 Current trends in RL

3.1 Data-efficiency / effective learning

Agent57: it outperforms the standard human benchmark on all 57 Atari games (paper)

3.2 Knowledge-based reinforcement learning

Model-based RL
Model may be known or can be learnt.

- AlphaGo: the strongest Go "player" in history (paper, video)

Reward shaping
It refers to the process of training an agent using an artificial reward rather than the reward naturally received by the environment, in order to favour or bias learning towards specific behaviours. Reward shaping constitutes a common way to incorporate domain or expert knowledge.

Imitation learning / learning from demonstration
An agent then tries to learn the optimal policy by following or imitating the expert's decisions.

- Autonomous helicopter: project website
- ViBe: Learn models of behaviour from unlabelled raw video data of a traffic scene collected from a single monocular camera with ordinary resolution. The approach calibrates the camera, detects relevant objects, tracks them through time, and uses the resulting trajectories to perform LfD. (paper)

3.3 Multiagent reinforcement learning

MARL deals with the construction of a system involving multiple reinforcement learning agents, which are situated in a common environment that can perceive through sensors, and act upon it through actuators.

- **Curse of dimensionality**: It refers to the exponential growth of the search space, in terms of the number of states and actions; SARSA and Q-learning are most affected by this. In MARL, the complexity is also exponential in the number of agents in the system.
- **Partial observability**: An agent cannot perceive all of the state information due to data unavailability, sensor misinterpretation or hidden information. In MARL, each agent has only knowledge about its local environment. Also, the effects of an agent's action on the environment, also depend on the other agents' actions. The Markov property doesn't hold if an agent cannot observe the joint space.
- **Credit assignment**: *Temporal* credit assignment deals with finding which action in the history of actions is responsible for a particular reward. Typically in a (cooperative) multiagent system an agent is provided with a reward at the global / system level. An agent may be rewarded for taking a bad action, or punished for taking a good action. This is termed the *multiagent* credit assignment problem.

Hide and seek: Emergent tool use (paper, video)
AlphaStar: Grandmaster level in StarCraft II (paper, video)

3.4 Safe reinforcement learning

Safe Reinforcement Learning can be defined as the process of learning policies that maximize the expectation of the return in problems in which it is important to ensure reasonable system performance and/or respect safety constraints during the learning and/or deployment processes.

Survey: García, & Fernández, A Comprehensive Survey on Safe Reinforcement Learning, JMLR, 2015.