CentraleSupélec 2015-2016 Date : 30 novembre 2015
SIR

Reinforcement Learning
Bureau d'Études (Machine Learning)

Objectives

This *Bureau d'Études* aims at studying reinforcement learning algorithms. The first part consists in studying batch methods (fitted-Q and LSPI) on the (toy) "mountain car" problem. The second part consists in studying online methods (Q-learning and SARSA) on the (toy) "cliff walking" problem (notably the effect of exploration and the difference between on- and off-policy learning).

About episodic reinforcement learning

In some reinforcement learning problems, the task is episodic. Consider for example the maze problem, where a robot has to go out of a maze. When the robot reaches the exit, the task is ended. Yet, the framework studied in course considers infinite trajectories. An episodic problem can be formalized by the addition of an *absorbing state*. An absorbing state s_{ab} is such that the reward for taking any action in this state is nul ($r(s_{\text{ab}}, a) = 0$, for all $a \in \mathcal{A}$) and the state is self-transient (that is, $P(s_{\text{ab}}|s_{\text{ab}}, a) = 1$, for all $a \in \mathcal{A}$). In the maze example, when the robot reaches the exit, it indeed transits to an absorbing state.

1. What is the value of an absorbing state (for any policy)?

Given that the value of an absorbing state is analytically known, it is useless to learn it (and in practice it is useless to model the absorbing state, see next). When interacting with the environment, assume that the system provides also a boolean flag indicating if the episode is ended or not.

2. How can you modify easily any studied reinforcement learning algorithm to handle the end of episode (using this boolean flag)?

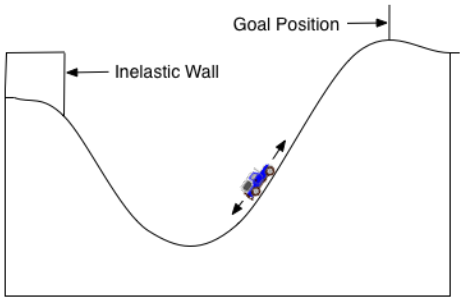

Figure 1: The mountain car problem (image taken from `http://library.rl-community.org/wiki/File:MountainCar-Envirornment.png`).

1 Batch learning

1.1 The mountain car problem

The task consists in driving an underpowered car up a steep mountain road, as shown in Fig. 1. The gravity being stronger than the car's engine, this requires first moving away from the goal, in order to build enough inertia to carry it up the steep slope. There are tree possible actions: full throttle forward ($+1$), full throttle reverse (-1) and zero throttle (0). The (continuous) state of the system is given by the position x_t and the velocity v_t of the car. The simplified dynamics are as follows:

$$v_{t+1} = \min\left(\max\left(v_t + 0.001a_t - 0.0025\cos(3x_t), -0.07\right), 0.07\right), \quad (1)$$

$$x_{t+1} = \min\left(\max\left(x_t + v_{t+1}, -1.2\right), 0.5\right). \quad (2)$$

Moreover, when x_{t+1} reaches the left bound, the velocity v_{t+1} is set to zero. The reward is -1 at each time step, $+10$ for reaching the top of the mountain, and -10 for hitting the inelastic wall.

3. Give the only stable state when no throttle is applied (that is, the car does not move).

1.2 Work to be done

You will apply fitted-Q (approximate value iteration with an ensemble of extremely randomized regression trees as the regressor) and LSPI (approximate policy iteration with LSTD as the value function estimator) to a set of transitions randomly sampled from this domain (in the dataset $\mathcal{D} = \{(s_i, a_r, r_i, s'_i)\}$, each state s_i is sampled uniformly, each action a_i is sampled uniformly, the state s'_i is sampled from the system dynamics and the reward r_i is as given in the previous section).

4. Plot a dataset (for a not too large number of transitions).

5. For each iteration of each algorithm:

   - plot the currently estimated value function as well as the estimated greedy policy (to see how things evolve),

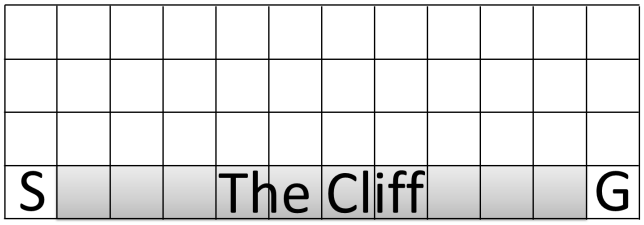

Figure 2: The cliff walking problem.

- quantify the quality of the learnt policy by evaluating the true value (using a Monte Carlo Rollout) of the equilibrium state discussed above (define a maximum number of step for each trajectory, a bad policy can get stuck indefinitely). You can also plot the related trajectory.

6. Explain the observed results.

Some hints:

- use a large value for the discount factor (say, $\gamma = 0.99$),
- for the ensemble, you can grow the trees until a small number of examples is in each leaf (say, 5), and take a large enough ensemble (say, 50),
- for fitted-Q, do a large enough number of iterations (say, 150),
- for LSPI, you can use the provided features (explain what they are),
- for LSPI, a (very) few iterations are sufficient (say, 5),
- for training, use a large enough dataset (say 30'000 transitions),
- you can also play on all these parameters and see (and try to explain) what happens.

2 Online learning

2.1 The cliff walking problem

The cliff walking problem is the gridworld illustrated in Fig. 2. Each square corresponds to a state, and there are four actions: go left, go right, go up and go down. Applying an action to a state will lead to the next state deterministically (for example, applying the action "right" will move the agent to the right square). If a movement leads to outside the environment, the corresponding action has no effect. All episodes begin in the start state ("S") and the goal is to reach the Goal state ("G"). If the agent moves to the cliff, the episode ends and the agent is punished with a reward of -100. If the agent moves to the goal, the episode ends and the agent receives a reward of 0. All other steps are rewarded -1.

7. Model (informally) this problem with an MDP (notably, what is the size of the state space?).

8. The state space being finite and small enough, a tabular representation for the Q-values can be envisioned. The corresponding feature vector $\phi(s, a)$ is zero everywhere, except for the index corresponding to the (s, a) couple. Rewrite the SARSA and Q-learning update rules by taking advantage of this.

2.2 Work to be done

Here, you will consider two algorithms for learning a good controller in an online setting, namely SARSA and Q-learning. Both algorithms will be combined with an ϵ-greedy strategy. Q-learning is called an *off-policy* algorithm, because it estimates the optimal Q-function while following any policy. SARSA is called an *on-policy* algorithm, because it estimates the Q-function of the policy being followed.

Apply Q-learning and SARSA to the cliff walking problem.

9. For each algorithm and each learning episode:
   - provide the discounted cumulative rewards gathered during the learning episode,
   - test the greedy policy on the problem (without learning!) and plot the corresponding trajectory.

10. Compare Q-learning and SARSA in terms of rewards gathered during learning as a function of episodes, comment.

11. Do the same thing for rewards gathered during testing, as a function of episodes, comment.

12. Do the same study without exploration at all ($\epsilon = 0$). How do you explain these (surprising) results?

Some hints:

- take a large discount factor (say, $\gamma = 0.99$),
- consider a large enough number of episodes (say, 2000),
- take $\epsilon = 0.1$, for example (can go to $\epsilon = 0.3$), keep it constant,
- take a constant learning rate (say, $\alpha = 0.1$),
- you can also play on all these parameters and see (and try to explain) what happens.