Pierre and Marie Curie University

Experiments

Learning from expert

Author:
Matthieu Zimmer

Supervisors:
Paolo Viappiani
Paul Weng

September 17, 2013
Version 1.1

Contents

1 Environnement

1.1 Mountain car

A standard testing domain in reinforcement learning, is a problem in which an underpowered car must drive up a steep hill. Since gravity is stronger than the car's engine, even at full throttle, the car cannot simply accelerate up the steep slope. The car is situated in a valley and must learn to leverage potential energy by driving up the opposite hill before the car is able to make it to the goal at the top of the rightmost hill, Sutton and Barto (1998) .

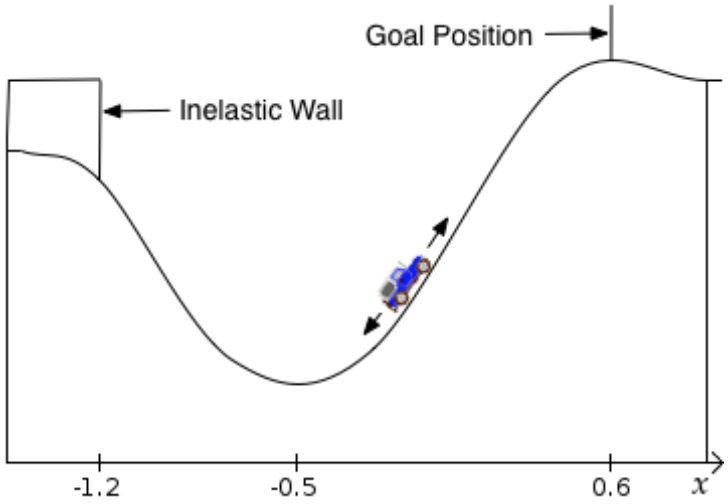

Figure 1: The Mountain Car Problem

State Variables Two dimensional continuous state space :
$(x, y) \in Position \times Velocity \; with \begin{cases} Position = [-1.2, 0.6] \\ Velocity = [-0.07, 0.07] \end{cases}$

Actions One dimensional discrete action space :
a ∈ Motor = {-1,0,+1} corresponding respectivly to left, neutral, right

Reward For every time step: -1, when goal is reached: 0

Transition $\begin{cases} y = y + a \times 0.001 + cos(3 \times x) \times -0.0025 \\ x = x + y \end{cases}$

Initial State $\begin{cases} x = -0.5 \\ y = 0.0 \end{cases}$

Termination Condition $x >= 0.6$

1.2 Open the Gate

The mountain car problem gives a -1 reward at every step, and the state allows to determine exactly the best action to take. So we could think that there is no state more important than other (to give feedbacks).

Here, we introduce an other problem where rewards will be different according to our state.

The agent have to reach every green boxes **according to the given order** to solve the problem.

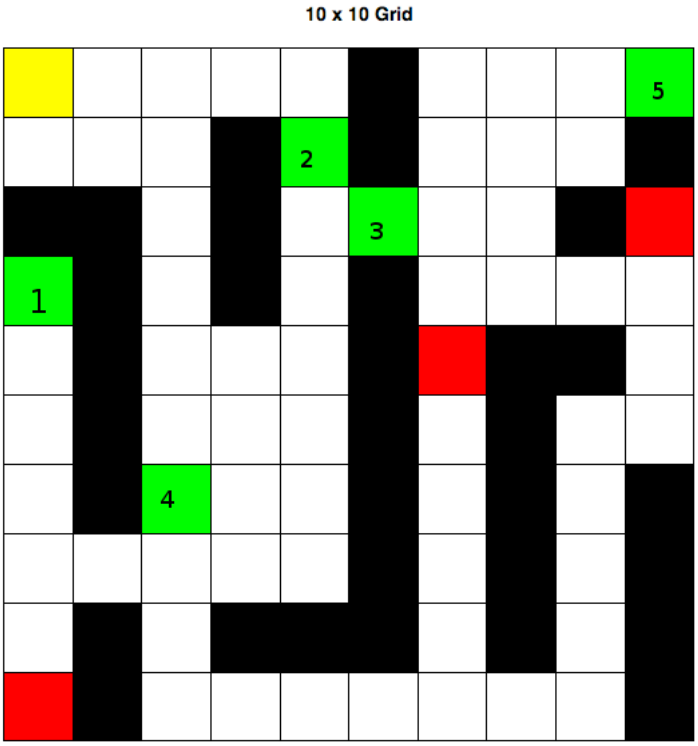

Figure 2: Grid World

State Variables Three dimensional discretized state space :
$(x, y, n) \in \{1, 10\} \times \{1, 10\} \times \{1, 5\}$
x and y are the coordinates of the grid
n determines which future case must be reached

Actions One dimensional discrete action space : the direction
$d \in \{left, up, right, bottom\}$

Reward white boxes : -1
green boxes : +20 if n = number_box(x, y) else -1
red boxes : -20

Transition x = x + 1 if d = right $\wedge\neg$ black_box(x + 1, y)
y = y + 1 if d = up $\wedge\neg$ black_box(x, y + 1)
x = x - 1 if d = left $\wedge\neg$ black_box(x - 1, y)
y = y - 1 if d = bottom $\wedge\neg$ black_box(x, y - 1)
n = n + 1 if green_box(x, y) $\wedge$ n = number_box(x, y)

Initial State (x,y,n) = (1, 1 ,1)

Termination Condition n > 5

2 Effects of advices

In this section, we are experimenting how the advice should affect the learner in reinforcement learning algorithms like Q-Learning or Sarsa. We are working with an approximation of the Q-function with a common linear function $Q(s,a) = \sum_i \theta_i \times f_i(s,a)$. The features f are binary and follow a tile coding, Sutton and Barto (1998). Then, it is necessary to perform a gradient descent to approximate $\vec{\theta}$.

Let's

2.1 Strategies

We always advise with a action. Here a is the recommended action, and s the current state.

Let us also introduce an other notation, the vector $\vec{\theta}(f(s,a))$, this represents the actived θ_i in the state s with the action a. It only make sense because the features f are binary. Thus, $Q(s,a)$ can be rewrite like $Q(s,a) = \sum_i \theta_i(f(s,a))$. We also asume that features f always depends of s and a.

Here is our different strategies :

Max In this strategy, we increase the Q value of the recommended action, with the best possible Q value in the current state, so that it is selected next time.

$$Q(s,a) = \max_{a'} Q(s,a') + \delta$$

With the approximation :

$$\vec{\theta}(f(s,a)) = \vec{\theta}(f(s,a'))$$

Decrease Instead of increasing the Q value of the recommanded action, we will decrease the Q-value of the others actions.

Informed Exploration This strategy can only be used in the case of advising before acting. Agent takes action a, and $Q(s,a)$ will be updating according with the next reward.

Fixed $\pi(s) = a$
$Q(s,a) = max(Q(s,\_) + \delta$

Others We could imagine an other strategy as Lucky Exploration only for the next time we encounter this state (not fixed for everytime).

2.2 Advice after acting

In this experiment, we have two agent, a teacher (QLearning) and a learner (QLearning). The teacher is favored to advice (it is done to compare advice effects) : +10 if advice else -10. With this reward function, it is not important to display the algorithm "advice before" because it will advice all the time.

The teacher advices the learner with the best action to take (according to the best policy previously computed). They have the same state representation.

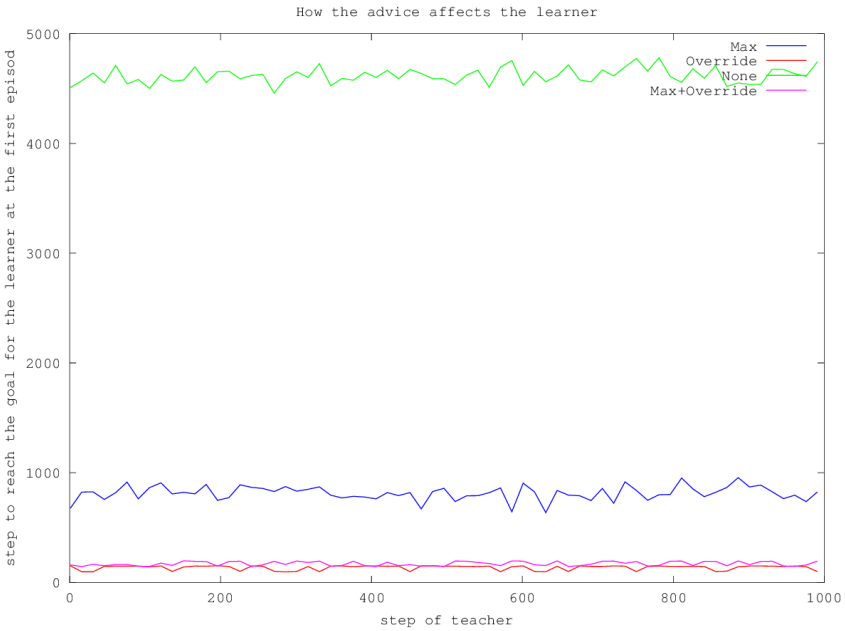

Figure 3: Teacher advices all the time. Number of step to reach to goal at the first episod

Mountain Car The two best strategies are Override and Max+Override. (Max+Override is a little quite better because it updates the Q value directly).

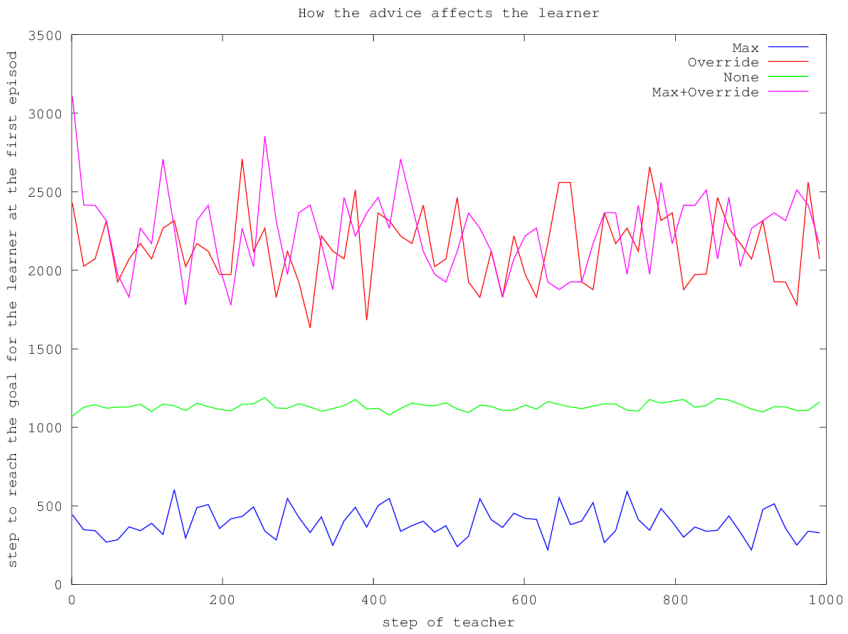

Figure 4: Teacher advices all the time. Number of step to reach to goal at the first episod

Grid World This time in the grid world, the best strategy is Max. The fixed policy strat are really bad, more than without advice.

2.3 Advice before acting

This experiment is used to compare the advice effect when the teacher can correct a mistake of the learner before he perform it. Also, we need to use a different reward function (based on a cost when he advice) for the teacher, otherwise he will advice always and all algorithms will be at the same performance.

2.4 Conclusion

3 Teacher Model

How the teacher is computed?

3.1 Criterion

3.2 Learn to teach

Reward functions Minimizing the number of step -1 everytime.
Maximazing the reward

3.2.1 Advice or not

3.2.2 Learn the action

3.3 Comparaisons

4 Conclusion

4.1 Future research

Differents state representations.
Softmax

4.2 Main results

References

Sutton, Richard S and Andrew G Barto (1998). *Reinforcement Learning : An Introduction*. Cambridge, MA: MIT Press.