Pierre and Marie Curie University

Experiments

Learning from expert

Author:
Matthieu Zimmer

Supervisors:
Paolo Viappiani
Paul Weng

February 1, 2014
Version 1.1

Contents

1 Introduction

Learning to teach : A reinforcement learning approach. The only assumption we made here is that the learner and the teacher have the same action set.

2 Environnement

2.1 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] .

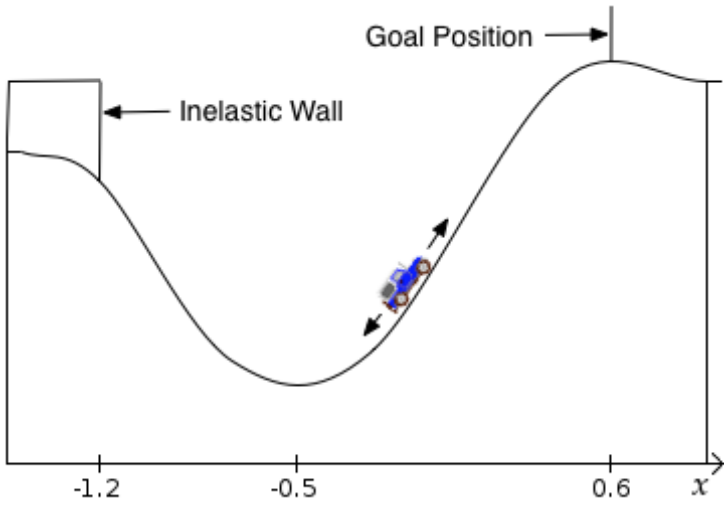

Figure 1: The Mountain Car Problem

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 $x >= 0.6$

2.2 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.

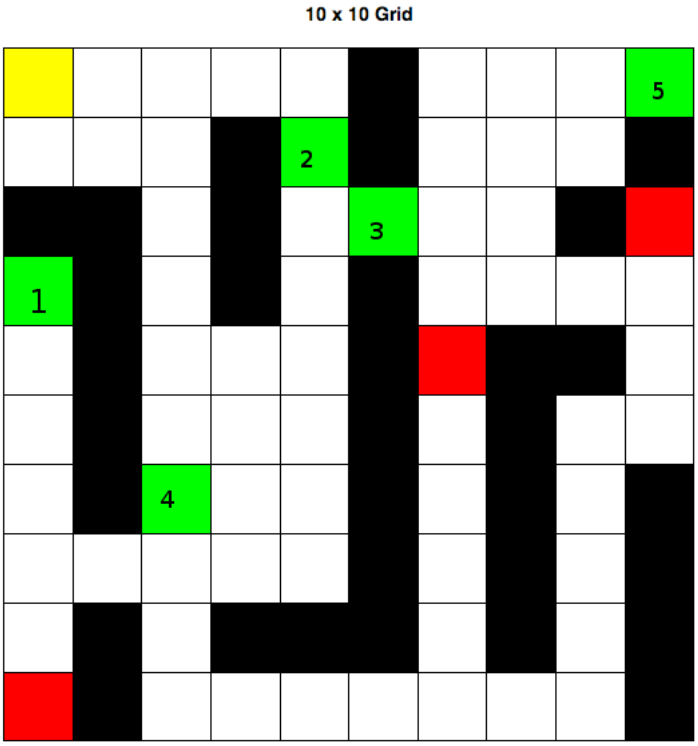

Figure 2: Grid World

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

Conclusion Thus, we have 2 environnement with different property :

- In each state, (except the goal) mountain car receives the same reward
- In each state, mountain car receives a negative reward (the Q function decreased over time)
- The grid environnement reverses these 2 points

3 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 $\vec{\theta}$. Our advise is always the best action.

Notations

- Here a is the recommended action, and s the current state.
- $\vec{\theta}(f(s,a))$ 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 $Q(s,a) = \sum_i \theta_i(f(s,a))$.

Moreover, we asume that :

$$\forall s \in State, \forall a \in Actions(s), \forall a' \in Actions(s), |\vec{\theta}(f(s,a))| = |\vec{\theta}(f(s,a'))|$$

with means that the number of actived features is always the same for a state, in case of using tile coding, it is trivial.

3.1 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.
Tabular version :

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

Approximation version :

$$\vec{\theta}(f(s,a)) = \vec{\theta}(f(s,a')) + \delta$$
$$with\ a' = \operatorname*{argmax}_b Q(s,b)$$

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

3.2 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. We still give advice 10% of time to compare the different strategy.

3.3 Conclusion

4 Teacher Model

How the teacher is computed?

4.1 Heuristics

[Torrey and Taylor, 2013] introduce different teacher model, where 2 reinforcement learning agent learn the same task. One have fully learn, the teacher, and the other one, the learner, starts.

During the learning, using a heuristics, the learner will receive the best action compute by the teacher for a state. The number of advice is fixed.

4.2 Learn to teach

Now, we could like to introduce an other configuration, that we trust better. The teacher will not learn the same task that the learner. However, he computes the best policy internally. He will have to solve an other problem, his state is define like $Teacher\_States = Learner\_States \times Learner\_Actions$
which is the state of the learner and the action he takes in this state.

Reward functions We have to choose which criterion we want to optimize :

Minimizing the number of step

if *giveAdvice* **then**
  $r \leftarrow -1 \times cost$
else
  if *learner_take_best_action* **then**
    $r \leftarrow 0$
  else
    $r \leftarrow -1$
  end if
end if

Maximazing the learner reward

if *giveAdvice* **then**
  $r \leftarrow -1 \times cost \times learner\_reward$
else
  $r \leftarrow learner\_reward$

end if

We will mainly focus on minimizing the number of step.

4.2.1 Advice or not

Our first model will decide if the teacher have to advice with the best action or not. $Teacher\_Actions = \{\neg F, F\}$

4.3 Comparaisons

5 Conclusion

5.1 Future research

Differents state representations.

Softmax

5.2 Main results

After severals fails to define a cost on a advice, we decide to abandon this idea and only use a linear fonction on learner.

- If the number of advice is too big, early advice is the better strategy to adopt for a teacher.
- However, if the number of advice isn't that big, mistake correction and importance advice can do better. In addition, we showed that the best strategy in this case, is our method.

Bibliographie

[Sutton and Barto, 1998] Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning : An Introduction*. MIT Press, Cambridge, MA.

[Torrey and Taylor, 2013] Torrey, L. and Taylor, M. (2013). Teaching on a budget: agents advising agents in reinforcement learning. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, AAMAS '13, pages 1053–1060, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.