TP RL (Reinforcement learning)
Techniques of AI [INFO-H-410]
Correction

v1.0.0

Source files, code templates and corrections related to practical sessions can be found on the UV or on github (`https://github.com/iridia-ulb/INFOH410`).

Value Iteration algorithm

Question 1. Imagine a maze where there is a cookie giving you +20 reward, you want to maximize your reward by learning the policy using value iteration algorithm. There is also a cliff you might fall into which would lead to -50 reward, At any point when taking an action you might slip and go in on a side with probablity 0.05, this means, for a given action for example "forward", you could go "left" or "right" with probablity 0.05 (but not backwards). If you are next to a border any action taking you trough that border makes you stay in place. You start in the black spot with the following default policy and value function.

0	0	-50	+20
0	0	-50	0
0	0	0	0

Value function $V^\pi(s)$

↑	↑	-50	+20
↑	↑	-50	↑
↑	↑	↑	↑

Policy π

a) Is this system markov ?

b) Represent the transition model for any given action.

c) Compute 3 iterations of value iteration using $\gamma = 1$

d) Update the policy accordingly

Answer: (a) yes, it is observable and each action only depends on the current state

(b) the transition model is for a given action:

	0.9	
0.05	↑	0.05

We are in a markovian setup and we want to compute the value function, this function boils down to $V^\pi(s) = Q^\pi(s, \pi(s))$, so we solve the problem using the bellman equation: $Q(s, a) \leftarrow \sum_{s' \in S} P^a_{ss'}(R^a_{ss'} + \gamma \max_{a'} Q(s', a'))$

(c) and (d) Iterations:

0	0	-50	+20
0	0	-50	15.5
0	0	0	0

Value function $V^{\pi}(s)$

↑	←	-50	+20
↑	←	-50	↑
↑	↑	↓	↑

Policy π

0	0	-50	+20
0	0	-50	16.27
0	0	0	13.95

Value function $V^{\pi}(s)$

↑	←	-50	+20
↑	←	-50	↑
↑	↑	↓	↑

Policy π

0	0	-50	+20
0	0	-50	16.3
0	0	10.1	15.34

Value function $V^{\pi}(s)$

↑	←	-50	+20
↑	←	-50	↑
↑	↑	→	↑

Policy π

0	0	-50	+20
0	0	-50	16.31
0	9.09	11.8	15.94

Value function $V^{\pi}(s)$

↑	←	-50	+20
↑	←	-50	↑
↑	→	→	↑

Policy π

0	0	-50	+20
0	5.68	-50	16.32
8.18	10.42	12.43	16.08

Value function $V^{\pi}(s)$

↑	←	-50	+20
↑	↓	-50	↑
→	→	→	↑

Policy π

0	2.61	-50	+20
7.64	6.88	-50	16.32
9.79	11.71	12.59	16.11

Value function $V^\pi(s)$

↑	↓	-50	+20
↓	↓	-50	↑
→	→	→	↑

Policy π

7.01	3.69	-50	+20
9.53	8.42	-50	16.32
11.41	12.26	12.62	16.12

Value function $V^\pi(s)$

↓	↓	-50	+20
↓	↓	-50	↑
→	→	→	↑

Policy π

Q Learning

Question 2. Imagine you have a vertical pole that you want to balance verticaly along its axis, let us use Q learning in order to learn to equilibrate this pole. We will use the "Cartpole-v1" environnement of openai gym (see `https://github.com/openai/gym/wiki/CartPole-v0` and `https://github.com/openai/gym/blob/master/gym/envs/classic_control/cartpole.py` in the description).

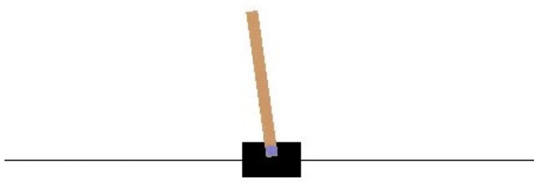

Figure 1: The cartpole to balance verticaly. The cartpole can be seen as an inverted pendulum sitting on a small moving cart.

a) Could we tackle this problem without machine learning ? Do you have any idea how ?

b) Given the state vector of the cart pole, let us keep only 2 features, the angle of the pole and the cart velocity. How can you transform those continuous variables to categorical ones ? Why do you need to do so ?

c) What is the reward function of this problem ? What are the possible actions ?

d) Using Q Learning to solve this problem, what will be the dimensionnality of the Q table.

e) What is the Bellman equation ? Where is it used in Q learning ?

f) Using the provided template, implement you solution in python.

Answer: (a) It could be done using regular control theory (https://en.wikipedia.org/wiki/Inverted_pendulum)

(b) Create bins of values, since Q learning uses a Qtable we need to be able to categorize so that the number of cases in the table in finite.

(c) +1 for each timestep it still balanced

(d) If we keep only 2 features and categorize in 20 bins, and knowing that there are 2 possibles actions, the table will be : 20x20x2

(e) The bellman equation is used to update the approximation of the Q table at each timestep so that: $Q_{t+1}(s,a) = (1-\alpha)Q_t(s,a) + \alpha(R^a_{t+1} + \gamma \max_a Q_t(s_{t+1}, a))$

(f) see github for implementation

Found an error? Let us know: https://github.com/iridia-ulb/INFOH410/issues