UNIVERSITÄT ZU LÜBECK
INSTITUTE FOR ROBOTICS
AND COGNITIVE SYSTEMS

Course: Reinforcement Learning (SS2020)
Prof. Dr. Elmar Rueckert & Honghu Xue, M.Sc.
Assignment Report Nr. II
Date: 27. Mai 2020

Team Nr. 1
Matthias Klatt
Abby Kandathil Abraham
Suneesh Omanakuttan Sumesh Nivas
Sameed Quais
Lakshith Nagarur Lakshminarayana Reddy
Contact: matthias.klatt@student.uni-luebeck.de

Task I, q.1:

For dynamic Programming the Markov reward process must be known, for MC / TD approaches its enough if we can sample the MRP, so they learn from the environment by collecting samples. We will call MC / TD approaches model free approaches because for this algorithms its ok if we don't know the model of the the environment, it's enough if we can observe what happens with if we take actions with the so generated samples we try to estimate our model. So for example if we don't know the transition Probability matrix $\mathcal{P}$ we can try to learn this matrix with MC / TD approaches but we can't do that with DP approaches so we can't use DP approaches because for DP approaches the whole model of the environment must be known.

Task I, q.2:

During the learning process a so called on-policy learning approach follows a policy which is still improving during the learning process. The policy which is improving during the learning process will be denoted with π, where $\pi(s)$ defines what action is to take for the state s.
If we use a $\epsilon - greedy$ approach to ensure that we will observe every possible state-action pair the on-policy learning policy can converge in the limit to this greedy policy, we can work with that if we for example choose $\epsilon = \frac{1}{t}$, where t denotes the episodes number, if we choose them large enough ϵ will converge to zero and our learned policy can converge to the optimal policy.
An off-policy approach also improve a policy π (called *target policy*) but the agent follows a policy b (called *behaviour policy*) during the learning process. The policies aren't the same ($\pi \neq b$) and we require the so called assumption of *coverage*, which means $\pi(a|s) > 0 \implies b(a|s) > 0$ so every action taken under π needs to be taken at least some times from the policy b, if this isn't so we can't use the episodes from b. SO we improve the policy π while the agent follows the policy b during the learning process because b is more greedy but in the end not optimal.
If we choose $\pi(a|s) = b(a|s)$ so the policies are the same we would have the same result as with an on-policy learning approach if we choose the additional variables also the same.

Task I, q.3:

Both methods learn from samples so no model is needed to learn with this approaches. To do an update with the MC approach we need to wait a hole episode to get the return of the episode to update while we can learn online with the one-step TD approach and update after we get a new reward, this can be done because we estimate the following rewards after the current one this leads to some decent bias (TD bootstrap).
In general because of the estimation TD converge faster and needs less samples than MC.
The bias in TD approaches can lead to some problems especially for off-policy learning and when we use function approximation that problem is called *the deadly triad* in [1] because this combination is very likely to fail convergence.
So MC approaches are in general conceptually simpler as TD approaches. MC - learning approaches are more robust and easy to implement in most of the cases but they converge slower than TD approaches.

Task I, q.4:

If the convergence conditions are fulfilled for the step size α we need to ensure that every state-action pair is visited. To have a 100 % successrate we need to ensure that we visite every possible state-action pair infinitely often just under this condition we can ensure that we converge, but this convergence is limited by the greedy

strategy, so we need to define ϵ as described in task I, question 2 this will lead our learned policy converge to th eoptimal policy π^*.

Task II, q.1:

As described above the MC approach just update the $\mathcal{Q}$ function after each episode not during the episode.
We choose an $\epsilon - greedy$ strategy to ensure that we can deal with the exploration-exploitation trade off. So we sample before the agent needs to choose an action a variable p ($p \in [0, 1]$), after that we look if p is smaller than $1 - \epsilon$ the agent will exploit(so we will choose the action so far seems to be the best) and if p is larger we will explore (we will choose a random action).
Also we choose the so called "first-visit "strategy which means that just the first observation of an episode for a state-action pair influence our $\mathcal{Q}$-function, our learning step size is defined as: $\alpha = \frac{1}{N_{visited}}$, so the learning rate changed specific for each state-action pair by counting how often we visit the state-action pair first during the episodes.
We tried to evaluate the learned policy (you can see the learned policy in table 1) and makes 100 runs with the learned policy and count 77 runs were successful, what is the same as with DP approaches. The optimal policy

Tabelle 1 learned policy with the MC approach with 150000 episodes, $\epsilon = 1 - \frac{E_c}{E_t}\%$ and $\alpha = 5\%$

Pos	0	1	2	3
0	left	up	left	up
1	left	right	left	left
2	up	down	left	down
3	left	right	down	right

π^* can be calculated with:

$$\pi^*(s) = \arg\max_a \mathcal{Q}^*(s, a) \forall s \in S. \tag{1}$$

Task II, q.2:

We used the same $\epsilon - greedy$ strategy as in the exercise above to ensure that we will visit every state-action pair and that ϵ will converge to zero.
The update function for the $\mathcal{Q}$ matrix is defined as:

$$\mathcal{Q}(S_t, A_t) \leftarrow \mathcal{Q}(S_t, A_t) + \alpha[\mathcal{R}_{t+1} + \gamma \mathcal{Q}(S_{t+1}, A_{t+1}) - \mathcal{Q}(S_t, A_t)], \tag{2}$$

So we learn the values of state-action pairs. The result of the second and the third subtask can be seen in figure 1. Over 2000 episodes every training run was able to converge around -150.

Task II, q.3:

The result of the Q-learning algorithm and the SARSA algorithm can be seen in figure 1, and as you can see their is no big difference between their rewards during the learning process, because their nearly the same just the update of the Q-function is slightly different.
The update for the $\mathcal{Q}$ matrix is defined as follows:

$$\mathcal{Q}(S_t, A_t) \leftarrow \mathcal{Q}(S_t, A_t) + \alpha[\mathcal{R}_{t+1} + \gamma \cdot \max_a \mathcal{Q}(S_{t+1}, a) - \mathcal{Q}(S_t, A_t)]. \tag{3}$$

Over 2000 episodes every training run was able to converge around -150. So if we compare equation 2 and equation 3 we notice that the factor behind the discount factor γ is different the rest stays the same.
The optimal policy π^* can be calculated for both algorithm with equation 1.
where $\mathcal{Q}^*$ denotes the optimal Q-function, how we can ensure to find the optimal Q-function for Q-learning and SARSA is described in task I, question 4.

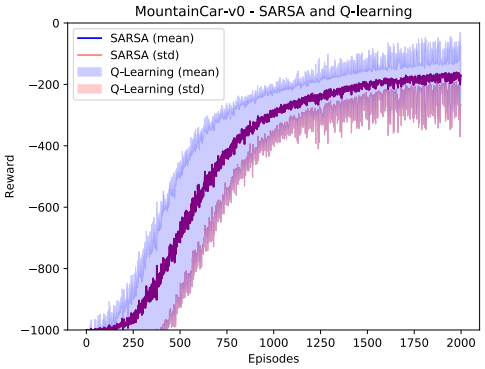

Figure 1 You can see the mean reward of Q-learning and SARSA with over 100 training evaluations, 2000 episodes each, the episode length is set to 1000, with a fixed $\alpha = 0.05$ and $\epsilon = 1 - \frac{E_c}{E_t}$

Task III, q.1:

For the MC approach we learned a policy just by samples from different runs through our environment, our agent don't know in the beginning what states are possible or how the transitions probability matrix $\mathcal{P}$ looks like, so he try's to estimate how it looks like based on the generated samples.
With MC we update our $\mathcal{Q}$ function after each sampled episode during the training phase, but in the beginning our agent don't know where our goal state is and how to get to it, he is just observing what happens if he is choosing different actions for some states, because of our ϵ greedy exploration strategy we assure also that he is trying every possible combination of state-action pairs out.
Because of the estimation of our model for the environment by samples it can happens that the estimation isn't totally right and because of that the learned policy can be (slightly) different to PI/VI where the model is completely known.
In our case the the learned policy from the MC approach is different for state $s(1,1), s(0,2), s(2,3)$ and our terminal state $s(3,3)$.

Task III, q.2:

For the in task II, question 1 described MC approach and environment we changed our ϵ value to see what happens. The result can be seen in table 2. As you can see the states $s(0,0), s(1,0), s(0,2)$ and $s(1,2)$ are different from the above MC approach.

Tabelle 2 learned policy with the MC approach with 150000 episodes, $\epsilon = 5\%$ and $\alpha = \frac{1}{N_{visited}}\%$

Pos	0	1	2	3
0	up	up	right	up
1	right	right	right	left
2	up	down	left	down
3	left	right	down	right

Again we tried the learned policy 100 times to see how successful it is, we get a success-rate of 35 % which isn't good. Our ϵ wasn't greedy enough and so the agent doesn't explore enough at the beginning of the training and the $\mathcal{Q}$ function doesn't improve enough. The agent just exploit and that's leads to this worse policy for this environment with this maximum episode numbers, if we double the episode number the result is still the same. We also get the problem that the fixed ϵ let the policy doesn't converge to the optimal policy it will still be a

little bit greedy (the convergence is limited by ϵ) which also leads to that bad policy.
For SARSA learning and Q-learning not much changes, the reward is nearly the same for this environment this less exploration in the beginning and the limit due to the permanent, constant greediness the learned policy will still explore a little bit but that's tolerable for this environment with this learning algorithms because we have less randomness so for example if want to drive backwards the car will drive backwards to a certain point.

Task III, q.3:

If we set our learning rate α equal to $\frac{1}{N_{visited}}$ it will decrease during the training. The decreasing leads to a bigger influence of the first visited observations for one particular state-action pair (if the agent visit a state-action pair for the first time $\alpha = 1$ but for example if we visit it for the 500th time will α be $\frac{1}{500}$).
First let us check if the convergence conditions (mentioned already in the assignment) will hold for this α:

$$\sum_{k=0}^{\infty} \frac{1}{N_{visited}} = \infty,$$
$$\sum_{k=0}^{\infty} (\frac{1}{N_{visited}})^2 = \frac{\pi^2}{6},$$

So this conditions holds for this step size setting. No let us check if the conditions which are mentioned in task I, question 4 are hold.
Our SARSA and Q-learning approach will visit every state-action pair but not infinitely often, we limit it to 2000 episodes with a maximum length of 1000, so in theory if we would increase the episode number dramatically we will get at least a near optimum policy and if set the maximum episodes to infinity we will reach the optimum policy with a probability of 100 %.
We run our Q-learning and SARSA algorithm with the average sample α and notice that the convergence is so slow that we can't guaranty success during the evaluation, we also tried different initial Q-Values (e^{-7},0,1,100, -100 and 2) but our algorithms converges to slowly to reach the optimal policy also to $\epsilon$to fixed 5% doesn't changed that to increase the episode number to 5000 during training doesn't let the policy converge.
So in theory the convergence to an optimal policy π^* is possible but we need a big episode number during training which lead to a long time period for training.

Task III, q.4:

Optimistic initial values are used to ensure exploration of the model even if we just picked greedy actions, the agent expect because of the high value a much better estimate than he get so he tried out different actions with the result that all possible action are tried before the value estimate converge. So we get a trade-off. Possible valid optimistic initial values for the MountainCar-v0 environment are:

- B:100
- E 10

This values are much better than the agent can receive (they are very optimistic) in this environment were he just get for every step he need to perform to reach the goal a reward with the value of -1, which we can also call punishment, we want the agent to reach the goal as fast as possible and punish every time step he needs to reach the goal.

Literatur

[1] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: an Introduction*. Second edition.