UNIVERSITÄT ZU LÜBECK
INSTITUTE FOR ROBOTICS
AND COGNITIVE SYSTEMS

Course: Reinforcement Learning (SS2020)
Prof. Dr. Elmar Rueckert & Honghu Xue, M.Sc.
Assignment Report Nr. I
Date: 5. Mai 2021

Team Nr. 1
Mevlude Tigre
Fazli Faruk Okumus
Contact: fazli.okumus@student.uni-luebeck.de
mevluede.tigre@student.uni-luebeck.de

Task I: Basics of Reinforcement Learning

1) In probability theory and statistics, A stochastic process has the Markov property if the conditional probability distribution of future states of the process depends only upon the present state. The formal definition of the MDP is that stochastic processes that exhibit the Markov property.
Markov decision process is a tuple of 5 components $\rightarrow (\mathrm{S}, \mathrm{A}, \mathrm{P}, \mathrm{R}, \gamma)$
S $\rightarrow$ A set of states in the environment
A $\rightarrow$ A finite set of actions an agent can take in the environment
P $\rightarrow$ Transition Probability Matrix (for all states)
R $\rightarrow$ Reward component for each state
Y $\rightarrow$ Discount factor ($\gamma \in [0, 1]$)

A Markov chain is a representation of a sequence of variables $(\mathrm{x1}, \mathrm{x2}, \dots)$ which are specified by the conditionals $p(x_i \mid x_{i-1}, \dots, x_1)$. it is assumed that the next state is dependent only upon the current state. This is called the Markov assumption and the resulting model becomes actually a first order.

First-order Markov assumption : $p(x_i \mid x_{i-1}, \dots, x_1) = p(x_i \mid x_{i-1})$

2) The purpose of Reinforcement Learning is to provide more reward transfer from the environment to the agent. The awards achieved as a result of actions that are chosen agent is usually a numeric value. The ultimate goal of the agent is to maximize the total amount of rewards in the long run. This system is called the reward hypothesis.

If the awards that the agent receives from his actions are specified as $R_t, R_{t+1}, R_{t+2}, \cdots$, the formulation below shows the cumulative reward value of the agent. Therefore, the agent's goal is to try to maximize the Gt value. This value is called expected return.

$$G_t \doteq R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_T$$

An additional concept needed when calculating the award is discounting. The sum of the rewards that have been reduced using the discount rate defines the discounted return. The discount rate is indicated by γ and takes a value between 0 and 1. The discount rate determines the importance of future rewards at the time of decision. If it is 0, the agent tends to choose the action that can receive the highest reward at the moment, the closer the parameter is to 1, the more powerful it chooses its actions, taking into account future rewards. Determination of this rate directly affects the learning of the agent.

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

3) The policy is a system that determines probabilistically which actions are chosen. The policy defines the action be taken in the state of the agent. It can be thought of as a kind of action-reaction pairing. If the current situation is accepted as an effect, the agent gives an action in response. The agent seeks the actions it can take by evaluating the situation agent is in.

A deterministic policy, it is the action taken at a specific state: $u = \pi_\theta(u \mid s)$

A stochastic policy, it is the probability of taking an action a given the state s:$p(u \mid s) = \pi_\theta(u \mid s)$

4) While Episodic tasks are the tasks that have a terminal state and start states, In a continuous task, there is not a terminal state and will never end. Episodic RL has a finite amount of states. It tries to maximize the total reward, goal which is expressed as $G_t \doteq R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_T$

On the other hand continuing tasks has no terminal states. So we need to include some discount factor to converge the cumulative reward. $G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$

Task II: Dynamic Programming

1.i) A policy (π) is a function that maps a given state to probabilities of selecting each possible action from current state.

Formally, ***state-value function*** (v_π) (the value of state s under policy π) is the expected return from starting from state s at time t and following policy π thereafter. In other words, it enables us to select the most appropriate possible functions by looking at current state. Mathematically we define $v_\pi(s)$ as

$$v_\pi(s) = E_\pi \left[G_t \mid S_t = s \right]$$

$$= E_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s \right]$$

Formally,the ***action-value function(q_π)*** for policy is the value of action a in state s under policy. Similarly, it enables us to select for the agent the most appropriate any given action from a given state by following policy. Mathematically, we define $q_\pi(s, a)$ as

$$q_\pi(s, a) = E_\pi \left[G_t \mid S_t = s, A_t = a \right]$$

$$= E_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a \right]$$

1.ii) The **optimal policy(v_*)** is related to ***optimal*** **state-value function($v_\pi(s)$)**. Relation between optimal policy and *optimal* state-value function can be define as

$$v_*(s) = \max_\pi v_\pi(s)$$

for all $s \in S$. In other words, v_* gives the most possible appropriate expected return achievable by any policy π for each state.

Similarly, the **optimal policy**(q_*) is related to ***optimal*** **Q-function / action-value function**($q_\pi(s,a)$) and relation between them defined as

$$q_*(s,a) = \max_\pi q_\pi(s,a)$$

for all $s \in S$ and $a \in A(s)$. In other words, q_* gives the most possible appropriate expected return achievable by any policy π for each possible state-action pair.

1.iii) Always, a policy exist and it is better or equal other ones. This called as optimal policy and satisfies $v_\pi(s) \geq v_{\pi'}(s)$ when $\pi > \pi'$. Also, we may have more than one optimal policy. Their action-value function returns the same value even if agent takes different future actions and future discounted rewards, when it reaches the goal. Each state-action pairs are unique for each MDP.

1.iv) Action $a \in A(s)$ which is actions possible in the current state. On the other hand $q_\pi(s,a)$ estimated by using experience, If agent follows current policy distribution for current state,then average will converge. When the $q_\pi(s,a)$ follow the current policy, then it will stabilizes. The distribution depends on the policy changes. That is why action should follow the current policy distribution.

2) Bellman equation is a equation that helps us to solve MDP. In summary, we can say that the Bellman equation decomposes the value function into two parts, the immediate reward plus the discounted future values. This equation simplifies the computation of the value function, such that rather than summing over multiple time steps, we can find the optimal solution of a complex problem by breaking it down into simpler, recursive subproblems and finding their optimal solutions.

General form of Bellman equation

$$v = R + \gamma P v$$

Bellman equation for the State-value function

$$V^\pi(s) = \sum_{a \in \mathcal{A}} \pi(a \mid s) \sum_{s' \in \mathcal{S}} P\left(s' \mid s, a\right) \left[R\left(s, a, s'\right) + \gamma V^\pi\left(s'\right)\right]$$

Bellman equation for the Action-value function

$$Q^\pi(s,a) = \sum_{s' \in \mathcal{S}} P\left(s' \mid s, a\right) \left[R\left(s, a, s'\right) + \gamma \sum_{a' \in \mathcal{A}} \pi\left(a' \mid s'\right) Q^\pi\left(s', a'\right)\right]$$

3) Policy evaluation and Policy development converge to the most appropriate value and most appropriate policy at the end of the updates. The general idea of how these evaluation and development processes interact is called Generalized Policy Iteration (GPI). The value function directly affects the current policy and the current policy value function. After using the value function $v_\pi$to improve the policy, the new $v_\pi^{'}$ is learned and the iteration is repeated to reveal a better π. If both the evaluation process and the improvement process stabilize, that is, no longer produce changes, then the value function and policy must be optimal. We can say that GPI has 2 process named as evaluation and improvement. After some point, can be easily seen that process stabilize and produce no changes. That means, Bellman optimality equation is valid. In each cycle GPI holds stability and optimality to joint process to overall optimality.

Task III: Programming part on Dynamic Programming

1) Frozen-lake-v0 environment consist of 16 grid each of them has to be one of the **F**= Frozen, **H**= Hole, **G**= Goal, **S**= Start

Also, we have 4 actions to take **0**=LEFT, **1**=DOWN , **2**=RIGHT, **3**=UP

On the other hand Frozen-lake-v0 is stochastic environment. Each transition probability 0.333333=1/3. For example, if agent want to go right, there is only have 1/3 probability to go right.

2) Results of the value iteration algorithm in value-iteration-frozen-lake.py is shown below which are provided (Heatmap and actions images respectively) by script itself

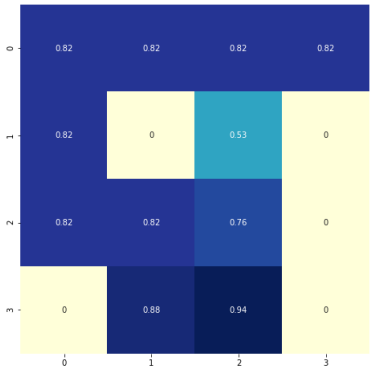

Figure 1 This figure above shows that each grid's probability as a heatmap

```
[[0.82182145 0.82126109 0.82087163 0.82067347]
 [0.82199325 0.         0.52824715 0.        ]
 [0.82226231 0.82260733 0.76389785 0.        ]
 [0.         0.88171208 0.94085038 0.        ]]
[['<' '^' '^' '^']
 ['<' '<' '<' '<']
 ['^' 'v' '<' '<']
 ['<' '>' 'v' '<']]
```

Figure 2 This figure above shows each action taken in each transition

3) Results of the policy iteration algorithm in policy-iteration-frozen-lake.py is shown below which are provided (Heatmap and actions images respectively) by script itself

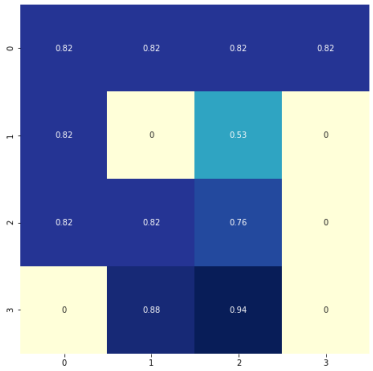

Figure 3 This figure above shows that each grid's probability as a heatmap

```
[[0.82335968 0.82330399 0.82326529 0.8232456 ]
 [0.82337675 0.         0.52929603 0.        ]
 [0.82340349 0.82343778 0.76462558 0.        ]
 [0.         0.88228926 0.94114407 0.        ]]
[['<' '^' '^' '^']
 ['<' '<' '<' '<']
 ['^' 'v' '<' '<']
 ['<' '>' 'v' '<']]
```

Figure 4 This figure above shows each action taken in each transition

4) Policy iteration required admissible initial policy whereas Value iteration does not required.
Policy iteration called full backup solution, it can take significant computations whereas Value iteration partial backup solution and take less computations.
Value iteration is a recursive method, it use previous policies to update new policy.
Thus, the value of a state s is linked to the value of its neighbors. When we get closer to goal we get higher values and same thing occur oppositely when agent get closer to the Holes. Both results of value iteration and policy iteration satisfies this. On the other hand, optimal policy does not mean the optimal trajectory every time. For example, agent would not take a risk by using (1,2) in the grid (in Figure2 and 4), so optimal policies will take long way for stay away from the H grids.
Dynamic programming can be used to find the optimal path length to a goal in the simplified problem. The optimal trajectory does not guarantee the agent to reach the goal just guarantee to find the optimal path.