Norwegian University of Science and Technology
Department of Engineering Cybernetics

Exercise 2
Reinforcement Learning, Q-Learning and SARSA

Introduction

Reinforcement Learning (RL) is an area of Machine Learning (ML) in where an **Agent** learns to take **Actions** in an **Environment** in a way that optimizes a performance measure, usually in terms of maximizing the cumulative **Reward**. Reinforcement Learning builds on many of the same ideas as Dynamic Programming (DP) but the main difference is while DP assumes perfect knowledge of the underlying model of the Markov Decision Process (MDP), RL assumes no knowledge. This means RL is a model free approach, which uses exploration in order to learn how to act in the environment.

In this assignment you will implement two important RL algorithms, namely Q-Learining and SARSA, and test them on a simple Grid World MDP.

Grid World

The Grid World is a simple 2D world, in which an agent is free to roam around, and accumulate reward, see Figure 1. The state is given by the position of the agent within the world, and is represented by a number, as seen in Figure 1, giving the state space $\mathcal{S} = \{0, 1, \dots N-1\}$, where N is the number of possible states. The actions the agent can take, is to either go up, down, left or right, giving the action space $\mathcal{A} = \{U, D, L, R\}$. The probability of the agent taking the desired action is given as p_d, while the probability ending up to either side is given as $\frac{1-p_d}{2}$. If something is blocking our path, and we are not able to go in the desired direction, the agent will stay in the same position.

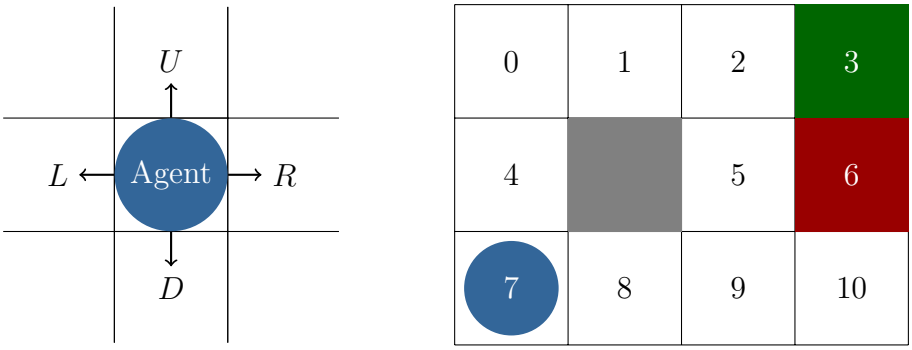

Figure 1: Grid World example with states $\mathcal{S} = \{0, 1, \dots 10\}$, where terminal states are green ($r = +1$) and red ($r = -1$), and available actions for an agent is $\mathcal{A} = \{U, D, L, R\}$

Figure 2: Pendulum environment, where the objective is to swing the pendulum to the upright position ($\theta = 0$) where the actions correspond to applying different amounts of torque to the pendulum.

Pendulum

Pendulum is an environment in where the goal is to swing a pendulum to the upright position as seen in Figure 2. The pendulum is defined by the following differential equation:

$$\ddot{\theta} = \frac{g}{l}\sin(\theta) + \frac{1}{l^2 m}u$$

where $g = 9.81$, $l = 1$ and $m = 1$ is gravity, pendulum length, and pendulum mass respectively. The state of the environment is given as a vector containing the angle, and angular velocity.

$$s = \begin{bmatrix}\theta \\ \dot{\theta}\end{bmatrix}$$

There are three actions available in the environment, giving the action space $\mathcal{A} = \{0, 1, 2\}$, where actions correspond to selecting one of three different control laws:

$$\begin{aligned} a = 0 \quad &: \quad u = -5 \\ a = 1 \quad &: \quad u = +5 \\ a = 2 \quad &: \quad u = -\text{clip}(k_1\theta + k_2\dot{\theta}, -5, 5) \end{aligned}$$

Due to the saturation limits of the control laws it is not possible to directly swing up the pendulum, but the pendulum must in stead swing back and forth to build up momentum in order to swing to the upright position. The reward for the environment is given as:

$$R(s, a) = -\theta^2 - \dot{\theta}^2 - u^2$$

The environment terminates when either when the pendulum starts swinging around to fast ($|\dot{\theta}| > 10$). Or the environment gets sufficiently close to the upright equilibrium ($\theta^2 + \dot{\theta}^2 < 0.001$).

Problems

Problem 1 (40 %) Q-Learning

a Implement Q-Learning (Algorithm 1) for the Grid World problem with an epsilon greedy policy. With learning rate $\alpha = 0.1$, discount rate $\gamma = 1$, and $\epsilon = 0.9$, run the code and report back on the resulting policy and value function.

b Run Q-Learning with maximally greedy policy $\epsilon = 1.0$, minimally greedy policy $\epsilon = 0.0$, and a balanced policy $\epsilon = 0.5$. How do the results relate to the problem of exploration vs exploitation?

c Is Q-Learning an **on policy** or **off policy** method? Explain your reasoning.

Problem 2 (40 %) SARSA

a Implement SARSA (Algorithm 2) for the Grid World problem with an epsilon greedy policy. With learning rate $\alpha = 0.1$, discount rate $\gamma = 1$, and $\epsilon = 0.9$, run the code and report back on the resulting policy and value function.

b Run SARSA with maximally greedy policy $\epsilon = 1.0$, minimally greedy policy $\epsilon = 0.0$, and a balanced policy $\epsilon = 0.5$. How do the results compare to Q-Learning? Explain.

c Is SARSA an **on policy** or **off policy** method? Explain your reasoning.

Problem 3 (20 %) Tabular methods for continuous state spaces

Both Q-Learning and SARSA are tabular methods, which means the action-values are individually stored for each state action pair in a table. This means the methods only works if we have discrete and finite action space $\mathcal{A}$ and state space $\mathcal{S}$. In this problem we will look at the pendulum problem, which has a two dimensional state space $s = [\theta, \dot{\theta}]^\top$, where $\theta \in [-\pi, \pi]$, and $\dot{\theta} \in [-10, 10]$.

a Choose a discretization of the pendulum state space, how many entries in the Q table does the discretization result in?

b Chose either Q-Learning or SARSA, and implement the algorithm for the pendulum problem. With learning rate $\alpha = 0.2$, and discount rate $\gamma = 0.99$, run the code and report back on the resulting behaviour after training, is it what you expected? Also plot the value function, and discuss what you see.

(Hint: The value function is found by as $\max_a Q(\theta, \dot{\theta}, a)$, plotting the intensity, with one state along the x axis and the other state along the y axis you should be able to see a pattern emerge.)

c Try increasing and decreasing the number of points in your discretization, what are the benifits and drawbacks of doing this? How does this relate to the curse of dimensionality?

A Algorithms

The algorithms given below are based on the algorithms in Sutton and Barto[1] chapter 6. It is recommended that you read this chapter before you do the exercise, as it gives a more in depth explanation of the algorithms.

[1] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Algorithm 1 Q-learning

Initialize $Q(s, a)$ arbitrarily; and $Q(terminal, \cdot) = 0$
repeat (For each episode)
    $s \leftarrow$ current state
    repeat (For each step in episode)
        $a \leftarrow$ action from arbitrary policy.
        Take action a observe r, s'
        $Q(s, a) \leftarrow Q(s, a) + \alpha\big[r + \gamma \max_{a'} Q(s', a') - Q(s, a)\big]$
        $s \leftarrow s'$
    until s is terminal
until termination conditions reached

Algorithm 2 SARSA

Initialize $Q(x, u)$ arbitrarily; and $Q(terminal, \cdot) = 0$
repeat (For each episode)
    $s \leftarrow$ current state
    $a \leftarrow$ action from policy derived from Q (e.g. ϵ-gready)
    repeat (For each step in episode)
        Take action a observe r, s'
        $a' \leftarrow$ action from policy derived from Q (e.g. ϵ-gready)
        $Q(s, a) \leftarrow Q(s, a) + \alpha\big[r + \gamma Q(s', a') - Q(s, a)\big]$
        $s \leftarrow s'$, $a \leftarrow a'$
    until s is terminal
until termination conditions reached