TTK23 Autonomous Robotics Systems
Spring 2019

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Exercise 1
Value Iteration and Policy Iteration

Introduction

Dynamic Programming (DP) refers to a collection of algorithms that can be used to compute optimal policies given perfect knowledge of the underlying Markov Decision Process (MDP). While classical DP algorithms are of limited utility for most control problems, because of their assumption of perfect model knowledge and computational expense, they are still theoretically important.

In this assignment you will implement two important DP algorithms, namely Value Iteration and Policy Iteration 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 $\mathcal{S} = \{0, 1, \ldots N-1\}$, 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.

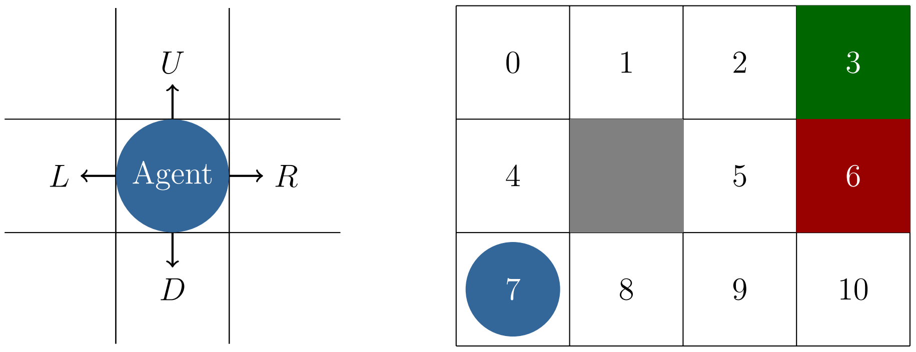
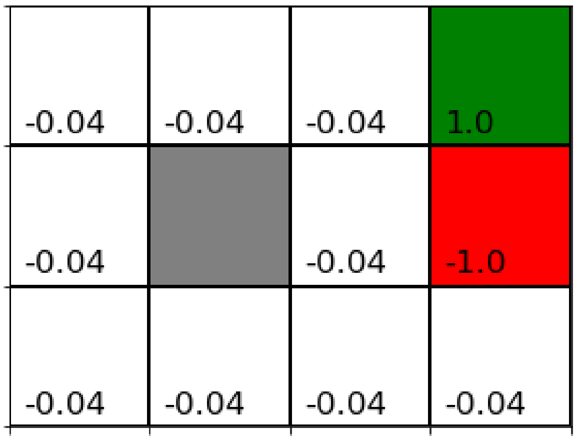
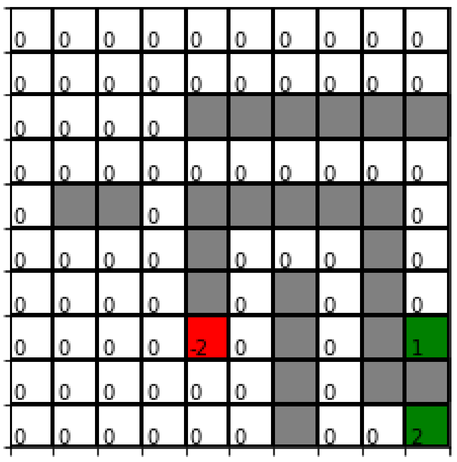


Figure 1: Grid World example with states $\mathcal{S} = \{0, 1, \ldots 10\}$, where terminal states are green ($r = +1$) and red ($r = -1$), and available actions for an agent is $\mathcal{A} = \{U, D, L, R\}$



(a) Small Grid World



(b) Large Grid World

Figure 2: For the exercises we will be using the Grid Worlds above, the numbers in the squares represent the state rewards. Red and green squares are terminal states with positive and negative rewards respectively.

Example

Given the Grid World in Figure 1, with a probability of transitioning in the desired direction $p_d = 0.8$, and the agent starting in the state $s = 7$ taking the action of going to the right $a = R$, what are the probabilities of ending up in all the other states? Since $s' = 8$ is in the desired direction, the probability of transitioning there is $p_d = 0.8$. There is also a probability of moving to either side, i.e up or down of $\frac{1-p_d}{2} = 0.1$. We are free to move up, hence moving to $s' = 4$ has a probability of 0.1, however we can not move down, meaning we in stead end up in the same state as we started. This gives a probability of $s' = 7$ of 0.1.

$$p(s' = 7|s = 7, a = R) = 0.1$$
$$p(s' = 4|s = 7, a = R) = 0.1$$
$$p(s' = 8|s = 7, a = R) = 0.8$$

All other stats are not in reach within one step, hence the probability of ending in any other state is 0.0

For the rest of the exercise we will be looking at two different Grid Worlds given in Figure 2. The two Grid Worlds have the same behaviour, however differ slightly in that the smaller Grid World has a transition probability $p_d = 0.8$, while the larger Grid World has a transition probability of $p_d = 0.9$. It is also worth noting that the smaller Grid World has a small transition reward/cost between non terminal states of -0.04, while the larger Grid World has no transition reward/cost.

Problem 1 (40 %) Value Iteration

a Implement the Value Iteration algorithm (Algorithm 1) in the code handout. Run it on the Grid World in *gridworlds/tiny.json* with a discount rate $\gamma = 1.0$. Report the optimal value function value of all the states.

b Implement the policy evaluation algorithm (Algorithm 2), which finds the optimal policy when given the optimal value function. Report the optimal policy for all states in the Grid World *gridworlds/tiny.json*.

c Run your value iteration algorithm on *gridworlds/large.json* with a discount rate $\gamma = 0.9$ and $\gamma = 0.99$. Explain the difference observed in the policy.

d The gridworld in *gridworlds/large.json* has no penalty on transitioning between states. With a discount rate $\gamma = 1.0$, what do you expect the value function values to look like? Explain your reasoning. Run *gridworlds/large.json* with discount rate $\gamma = 1.0$ did you get the expected results, why/why not?

Problem 2 (40 %) Policy Iteration

a Implement a Policy Evaluation method, you can either choose Iterative Policy Evaluation (Algorithm 4), or Exact Policy Evaluation (Algorithm 5). Give an an explanation for your choice of method.

b Implement the Policy Iteration algorithm (Algorithm 3), using the Policy Evaluation method you just implemented. Run it on the Grid World in *gridworlds/tiny.json* with a discount rate $\gamma = 1.0$. Report the optimal value function values, and policy for all states. Do they match the results from Value Iteration?

c For grid worlds with multiple equally good policies the, the Policy Iteration algorithm may not terminate. Why is this? Why will Value Iteration terminate for the same problem? (Hint: look at the termination conditions)

d For the Grid World in *gridworlds/tiny.json* with a discount rate $\gamma = 1.0$, record the number of iterations, as well as the largest value function error $||V_{i+i} - V_i||_\infty$ for both Policy Iteration and Value Iteration. Explain the results.

Problem 3 (20 %) Satisfy your curiosity

Play around with Policy Iteration and/or Value Iteration. What you do is up to you, but you may also choose from the suggestions below. All we require is that you describe what you did, and your findings.

Suggestions:

- Try a range of discount rates γ on the different Grid Worlds.
- Make your own grid world, and test it on your algorithms.
- Change the parameters such as rewards, probabilities and/or terminal states of the Grid Worlds, and observe the difference in value function and policy.
- Implement and compare both policy evaluation methods (Exact Policy Evaluation, and Iterative Policy evaluation).
- Apply your algorithms to a completely different Markov Decision Process.

A Algorithms

The algorithms given below are based on the algorithms in Sutton and Barto[1] chapter 4.

A.1 Value Iteration

Algorithm 1 Value Iteration

```
function VALUE-ITERATION($mdp, \theta$)
    $V(s) \leftarrow 0 \quad \forall s \in \mathcal{S}$
    repeat
        $\delta \leftarrow 0$
        for each $s \in \mathcal{S}$ do
            $v \leftarrow V(s)$
            $V(s) \leftarrow \max_{a \in \mathcal{A}(s)} \sum_{s'} P(s'|s,a)\big(R(s',s,a) + \gamma V(s')\big)$
            $\delta \leftarrow \max(\delta, |v - V(s)|)$
        end for
    until $\delta < \theta$
    return $V$
end function
```

Algorithm 2 Policy

```
function POLICY($mdp, V$)
    for each $s \in \mathcal{S}$ do
        $\pi(s) = \text{argmax}_{a \in \mathcal{A}(s)} \sum_{s'} P(s'|s,a)\big(R(s',s,a) + \gamma V(s')\big)$
    end for
    return $\pi$
end function
```

A.2 Policy Iteration

Algorithm 3 Policy Iteration

```
function POLICY-ITERATION($mdp$)
    $\pi \leftarrow$ random initial policy
    repeat
        $\pi' \leftarrow \pi$
        $V^\pi \leftarrow$ POLICY-EVALUATION($mdp, \pi$)
        $\pi(x) \leftarrow \text{argmax}_{a \in \mathcal{A}(s)} \sum_{s'} P(s'|s,a)\big[R(s',s,a) + \gamma V^\pi(s')\big]$
    until $\pi = \pi'$
    return $\pi, V^\pi$
end function
```

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

Algorithm 4 Iterative Policy Evaluation

```
1: function ITERATIVE-POLICY-EVALUATION($mdp, \pi, \theta$)
2:     $V^\pi(s) \leftarrow 0 \quad \forall s \in \mathcal{S}$
3:     repeat
4:         $\delta \leftarrow 0$
5:         for each $s \in \mathcal{S}$ do
6:             $v \leftarrow V^\pi(s)$
7:             $V^\pi(s) \leftarrow \sum_{s'} P(s'|s,\pi(s))\big(R(s',s,\pi(s)) + \gamma V^\pi(s')\big)$
8:             $\delta \leftarrow \max(\delta, |v - V^\pi(s)|)$
9:         end for
10:    until $\delta < \theta$
11:    return $V^\pi$
12: end function
```

Algorithm 5 Exact Policy Evaluation

```
1: function EXACT-POLICY-EVALUATION($mdp, \pi$)
2:     $\boldsymbol{A} \leftarrow \begin{bmatrix} \gamma P(s_1|s_1,\pi(s_1)) - 1 & \gamma P(s_2|s_1,\pi(s_1)) & \dots & \gamma P(s_n|s_1,\pi(s_1)) \\ \gamma P(s_1|s_2,\pi(s_2)) & \gamma P(s_2|s_2,\pi(s_2)) - 1 & \dots & \gamma P(s_n|s_2,\pi(s_2)) \\ \vdots & \vdots & \ddots & \vdots \\ \gamma P(s_1|s_n,\pi(s_n)) & \gamma P(s_2|s_n,\pi(s_n)) & \dots & \gamma P(s_n|s_n,\pi(s_n)) - 1 \end{bmatrix}$
3:     $\boldsymbol{b} \leftarrow \begin{bmatrix} -\sum_{s'\in\mathcal{S}} P(s'|s_1,\pi(s_1))R(s',s_1,\pi(s_1)) \\ -\sum_{s'\in\mathcal{S}} P(s'|s_2,\pi(s_2))R(s',s_2,\pi(s_2)) \\ \vdots \\ -\sum_{s'\in\mathcal{S}} P(s'|s_n,\pi(s_n))R(s',s_n,\pi(s_n)) \end{bmatrix}$
4:     Solve linear system of equations: $\boldsymbol{A}V^\pi = \boldsymbol{b}$
5:     return $V^\pi$
6: end function
```