
Exercise 2: Policy & Value Iteration

Nico Ott 4214197, Lior Fuks 4251285, Hendrik Vloet 4324249

November 22, 2017

1 Policy Iteration

- See according code file `policy_iteration.py`

2 Value Iteration

a)

- See according code file `value_iteration.py`

b)

- The policy iteration algorithm consists of two explicit components, the policy evaluation and the policy improvement step. These two steps are looped until convergence is achieved:

$\pi_0 \xrightarrow{\text{eval}} v_{\pi_0} \xrightarrow{\text{improve}} \pi_1 \xrightarrow{\text{eval}} \dots \xrightarrow{\text{improve}} \pi_* \xrightarrow{\text{eval}} v_*$

Value iteration does not use an explicit policy evaluation step: $v_1 \rightarrow v_2 \rightarrow \dots \rightarrow v_*$.

After convergence to v_* the optimal policy π_* is instantly known. In praxis, convergence is obtained by check the difference of the two latest value functions. If it does only change for a small amount (~ 0.0001).

- One drawback of policy iteration is, that it can be computationally inefficient because we have to wait until it converges and the algorithm only does that in its limits (obviously). Value iteration converges much faster due to the lack of an explicit policy evaluation.
- value iteration uses the Bellman optimality equation in order to update their value function and policy iteration uses the bellman expectation equation and then greedily improves its policy.
- Similarity: value iteration is equivalent to policy iteration if policy iteration is terminated after one complete sweep of all states.

3 Experiences

- **Hendrik**

- Invested time:
 - * Lecture 3: 3h
 - * Exercise : ~ 24h
- understanding issues with the unittest structure of the python environment
- I had trouble with comprehending why the expected policy always had only one possible action per state. In my understanding there should be at least in the corner states of the gridworld, where the agent's distance to one target state is equivalent to more than one viable action