Deep Reinforcement Learning

Exercise 1

Lecturer: Armin Biess, Ph.D.

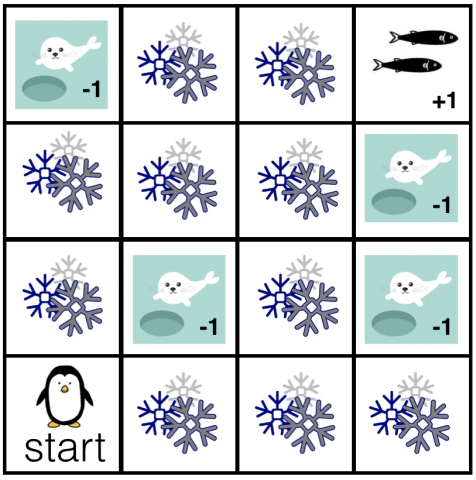
Due date: 11.6.2020

1. Define and ***explain*** the following terms [10 pts]

1. Markov Decision Process (MDP). Which problems can be solved with an MDP? Provide an example.
2. State-value function
3. Action-value function
4. Policy
5. Dynamic programming
6. Value iteration
7. Policy iteration
8. Reinforcement learning (RL). Which problems can be solved with RL? What is the difference to an MDP? Provide an example.

2. Markov decision process (MDP) - FrozenLake [90 pts]

Given is a penguin on a frozen lake, which is described by a 4x4 grid world with holes and a goal state (fish), both defining terminal states. For transitions to terminal states the penguin gets a reward of $+1$ for the goal state and a reward of -1 for the holes, whereas for all other transitions the penguin gets a reward of $r = -0.04$. The penguin can take four possible actions $= \{N, E, S, W\}$, but the surface is slippery and only with probability 0.8 the penguin will go into the intended direction and with probability 0.1 to the left and 0.1 to the right of it. It is assumed that the boundaries are reflective, i.e., if the penguin hits the edge of the lake it remains in the same grid location. Find the optimal policy for the penguin to get to the fish by solving the corresponding MDP.



For the implementation of the MDP a `World` class is provided. The `World` class includes three methods:

- `plot_world`: plots the gridworld
- `plot_value(v)`: takes a column vector of values $v(s)$ as input and plots the values into the gridworld
- `plot_policy(π)`: takes a column vector of actions $a = \pi(s)$ (deterministic policies) or a matrix of actions $\pi(a|s)$ (stochastic policies) as inputs and plots the policy into the gridworld

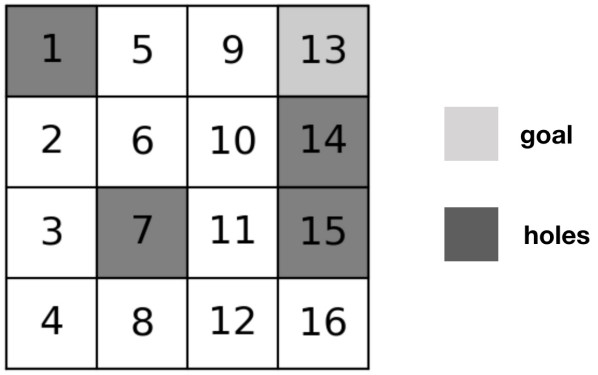
Experiment with the `World` class. For example, run the following python script:

```
>> world = World()
>> world.plot()
>> world.plot_value([np.random.random() for i in range(world.nStates)])
>> world.plot_policy(np.random.randint(1, world.nActions,(world.nStates, 1)))
```

a.)[30 pts] Construct the transition model $p(s'|s, a)$ and reward function $r(s)$ for this MDP. Draw a graphical model (by hand, powerpoint etc) of the MDP for $a = N$ showing all states, transition probabilities and rewards.
b.)[10 pts] Solve the MPD using *value iteration* with $\gamma = 1$ and a termination threshold of $\theta = 10^{-4}$. Plot the optimal values into the gridworld by using the `plot_value` of the `World` class. Plot the optimal policy in a different figure by using the `plot_policy` function of the `World` class.
c.)[10 pts] Repeat your value iteration algorithm with a reduced discount factor $\gamma = 0.9$ and $r = -0.04$. Explain how these changes in parameters affect the optimal policy. Generate two plots showing the optimal value function and optimal policy.
d.)[10 pts] Repeat your value iteration algorithm with a reduced discount factor $\gamma = 1$ and reward $r = -0.02$. Explain how these changes in parameters affect the optimal policy. Generate two plots showing the optimal value function and optimal policy.
e.)[30 pts] Solve the MPD using *policy iteration* with $\gamma = 0.9$ and $r = -0.04$. Initialize your policy iteration algorithm with a uniform random policy. Plot the value function and policy after each iteration step into two different figures of the gridworld by using the `plot_value` and `plot_policy` function of the `World` class, respectively. Compare your results with the results obtained using value iteration.

Important:

- Label the states as follows:



- Label the actions as follows: $\{N, E, S, W\} = \{1, 2, 3, 4\}$.