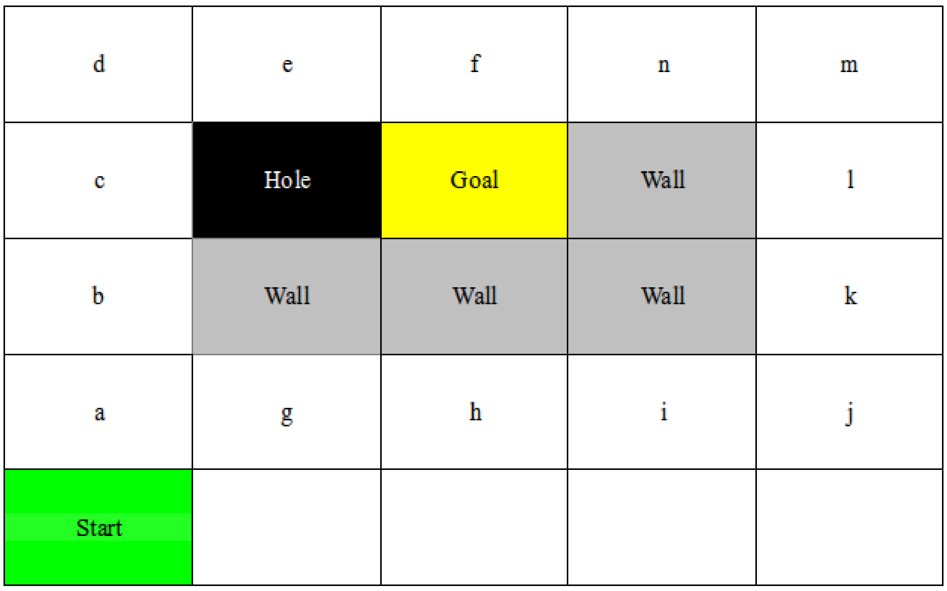
Homework 3: Sampling-Based Approaches, Function Approximation

T-747: Reinforcement Learning

Fall, 2023

1. *(Total points: 30)* Consider the grid-world domain shown below. The possible actions are the four cardinal direction movements: up, down, left and right. All state transitions are deterministic. Moving in some direction changes the agent's position by one square in the given direction, unless it bumps into a wall or hits the edge of the world, in which case the position stays unchanged. Falling in the hole or reaching the goal both leads to terminal states. The agent gets a reward of +10 points for reaching the goal state, -1 for bumping into a wall or the edge of the world, and -50 for falling into a hole. Future rewards are discounted at a rate of 0.9 per step.



   (a) (5 points) What is the value of state a under the optimal policy?

   (b) (5 points) Q-Learning is an Off-Policy TD control method while Sarsa is said to be On-Policy. What does that mean?

   (c) (10 points) Suppose Sarsa and Q-Learning would be used to learn a policy for the problem (without fixing the episodes, but picking them using an epsilon-greedy policy). Which policies (preferred paths) would you expect the two algorithms to learn? Justify your answer.

   (d) (10 points) Does the value for ϵ matter for the answer of the previous question? How? (That is, which of the methods is affected in which way by changing ϵ?)

2. Deadly Triad *(Total points: 20)*

   (a) (5 points) Shortly explain what is meant by "deadly triad" when discussing reinforcement learning algorithms. (What is the problem and what causes it?)

   (b) (15 points) Baird's counterexample is used to illustrate a potential instability of reinforcement learning algorithms. Show that dropping any of the three parts of the deadly triad (function approximation, temporal difference learning/bootstrapping, off-policy learning) would result in convergence to the correct values for Baird's counterexample.

3. *(Total points: 100)*

Consider the Lunar Lander environment from `https://gymnasium.farama.org/`.

The goal is to land a spacecraft on the moon safely (without crashing) and with as little manouvering as possible. The state space is 8-dimensional and consists of

- horizontal and vertical position of the lander (x and y)
- velocities in x and y direction
- the angle of the lander and angular velocity
- for both legs whether they touch the ground or not

Your task is to use Reinforcement Learning to find a good policy for this setting.

(a) (10 points) Implement a purely random agent for the enviroment as a baseline to compare other agents to.

(b) (10 points) In order to use a tabular approach for solving the problem, you need to first discretize the state space. One way to achieve that is to split each attribute (except for the boolean ones) evenly into n intervals each, resulting in at most $4 * n^6$ different states to consider (not all of those might be reachable). Implement a function that maps each state/observation to its discretized version. Keep n as a parameter to tune later and set $n = 10$ as a default.

(c) (15 points) Implement **on-policy Monte Carlo control** to learn a policy for the task. Use a tabular approach for storing $Q(s, a)$

Use an ϵ-greedy policy with $\epsilon = 0.1$, a learning rate of 0.1 and discount future rewards with 0.99.

(d) (10 points) Train a policy using MC and evaluate it. Plot a curve showing the progress of training, for example, the quality of the policy (average return) over the number of training episodes. Compare the performance to a purely random agent. What can be said about the performance of the algorithm with these parameters and this state representation on the given task?

(e) (15 points) Implement **Q-Learning** to learn a policy for the task. Use the same parameters as for Monte-Carlo as defaults.

(f) (10 points) Again, train a policy and evaluate it. Plot a curve showing the progress of training, i.e., the quality of the policy over the number of game steps (frames) trained on. Interpret the learning curve. What can be said about the performance of the algorithm with these parameters and this state representation on the given task? How does this compare to the performance of Monte-Carlo?

(g) (20 points) Pick one of the algorithms and try to tune the parameters as well as the discretization in order to improve the performance. Interpret the results of the experiments. Do the optimal parameter values depend on each other?

Hint: The paper `https://arxiv.org/abs/2011.11850` might give some useful information about reasonable parameter values.

(h) (10 points) Once you have learned a good policy, check whether it is robust enough to handle small variations in the environment. One way to do this relatively easily is to enable wind (`enable_wind` and `wind_power` parameters in texttgym.make) and measure the performance of the policy with different amounts of wind.

(i) **(50 bonus points)** Implement Deep-Q-Learning, i.e., use a neural network as function to predict Q values for the actions. Compare the performance to the previous best model. Also, compare the run-time required for training. Note that training with Deep-Q-Learning might take a lot longer than training the other methods.

Hint: The same paper as above might help getting a reasonable setup for the network and the training parameters. It is possible to make the network somewhat smaller (e.g., 2 hidden layers with 64 units each might do) to make training a bit less expensive.