732A96 Advanced Machine Learning

LAB 3: Reinforcement learning

Johannes Hedström

STIMA
Department of Computer and Information Science
Linköpings universitet

2024-09-27

Contents

1 1

Q-Learning. The file RL Lab1.R in the course website contains a template of the Q- learning algorithm. You are asked to complete the implementation. We will work with a grid- world environment consisting of H x W tiles laid out in a 2-dimensional grid. An agent acts by moving up, down, left or right in the grid-world. This corresponds to the following Markov decision process:

- State space: $S = (x, y) \mid x \in 1, ..., H, y \in 1, ..., W$.
- Action space: $A = up, down, left, right$.

Additionally, we assume state space to be fully observable. The reward function is a deterministic function of the state and does not depend on the actions taken by the agent. We assume the agent gets the reward as soon as it moves to a state. The transition model is defined by the agent moving in the direction chosen with probability $(1 - \beta)$. The agent might also slip and end up moving in the direction to the left or right of its chosen action, each with probability $\beta/2$. The transition model is unknown to the agent, forcing us to resort to model-free solutions. The environment is episodic and all states with a non-zero reward are terminal. Throughout this lab we use integer representations of the different actions: Up=1, right=2, down=3 and left=4.

2 2

For our first environment, we will use H = 5 and W = 7. This environ- ment includes a reward of 10 in state (3,6) and a reward of -1 in states (2,3), (3,3) and (4,3). We specify the rewards using a reward map in the form of a matrix with one entry for each state. States with no reward will simply have a matrix entry of 0. The agent starts each episode in the state (3,1). The function vis environment in the file RL Lab1.R is used to visualize the environment and learned action values and policy. You will not have to modify this function, but read the comments in it to familiarize with how it can be used. When implementing Q-learning, the estimated values of Q(S, A) are commonly stored in a data-structured called Q-table. This is nothing but a tensor with one entry for each state- action pair. Since we have a H $\times$ W environment with four actions, we can use a 3D-tensor of dimensions H x W x 4 to represent our Q-table. Initialize all Q-values to 0. Run the function vis environment before proceeding further. Note that each non-terminal tile has four values. These represent the action values associated to the tile (state). Note also that each non-terminal tile has an arrow. This indicates the greedy policy for the tile (ties are broken at random).

our are requested to carry out the following tasks.

- Implement the greedy and ϵ -greedy policies in the functions GreedyPolicy and EpsilonGreedyPolicy of the file RL Lab1.R. The functions should break ties at random, i.e. they should sample uniformly from the set of actions with maximal Q-value.
- Implement the Q-learning algorithm in the function q learning of the file RL Lab1.R. The function should run one episode of the agent acting in the environment and update the Q-table accordingly. The function should return the episode reward and the sum of the temporal-difference correction terms R + maxa Q(S , a) − Q(S, A) for all steps in the episode. Note that a transition model taking as input is already implemented for you in the function transition model.
- Run 10000 episodes of Q-learning with $\epsilon = 0.5$, $\beta = 0$, $\alpha = 0.1$ and $\gamma = 0.95$. To do so, simply run the code provided in the file RL Lab1.R. The code visualizes the Q-table and a greedy policy derived from it after episodes 10, 100, 1000 and 10000. Answer the following questions:
  - What has the agent learned after the first 10 episodes ?

– Is the final greedy policy (after 10000 episodes) optimal for all states, i.e. not only for the initial state ? Why / Why not ?

– Do the learned values in the Q-table reflect the fact that there are multiple paths (above and below the negative rewards) to get to the positive reward ? If not, what could be done to make it happen ?

3 3

Environment B. This is a 7x8 environment where the top and bottom rows have negative rewards. In this environment, the agent starts each episode in the state (4, 1). There are two positive rewards, of 5 and 10. The reward of 5 is easily reachable, but the agent has to navigate around the first reward in order to find the reward worth 10. Your task is to investigate how the ϵ and parameters affect the learned policy by running 30000 episodes of Q-learning with $\epsilon = 0.1, 0.5$, $\gamma = 0.5, 0.75, 0.95$, $\beta = 0$ and $\alpha = 0.1$. To do so, simply run the code provided in the file RL Lab1.R and explain your observations.

4 4

Environment C. This is a smaller 3 x 6 environment. Here the agent starts each episode in the state (1,1). Your task is to investigate how the β parameter affects the learned policy by running 10000 episodes of Q-learning with $\beta = 0, 0.2, 0.4, 0.66$, $\epsilon = 0.5$, $\gamma = 0.6$ and $\alpha = 0.1$. To do so, simply run the code provided in the file RL Lab1.R and explain your observations.

#5

he file RL Lab2 Colab.ipynb in the course website contains an imple- mentation of the REINFORCE algorithm. The file also contains the result of running the code, so that you do not have to run it. So, you do not need to run it if you do not want.1 Your task is to study the code and the results obtained, and answer some questions. We will work with a 4 x 4 grid. We want the agent to learn to navigate to a random goal position in the grid. The agent will start in a random position and it will be told the goal position. The agent receives a reward of 5 when it reaches the goal. Since the goal position can be any position, we need a way to tell the agent where the goal is. Since our agent does not have any memory mechanism, we provide the goal coordinates as part of the state at every time step, i.e. a state consists now of four coordinates: Two for the position of the agent, and two for the goal position. The actions of the agent can however only impact its own position, i.e. the actions do not modify the goal position. Note that the agent initially does not know that the last two coordinates of a state indicate the position with maximal reward, i.e. the goal position. It has to learn it. It also has to learn a policy to reach the goal position from the initial position. Moreover, the policy has to depend on the goal position, because it is chosen at random in each episode. Since we only have a single non-zero reward, we do not specify a reward map. Instead, the goal coordinates are passed to the functions that need to access the reward function.

5 6

Environment D. In this task, we will use eight goal positions for training and, then, validate the learned policy on the remaining eight possible goal positions. The training and validation goal positions are stored in the lists train goals and val goals in the code in the file RL Lab2 Colab.ipynb. The results provided in the file correspond to running the REIN- FORCE algorithm for 5000 episodes with $\beta = 0$ and $\gamma = 0.95$. Each training episode uses a random goal position from train goals. The initial position for the episode is also chosen at random. When training is completed, the code validates the learned policy for the goal positions in val goals. This is done by with the help of the function vis prob, which shows the grid, goal position and learned policy.

Note that each non-terminal tile has four values. These represent the action probabilities associated to the tile (state). Note also that each non- terminal tile has an arrow. This indicates the action with the largest probability for the tile (ties are broken at random). Finally, answer the following questions:

- Has the agent learned a good policy? Why / Why not ?
- Could you have used the Q-learning algorithm to solve this task ?

6 7

Environment E. In this task, the goals for training are all from the top row of the grid. The validation goals are three positions from the rows below. To solve this task, simply study the code and results provided in the file RL Lab2 Colab.ipynb and answer the following questions:

- Has the agent learned a good policy? Why / Why not ?
- If the results obtained for environments D and E differ, explain why