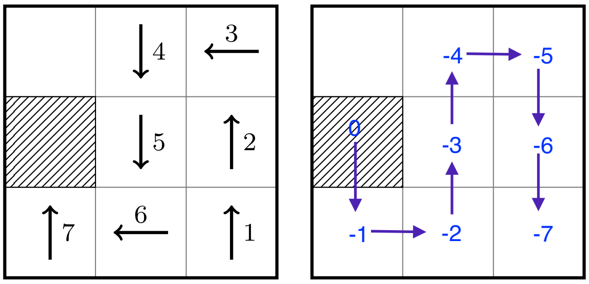
Reinforcement Learning Exercise

Question 1: Monte Carlo Estimation

Given is a gridworld with possible actions in each direction and a reward of -1 for each action. The marked grid is the terminal state. The actions for two episodes have been calculated and are displayed below. The numbers indicate the order of the steps.

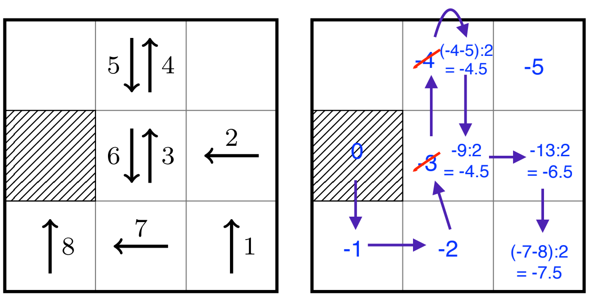
(a) The first episode uses the following actions. Calculate the value function after the episode.

Actions: Value Function:



(b) (8 points) Here are the actions for the second episode. What is the updated value function?

Actions: Value function:



Monte Carlo Prediction for estimating v_π

Input: a policy π
Initialize:
  $V(s) \in \mathbb{R}$ (arbitrarily)
  $\text{Returns}(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$
Loop forever:
  Generate episode following π: $S_0, A_0, R_0, S_1, ..., R_T$
  $G \leftarrow 0$
  Loop for each step of the episode, $t = T-1, T-2, ..., 0$:
    $G \leftarrow \gamma G + R_{t+1}$
    Unless S_t appears in $S_{t-1}, ...S_0$:
      Append G to $Returns(S_t)$
      $V(S_t) \leftarrow \text{average}(Returns(S_t))$

Question 2: Time Difference Learning

We would now like to use TD(0) prediction for the same problem. The actions are the same as above. The value function is initially -1 for every state except the terminal state. A constant step size $\alpha = 0.2$ is used. Run the algorithm for 2 episodes.

Initiale Value Funktion:

-1.0	-1.0	-1.0
	-1.0	-1.0
-1.0	-1.0	-1.0

(a) The following actions are used for the first episode. Specify the value function after the first episode.

Actions:

	↓ 4	← 3
	↓ 5	↑ 2
↑ 7	← 6	↑ 1

Value function:

-1.0	-1.2	-1.2
	-1.2	-1.2
-1.0	-1.2	-1.2

(b) These are the actions for the second episode. What is the updated value function?

Actions:

	5 ↓↑ 4	
	6 ↓↑ 3	← 2
↑ 8	← 7	↑ 1

Value Function:

-1.0	-1.6	-1.2
	-1.56	-1.4
-1.0	-1.36	-1.4

TD(0) for estimating v_π

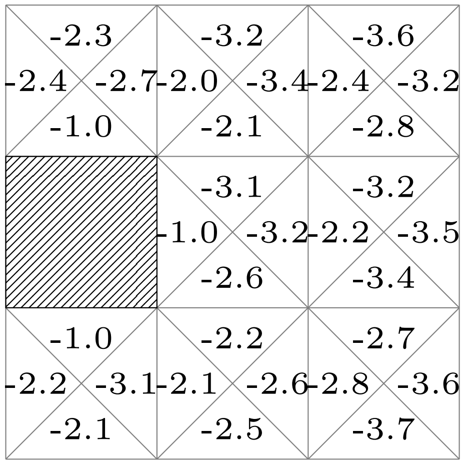
Input:
  the policy π to be evaluated
  step size $\alpha \in (0, 1]$
Initialize:
  $V(S)$ arbitrarily (except $V(\text{terminal} = 0)$

Loop for each episode:
  Initialize S
  Loop for each step of the episode:
    $A \leftarrow$ action given by π for S
    Take action A, observe R, S'
    $V(S) \leftarrow V(S) + \alpha[R + \gamma V(S') - V(S)]$
    $S \leftarrow S'$
  until S is terminal

Question 3: Q-Learning

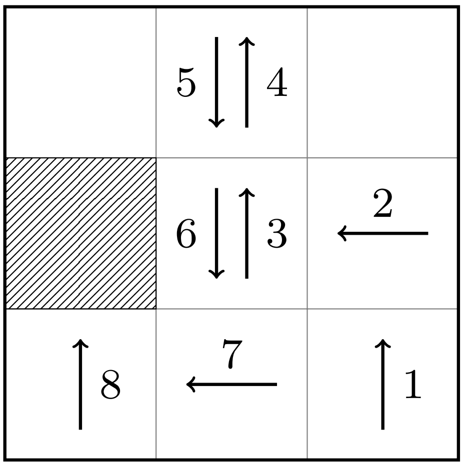
Instead of prediction, we now want to use Q-learning as the control algorithm. To do this, we want to calculate the action-value function and then a better policy in each case. You have already performed iterations, and the current action values for each state in the direction of the actions N, E, S, and W are as shown below.

Initial action-value function:

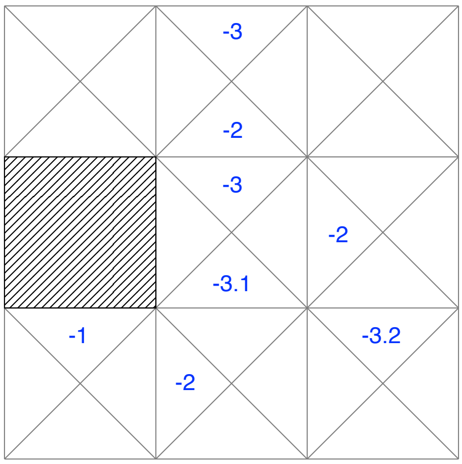


(a) The actions drawn from the policy for each state are shown below. Calculate the action-value functions for the episode shown. This time, use a constant step size $\alpha = 1$ for the update. Only enter the values that have changed.

Actions:



Update the action-value function Q:



Q-learning for estimating $Q \approx q_*$

Input:
  step size $\alpha \in (0, 1]$
  small $\epsilon > 0$
Initialize:
  $Q(s, a)$ for all $s \in \mathcal{S}^+, a \in \mathcal{A}$ arbitrarily (except $Q(\text{terminal}, \cdot) = 0$)

Loop for each episode:
  Initialize S
  Loop for each step of the episode:
    Choose A from S using a policy derived from Q (e.g., ϵ-greedy)
    Take action A, observe R, S'
    $Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma \max_a Q(S', a) - Q(S, A)]$
    $S \leftarrow S'$
  until S is terminal

(b) Now calculate the greedy policy for all states from the values after the above update and plot them.