Artificial Intelligence Homework 3
Reinforcement

2014 / 12 / 10

Question 1 – Value Iteration Agent

- An MDP is given
- U(s): self.values = util.Counter() – a dictionary
- __init__(self, mdp, discount = 0.9, iterations = 100):

  For every iteration, for every state in the MDP, find the maximum value of Q(*s*, *a*) for all possible actions of state *s*

  – Recall that $U(s) = \max_{a \in A(s)} Q(s, a)$
- getValue(self, state):

  return self.values[state]

Question 1 – Value Iteration Agent

- getQValue(self, state, action):

  Use getTransitionStatesAndProbs in mdp.py

  $$Q(s,a) = \sum_{s'} P(s'|s,a) \cdot \left[R(s'|s,a) + \gamma U(s')\right]$$

- getPolicy(self, state):

  If terminal state, return none.

  Else, return the action that results in the maximum value of

  $$E[\text{utility of taking } a] = \sum_{s'} P(s'|s,a) \cdot U(s')$$

Question 2 – Value Iteration Agent

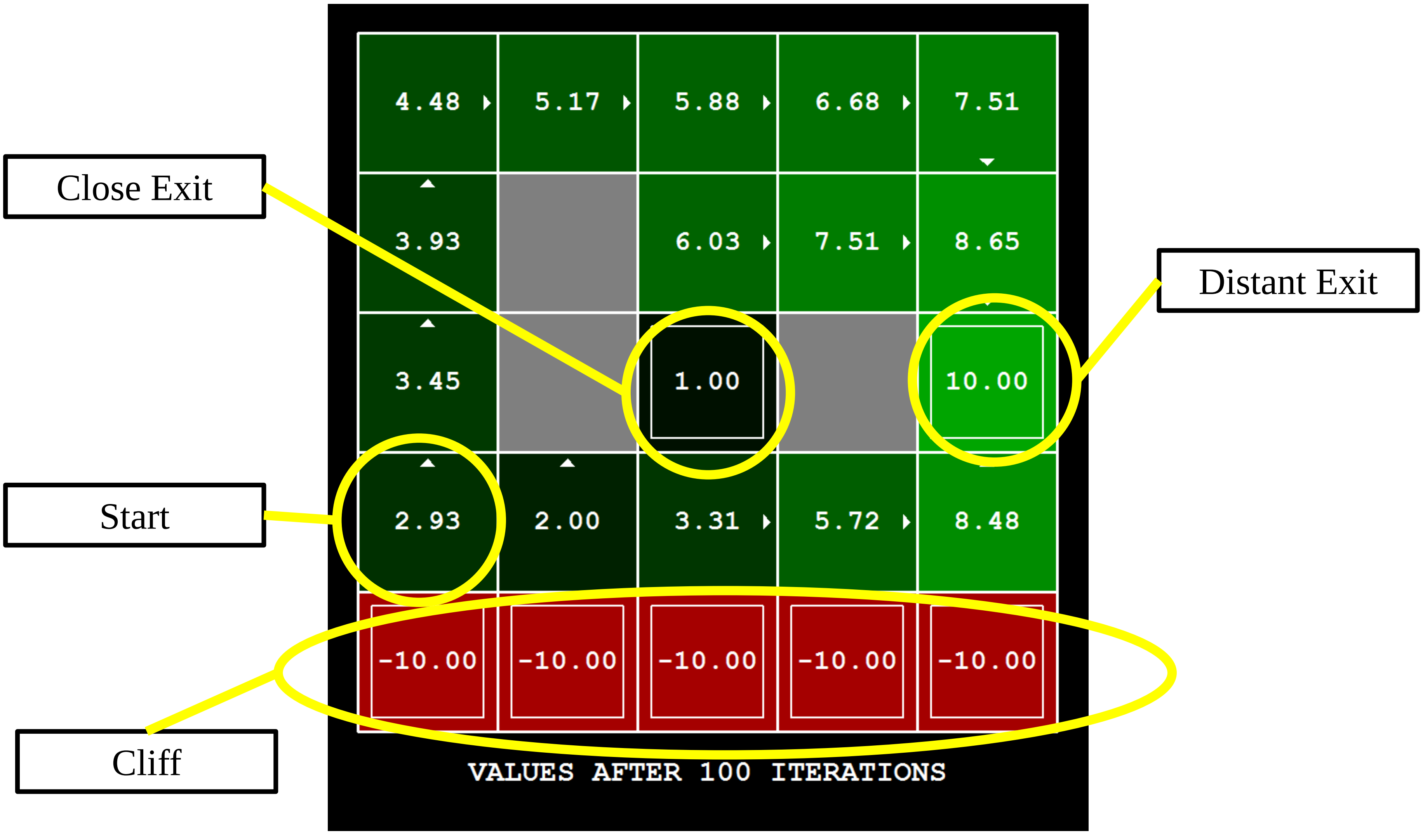
- Change only one of the parameters, the discount factor γ or the noise level, so that the agent will cross the bridge in the optimal policy
  - Noise level: the uncertainty of taking an action

    Ex: When noise=0, for any given state s and action a in $A(s)$, there will be one s' such that $P(s'|s,a)=1$; for any other state $s'' \neq s'$, it holds that $P(s''|s,a)=0$.
  - Discount factor: the level of importance of the future rewards
- The result should be something like this:

Question 3 – Value Iteration Agent

- Adjust the parameters, including the discount factor γ, the noise level, and the living reward, so that the agent acts as the descriptions
  - Living reward: The amount of reward given when the agent is still alive (i.e. doesn’t fall over the cliff)

Question 3 – Value Iteration Agent



Question 4~7 – Q Learning Agent

- Motivation: the transition probability and the reward of any given state are not known in advance.
- Construct a two dimensional (for states and actions) table to learn the utility of all states and the optimal policy.
  - One viable way to do this is to construct a “dictionary of dictionary” in python.

Question 4~7 – Q Learning Agent

- __init__(self, **args):

  Construct your Q table here.

- getQValue(self, state, action):

  If the state is already seen, return Qtable(state, action)

  Otherwise, construct a new 1D array in the Qtable and set all the elements to 0

    i.e.

    new Qtable(*state*, *)

    for all *action* in A(*state*) set Qtable(*state*, *action*)=0

Question 4~7 – Q Learning Agent

- getValue(self, state):

  If there are no legal actions, return 0

  Otherwise, return max $_{\textit{action} \text{ belongs to } A(\textit{state})}$ Qtable(*state*, *action*)

  – Note: Please be advised to use the function "getQValue" instead of directly accessing the data in the Qtable here.

- getPolicy(self, state):
- getAction(self, state):
- update(self, state, action, nextState, reward):

  Too simple to allow any hints…

Question 9 – Approximate Q Learning Agent

- Motivation: the original Q learning method is not scalable.
- Extract the features of the state-action pair and learn the "weights" of the features instead.
- You only have to initialize the weights (you can use util.Counter) and override two functions "getQValue" and "update" according to the equations in the html file. You might need to call the function "getFeatures" defined in "featureExtractors.py".