Assignment 4: Markov Decision Processes and Reinforcement Learning

CS486/686 – Winter 2014

Out: March 20, 2014

Due: April 4 (11:59 pm), 2014. Submit an electronic copy of your assignment via LEARN. Late assignments may be submitted within 24 hrs for 50% credit.

Be sure to include your name and student number with your assignment.

In this assignment you will implement value iteration and the Q-learning algorithm for the following simple grid-world problem.

S = start state

B = bad state
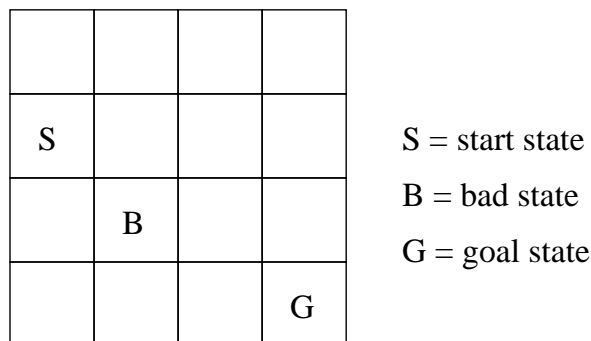
G = goal state

An agent starting in the start state S must reach the goal state G. At each time step, the agent can go *up*, *down*, *left* or *right*. However, the agent's movements are a bit noisy since it goes in the intended direction with a high probability a and in one of the two lateral directions with a low probability b. For instance, when executing the action *up*, the agent will indeed go up by one square with probability a, but may go left with probability b and right with probability b (here $a + b + b = 1$). Similarly, when executing the action *left*, the agent will indeed go left with probability a, but may go up with probability b and down with probability b, When an action takes the agent out of the grid world, the agent simply bounces off the wall and stays in its current location. For example, when the agent executes *left* in the start state it stays in the start state with probability a, it goes up with probability b and down with probability b. Similarly, when the agent executes *up* from the start state, it goes up with probability a, right with probability b and stays in the start state with probability b. Finally, when the agent is in the goal state, the task is over and the agent transitions to a special *end* state with probability 1 (for any action). This end state is *absorbing*, meaning that the agent cannot get out of the end state (i.e., it stays in the end state with probability 1 for every action).

The agent receives a reward of 100 when it reaches the goal state, -70 for the bad state and -1 for every other state, except the end state, which has a 0 reward. The agent's task is to find a policy to reach the goal state as quickly as possible, while avoiding the bad state.

In case you are not certain about the transition and reward model, I've put on the course webpage a file (grid-World.m) with a precise description (in Matlab) of the transition and reward models. Feel free to directly use this file if you program your assignment in Matlab or to port it to your favorite language otherwise. Tip: if you use Matlab, the implementation of value iteration and Q-learning in Questions 1 and 2 will be very short (i.e., a few lines of code).

1. **[50 pts]** Value iteration

   Compute the optimal policy by implementing the value iteration algorithm. Use a discount factor of 0.99 and run value iteration until the difference between two successive value functions is at most 0.01 (i.e., $|V_{t+1}(s) - V_t(s)| < 0.01 \; \forall s$). Run value iteration once with $a = 0.9$, $b = 0.05$ and a second time with $a = 0.8$, $b = 0.1$

   What to hand in:

   - A printout of your code.
   - The optimal policies and optimal value functions found for $a = 0.9$, $b = 0.05$ and for $a = 0.8$, $b = 0.1$.
   - Discuss the differences found in the optimal policies and value functions for the different combinations of a and b. Explain briefly how a and b impact the optimal policy.

2. **[50 pts]** Q-learning

   Assuming the transition and reward models are unknown, compute the optimal policy by Q-learning (a.k.a active temporal difference). In contrast to Question 1, the transition and reward models should only be used to *simulate* the environment when the agent executes an action. Use a transition model with $a = 0.9$ and $b = 0.05$, a discount factor of 0.99 and a learning rate of $\alpha = 1/N(s, a)$ where $N(s, a)$ is the number of times that action a was executed in state s. Always starting from the start state, run Q-learning for 10,000 episodes, where an episode consists of a sequence of moves from the start state until the end state is reached. Try two different ϵ-greedy exploration functions by setting ϵ to 0.05 and then to 0.2. In other words, when $\epsilon = 0.05$, select the optimal action with probability 0.95 and a random action with probability 0.05. Similarly, when $\epsilon = 0.2$, select the optimal action with probability 0.8 and a random action with probability 0.2.

   What to hand in:

   - A printout of your code.
   - The optimal policies and optimal value functions found for $\epsilon = 0.05$ and $\epsilon = 0.2$.
   - Discuss the impact of ϵ on the convergence of Q-learning. More specifically, discuss the impact on the **rate** of the convergence and the **policy** that it will eventually converge to.