CptS 570 Machine Learning, Fall 2017
Homework #5
Due Date: Dec 11 (Mon), 4pm

NOTE 1: Please use a word processing software (e.g., Microsoft word or Latex) to write your answers and drop a printed copy in my office (EME 133) during 4-5pm window on Dec 11.

NOTE 2: Please ensure that all the graphs are appropriately labeled (x-axis, y-axis, and each curve). The caption or heading of each graph should be informative and self-contained.

1. **(Finite-Horizon MDPs.)** Our basic definition of an MDP in class defined the reward function $R(s)$ to be a function of just the state, which we will call a *state reward function*. It is also common to define a reward function to be a function of the state and action, written as $R(s, a)$, which we will call a *state-action reward function*. The meaning is that the agent gets a reward of $R(s, a)$ when they take action a in state s. While this may seem to be a significant difference, it does not fundamentally extend our modeling power, nor does it fundamentally change the algorithms that we have developed.

   a) **(5 points)** Describe a real world problem where the corresponding MDP is more naturally modeled using a state-action reward function compared to using a state reward function.

   b) **(10 points)** Modify the Finite-horizon value iteration algorithm so that it works for state-action reward functions. Do this by writing out the new update equation that is used in each iteration and explaining the modification from the equation given in class for state rewards.

   c) **(10 points)** Any MDP with a state-action reward function can be transformed into an "equivalent" MDP with just a state reward function. Show how any MDP with a state-action reward function $R(s, a)$ can be transformed into a different MDP with state reward function $R(s)$, such that the optimal policies in the new MDP correspond exactly to the optimal policies in the original MDP. That is an optimal policy in the new MDP can be mapped to an optimal policy in the original MDP. *Hint: It will be necessary for the new MDP to introduce new "book keeping" states that are not in the original MDP.*

2. **(k-th Order MDPs.) (15 points)** A standard MDP is described by a set of states S, a set of actions A, a transition function T, and a reward function R. Where $T(s, a, s')$ gives the probability of transitioning to s' after taking action a in state s, and $R(s)$ gives the immediate reward of being in state s.

   A k-order MDP is described in the same way with one exception. The transition function T depends on the current state s and also the previous $k-1$ states. That is, $T(s_{k-1}, \cdots, s_1, s, a, s')$ $= Pr(s'|a, s, s_1, \cdots, s_{k-1})$ gives the probability of transitioning to state s' given that action a was taken in state s and the previous $k-1$ states were $(s_{k-1}, \cdots, s_1)$.

   Given a k-order MDP $M = (S, A, T, R)$ describe how to construct a standard (First-order) MDP $M' = (S', A', T', R')$ that is equivalent to M. Here equivalent means that a solution to M' can be easily converted into a solution to M. Be sure to describe S', A', T', and R'. Give a brief justification for your construction.

3. (**10 points**) Some MDP formulations use a reward function $R(s, a)$ that depends on the action taken in a state or a reward function $R(s, a, s')$ that also depends on the result state s' (we get reward $R(s, a, s')$ when we take action a in state s and then transition to s'). Write the Bellman optimality equation with discount factor β for each of these two formulations.

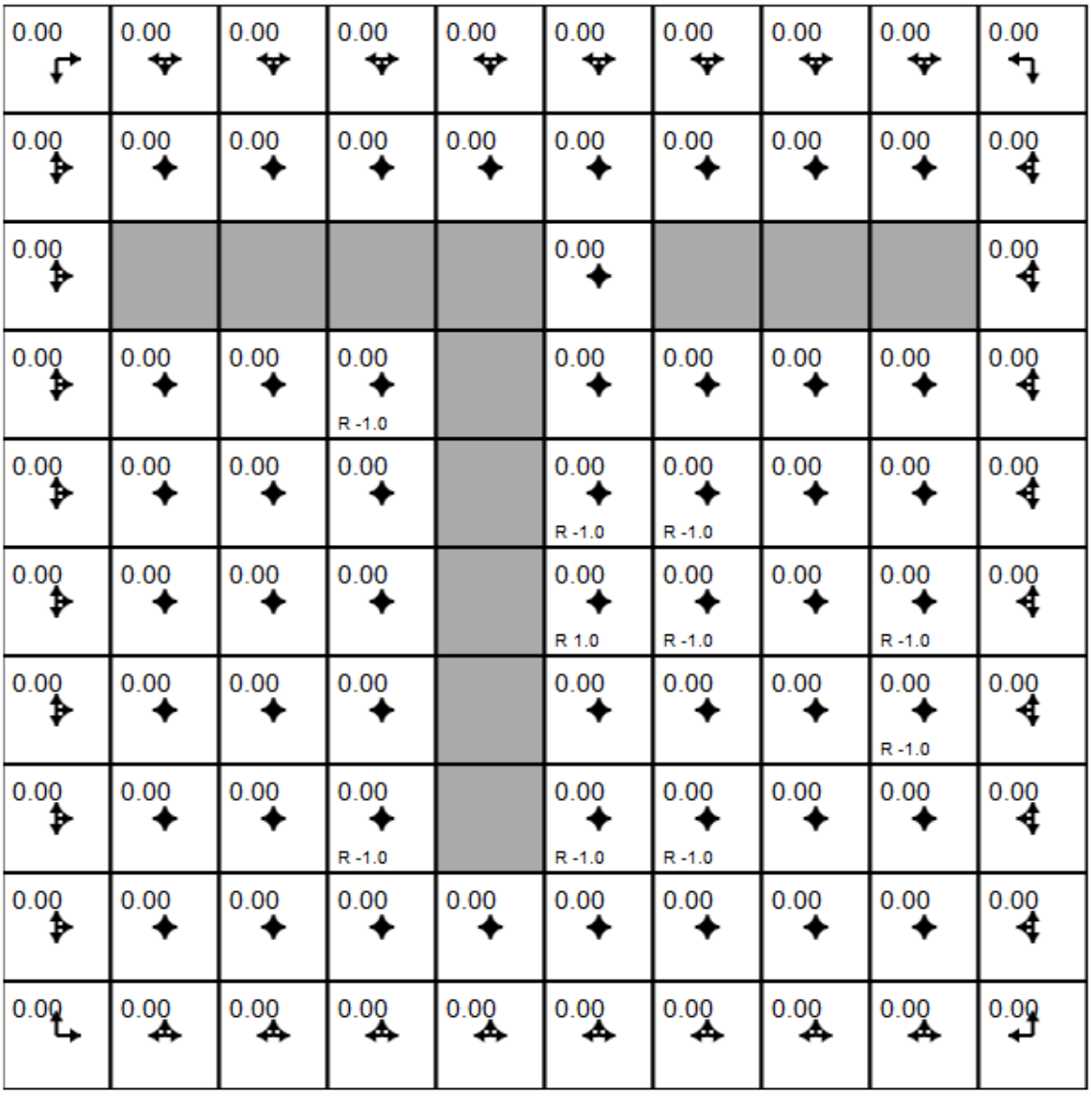

Figure 1: Grid world domain with states and rewards.

4. (**10 points**) Consider a trivially simple MDP with two states $S = \{s_0, s_1\}$ and a single action $A = \{a\}$. The reward function is $R(s_0) = 0$ and $R(s_1) = 1$. The transition function is $T(s_0, a, s_1) = 1$ and $T(s_1, a, s_1) = 1$. Note that there is only a single policy π for this MDP that takes action a in both states.

   a) Using a discount factor $\beta = 1$ (i.e. no discounting), write out the linear equations for evaluating the policy and attempt to solve the linear system. What happens and why?

   b) Repeat the previous question using a discount factor of $\beta = 0.9$.

5. (**30 points**) Implementation of Q-Learning algorithm and experimentation.

   You are given a Gridworld environment that is defined as follows:

   State space: GridWorld has 10x10 = 100 distinct states. The start state is the top left cell. The gray cells are walls and cannot be moved to.

   Actions: The agent can choose from up to 4 actions (left, right, up, down) to move around.

Environment Dynamics: GridWorld is deterministic, leading to the same new state given each state and action

Rewards: The agent receives +1 reward when it is in the center square (the one that shows R 1.0), and -1 reward in a few states (R -1.0 is shown for these). The state with +1.0 reward is the goal state and resets the agent back to start.

In other words, this is a deterministic, finite Markov Decision Process (MDP). Assume the discount factor β=0.9.

Implement the Q-learning algorithm (slide 46) to learn the Q values for each state-action pair. Assume a small fixed learning rate α=0.01.

Experiment with different explore/exploit policies:

1) ϵ-greedy. Try ϵ values 0.1, 0.2, and 0.3.

2) Boltzman exploration. Start with a large temperature value T and follow a fixed scheduling rate. Give these details in your report.

How many iterations did it take to reach convergence with different exploration policies?

Please show the converged Q values for each state-action pair.

6. (**10 points**) Automatic hyper-parameter tuning via Bayesian Optimization (BO). For HW2, you tuned the hyper-parameters of SVM classifier learning via grid search. For this homework, you need to use BO software to perform hyper-parameter search.

   You are provided with a training set, and testing set of multi-class classification examples (Please use *only the first fold data* from the first homework). Please divide the training data into sub-train (80 percent) and validation (20 percent) for your experiments. You will run a SVM classifier learning software on the training data to answer the following questions.

   Using a Gaussian kernel (-t 2 option), find the best hyper-parameters for SVM training. There are two hyper-parameters: C parameter (-c option) and γparameter (-g option). C and γ can take any value between 10^{-4} and 10^4.

   You will employ Bayesian Optimization (BO) software to automate the search for the best hyper-parameters by running it for 50 iterations. Plot the number of BO iterations on x-axis and performance of the best hyper-parameters at any point of time (performance of the corresponding trained SVM classifier on the validation data) on y-axis.

   Additionally, list the sequence of candidate hyper-parameters that were selected along the BO iterations.

   You can use one of the following BO softwares or others as needed.
   Spearmint: https://github.com/JasperSnoek/spearmint
   SMAC: http://www.cs.ubc.ca/labs/beta/Projects/SMAC/