CS533
Intelligent Agents and Decision Making
MDP Basics and Finite Horizon Problems

1. Construct a simple Markov Decision Process such that the optimal policy for maximizing finite-horizon total reward must be non-stationary. That is, for some specified horizon, the MDP should not have a stationary policy that maximizes the finite horizon total reward.

   Answer: Let the set of states be $S = \{s_1, s_2, s_3\}$ and actions $A = \{a_1, a_2\}$. Let the reward function be $R(s_1) = 1, R(s_2) = 0, R(s_3) = 3$. Finally, let the transition function be such that a_1 moves deterministically from s_1 to s_2 and from s_2 to s_3, and a_2 causes a self-transition in all states. That is, $T(s_1, a_1, s_2) = 1, T(s_2, a_1, s_3) = 1, T(s_3, a_1, s_3) = 1$, and for all s_i, $T(s_i, a_2, s_i) = 1$.

   It is easy to see that if there is just one stage to go, then the optimal policy at s_1 is to select a_2 and perform a self transition, giving a total reward of 2 (reward of one for the first step and then a reward of 1 for arriving in s_1 again). However, if there are two stages to go, then the optimal policy at s_1 is a_1 since there will be time to move from s_2 to s_3 at the next time step to get a total reward of 4 (one for initially being in s_1, then zero for going to s_2 and then three for going to s_3. Further we can see that when there are two stages to go, selecting a_2 in s_1 is sub-optimal, since there will no longer be enough time after doing so to reach s_3.

2. In many problems, not all actions are applicable in all states and many actions only lead to a small number of next states, compared to the total number of states. In this question, we consider how the complexity of finite-horizon value iteration and policy evaluation can be improved for such problems.

   To capture the notion of applicable actions, suppose that we have a function LEGAL(s) that takes a state s and returns the set of legal actions in s. Also suppose that we have a function NEXT(s, a), which takes a state s and action a as input and returns the set of states that have non-zero probability of occurring after taking a in state s. That is,

   $$\text{NEXT}(s, a) = \{s' \mid T(s, a, s') > 0\}.$$

   Assume that we are considering an MDP with n states and m actions such that for any state s and action a we have LEGAL(s) $\leq k$ and NEXT(s, a) $\leq r$. Assume that the time and space complexity of evaluating the functions NEXT and LEGAL are linear in the sizes of their output (i.e. the number of elements in their sets).

   (a) Describe how to modify the finite-horizon policy evaluation algorithm described in class, using one or both of the new functions, so that the time complexity is improved when $r < n$ and $k < m$. What is the time complexity? The time complexity should be expressed in terms of r and k when possible and may also involve n and m.

   Answer: We will abbreviate the set of states reachable from s via action a as $N(s, a)$. The basic step of policy evaluation was to compute $V_\pi^{k+1}(s)$ for a state given V_π^k. This was done via the equation:

   $$V_\pi^{k+1}(s) = R(s, \pi(s, k+1)) + \sum_{s' \in S} T(s, \pi(s, k+1), s') V_\pi^k(s').$$

   We can modify this calculation to only compute the summation for states s' that are in the set $N(s, \pi(s, k+1))$, rather than for all state in S giving:

   $$V_\pi^{k+1}(s) = R(s, \pi(s, k+1)) + \sum_{s' \in N(s, \pi(s, k+1))} T(s, \pi(s, k+1), s') V_\pi^k(s').$$

The complexity of this computation for a single state is bounded by r. We must perform the computation for each state and each time step so we get a final upper bound on the time complexity of hrn compared to hn^2. Note that we did not need to use LEGAL since for policy evaluation the action is selected by the policy under consideration at each state.

(b) Repeat part (a) but for the finite-horizon value iteration algorithm described in class.

Answer: Here the basic operation is the Bellman backup:

$$V^{k+1}(s) = \max_{a \in A} R(s,a) + \sum_{s' \in S} T(s,a,s') V^k(s').$$

We can restrict this to only consider states returned by NEXT and also only actions returned by LEGAL, which we will abbreviate by L as follows:

$$V^{k+1}(s) = \max_{a \in L(s)} R(s,a) + \sum_{s' \in N(s,a)} T(s,a,s') V^k(s').$$

So for each Bellman backup we must compute the max over at most k actions and for each max a sum over at most r states giving an upper bound on complexity of kr. This must be repeated for each state at each stage yielding $hnrk$ compared to hmn^2.

3. Our basic definition of an MDP in class defined the reward function $R(s,a)$ to be a function of the state and action, which we will call an *sa-reward function.* It is also common to define a reward function to be a function of the state s, the action a, and resulting state s', written as $R(s,a,s')$, which we will call a *sas-reward function.* The meaning is that the agent gets a reward of $R(s,a,s')$ when it enters state s' after taking action a in state s. While this may seem to be a significant difference, it does not fundamentally reduce the modeling power, nor does it fundamentally change the algorithms that we have developed.

(a) Give an example of a problem that is more naturally modelled using a sas-reward function compared to using an sa-reward function.

Solution: Consider a very simple "repeated gambler" MDP, which models a situation where a gambler repeatedly guesses the value of a coin flip and gets \$1 if the guess is correct and pays \$1 when the guess is not correct. This can be modeled with three states $S = \{s_0, s_t, s_h\}$ and two actions $A = \{\text{GuessHeads}, \text{GuessTails}\}$. State s_0 is the "guessing state" and when in s_0 for either action there is a 0.5 probability of reaching either s_t or s_h representing a flip of tails or heads respectively. The reward is +1 when the guess is correct and -1 reward when the guess is incorrect. When in state s_t or s_h each action results in a transition to s_0 with probability 1.0 and reward 0. That is we are reset to the guessing state. To model this reward function accurately we need to check whether or not the guess is the same as the next state or not. That is we need an sas-reward function of the form $R(s_0, \text{GuessHeads}, s_h) = R(s_0, \text{GuessTails}, s_t) = 1$, and $R(s_0, \text{GuessHeads}, s_t) = R(s_0, \text{GuessTails}, s_h) = -1$.

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

Solution: The original finite-horizon value iteration algorithm for sa-reward functions is given by:

$$\begin{aligned} V^0(s) &= 0 \\ V^{k+1}(s) &= \max_{a \in A} R(s,a) + \sum_{s' \in S} T(s,a,s') V^k(s') \end{aligned}$$

The updated equations for a sas-reward functions is as follows:

$$\begin{aligned} V^0(s) &= 0 \\ V^{k+1}(s) &= \max_{a \in A} \sum_{s' \in S} T(s,a,s') \left(R(s,a,s') + V^k(s') \right) \end{aligned}$$

The difference is that reward function $R(s,a,s')$ is put inside the expectation (i.e. summation), which is required since the reward function now depends on the resulting state s'.

(c) It turns out that for value functions defined in terms of expected reward, we can always create an equivalent sa-reward MDP for any sas-reward MDP. That is, for an sas-reward MDP M we can create an sa-reward MDP M' such that the value function for any policy π in M is equivalent to the value function in M'. This means that solving M' is equivalent to solving M. Describe how to perform this transformation from M to M'.

Solution: Consider an sas-reward MDP $M = (S, A, T, R)$. Using the same pattern as above we can write the finite horizon value of any policy π as:

$$V_\pi^{k+1}(s) = \sum_{s' \in S} T(s, \pi(s,k+1), s') \left(R(s, \pi(s,k+1), s') + V^k(s') \right)$$

. We can rewrite this by breaking up the sum as follows:

$$\begin{aligned} V_\pi^{k+1}(s) &= \sum_{s' \in S} T(s,\pi(s,k+1),s') R(s,\pi(s,k+1),s') + \sum_{s' \in S} T(s,\pi(s,k+1),s') V^k(s') \\ &= E_{s'} \left[R(s,a,s') \mid s, a = \pi(s,k+1) \right] + \sum_{s' \in S} T(s,\pi(s,k+1),s') V^k(s') \end{aligned}$$

where the expected value $E_{s'}[\cdot]$ is with respect to the random variable s'.

Now create a new sa-reward MDP $M' = (S, A, T, R')$ with $R'(s,a) = E_{s'} \left[R(s,a,s') \mid s, a \right]$. We know from class that the value function of π for M' is:

$$V_\pi^{k+1}(s) = R'(s, \pi(s,k+1)) + \sum_{s' \in S} T(s,\pi(s,k+1),s') V^k(s')$$

which we see is exactly equivalent to the value function of M above. Thus any policy has the same value under both M and M' as desired.

At first this might be surprising since it seemed that for examples such as the repeated gambler from above that sas-rewards were essential to defining the problem. However, while we have lost some ability to accurately model the exact details of the problem when we use just sa-rewards, the fact that we are only interested in expected reward means that it is valid to only think about the expected reward over next states s' when deciding what to do.

4. It is also common to define MDPs using state reward functions, or s-reward functions, where the reward function $R(s)$ gives the reward for being in state s. It turns out that nothing fundamental is lost by moving from sa-rewards to s-rewards. For the finite-horizon MDP setting, show how any MDP with an sa-reward function $R(s,a)$ can be transformed into a different, but "equivalent" MDP with an s-reward function $R(s)$. Here by equivalent, we mean that for any sa-reward MDP M and horizon H, we can construct an s-reward MDP M' and specify a horizon H' such that the optimal policy for M' can be mapped to an optimal policy for M. *Hint: It will be necessary for the new MDP to introduce new "book keeping" states that are not in the original MDP.*

   Solution: Given an MDP M with sa-reward function $R(s,a)$ and a horizon H we will create a new MDP M' with an s-reward function $R'(s)$ that is equivalent for a horizon of $H' = 2H$. The key idea is that we will introduce new "book keeping" states in M' that will keep track of the action that was just executed. Denote the state space, action set, and transition function, and reward function of M by S, A, T, and $R(s,a)$ respectively. The new state space S' of M' will contain all states in S along with a new set of states $\{q_{s,a} | s \in S, a \in A\}$. That is, there is a new state in S' named $q_{s,a}$ for each state-action pair of M. The transition function T' and reward function $R'(s)$ of M' are defined as follows:

$$\begin{aligned}
T'(s,a,q_{s,a}) &= 1, \text{for all } s \in S, a \in A \\
T'(q_{s,a},a',s') &= T(s,a,s'), \text{for all } s \in S, a \in A, a' \in A, s' \in S \\
R'(s) &= 0, \text{for all } s \in S \\
R'(q_{s,a}) &= R(s,a), \text{for all } s \in S, a \in A
\end{aligned}$$

   According to this definition when the agent is in a normal state $s \in S$ of M' and takes an action a, it gets zero reward and then deterministically transitions to state $q_{s,a}$, which is a memory state that indicates we just took a in s. In state $q_{s,a}$ we get the reward $R(s,a)$, which gives the same reward as if we had be in M and took a in s. When in $q_{s,a}$, for any action taken by the agent, it will transition to a regular state s' with transition probability given by $T(s,a,s')$. That is, the transition probability from $q_{s,a}$ is equal to the probability of going from s to s' after taking a in M.

   Thus, according to the above definition, when the agent is in state $s \in S$ of M' and takes action a followed by any other action, it will end up getting a reward of $R(s,a)$ over the two steps and transitioning to a state $s' \in S$ with the same probability as if the agent had taken a in s. In otherwords, we have simulated a single action in M via two actions in M', where the second action is arbitrary.

   Suppose that we want to solve M for a finite horizon of H. Then we can solve M' using a finite horizon of $2H$, noting that the solution at each time-to-go will alternative between being in one of the original states $s \in S$ and being in a new state $q_{s,a}$. Let $\pi'(s,t)$ be the non-stationary policy of M' with t steps to go. We can extract a non-stationary policy $\pi(s,t)$ for M by $\pi(s,t) = \pi'(s,2t)$. That is, we ignore the policy of M' at alternating time step.

5. **(k-th order MDPs.)** 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, $T(s_{k-1},\ldots,s_1,s,a,s') =$

$\Pr(s'|a, s, s_1, \ldots, s_{k-1})$ gives the probability of transitioning to state s' given that action a was taken in state s and the previous $k-1$ states were $(s_{k-1}, \ldots, s_1)$.

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.

Answer: The state space for M' is $S' = S^k$, so that each state in M' is a k-tuple of states in S. That is, each state in S' is of the form $(s, s_1, \ldots, s_{k-1})$ where each component is a state in S. The actions of M' are the same as those of M, i.e. $A' = A$. Intuitively each state $(s, s_1, \ldots, s_{k-1})$ of M' encodes the state s at the current time and the previous $k-1$ states. This is all the information needed to determine the distribution over next states. The reward function of M' is defined as $R'((s, s_1, \ldots, s_{k-1})) = R(s)$, which means that the reward in the new MDP only depends on the current state s of M. Finally the transition function of M' is defined as:

$$\begin{aligned} T'((s, s_1, \ldots, s_{k-1}), a, \vec{s}) &= \Pr(s'|a, s, s_1, \ldots, s_{k-1}), \text{if } \vec{s} = (s', s, s_1, \ldots, s_{k-2}) \\ &= 0, \text{otherwise} \end{aligned}$$

This definition of the transition function enforces that the history is maintained correctly after state transitions and that the new state s' has probability given by the k-th order model. In particular, there is zero transition probability of moving to a state that does not update the history correctly, which simply involves shifting the history in the current state by one step.

It is easy to verify that there is a one-to-one correspondence between sequences of states in M and M' that have non-zero probability of being generated by some policy. Further, the probability of those sequences under any policy is equal.

Given M', we can solve that MDP to get a policy π' over states of the form $(s, s_1, \ldots, s_{k-1})$. We can now define a policy π for M, which depends on the history of states as: $\pi(s, s_1, \ldots, s_{k-1}) = \pi'((s, s_1, \ldots, s_{k-1}))$.

Thus, we see that for any k-order MDP M there is an equivalent MDP M' in the sense described above. Note, however, that we did not remove the k-order dynamics for free. Rather, we needed to significantly increase the size of the state space. In particular $|S'| = |S|^k$ showing that the number of states in M' is exponential in k. Since standard planning algorithms for first-order MDPs are at least polynomial in the number of states, this shows that the computational complexity of this solution approach is exponential in k as well. In general, we cannot avoid this exponential dependence, though for a particular problem there may be special structure that could be analyzed to improve on the straightforward conversion approach described above.

6. Suppose that in a finite-horizon setting, we would like the reward function to depend on the time-to-go. That is, the reward function will be of the form $R(s, t)$, which says that we get reward $R(s, t)$ for being in state s when the time-to-go is t. Can finite-horizon value iteration be modified to take this reward function into account? If so, show how to modify the equations. If not, then give an argument why.

   Answer: Yes, finite-horizon value iteration can be modified to work for a non-stationary reward function $R(s, a, t)$. The modification to value iteration is trivial and given below:

$$V^0(s) = 0$$

$$V^{k+1}(s) \quad = \quad R(s,a,k+1) + \max_{a\in A} \sum_{s'\in S} T(s,a,s')V^k(s')$$

Here we can simply replaced the usual $R(s,a)$ in the Bellman Backup with $R(s,a,k+1)$. In fact, in the finite-horizon setting we could also allow the transition function to be non-stationary (i.e. depend on the time to go). A similar modification to the algorithm could be made.