CMPUT 366 — Assignment 4

Instructor: Russell Greiner
Due Date: Wednesday, 5 December 2007 at 6pm
(100 points, 12% of grade)

The following exercises are intended to further your understanding of Sequential Decision Processes, Game theory, Natural Language, Machine Learning
(From Chapters 17, 18, 21, 22)

Problem 1 *[5 points]* **Markov Decision Process – Foundation**
In class, we formulated an MDP using a reward function $R(s)$, that depends only on the current state s. In general, we can consider reward functions $R_2(s, a)$ that take both the state *and the action*, and again return a real value, or even $R_3(s, a, s')$ that also depend on the outcome state. Write down the Bellman equation for each of these formulations.

Problem 2 *[30 points]* **Markov Decision Process – Example**
The Markov Decision Process shown in Figure 1 has 4 states. Each state has 2 actions: L (left) and R (right). The reward function is: $R(S_1) = 1$, $R(S_2) = R(S_3) = 0$, and $R(S_4) = 3$.

Let $U^\pi(S_i)$ be the expected discounted sum of future rewards, starting from state S_i and following some policy π; recall this satisfies

$$U^\pi(S_i) \quad = \quad R(S_i) \ + \ \gamma \times \sum_j p_{i,j}^{\pi(s)} * U^\pi(s_j)$$

where p_{ij}^a represents the probability of a transition from state S_i to state S_j when following action a. Here, let the discount factor be $\gamma = 0.5$.

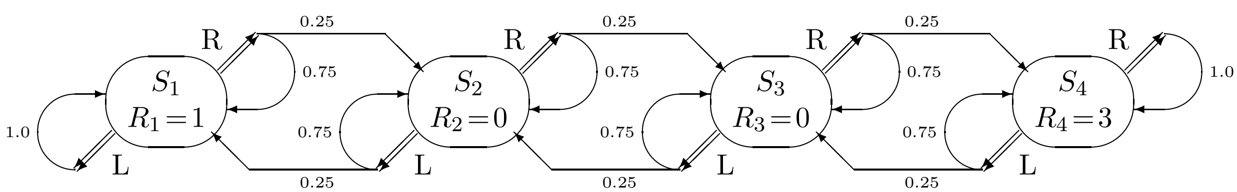

Figure 1: MDP: "4States"

a [2]: How many different possible policies are there?

b [5]: *Policy Valuation*
Assume we use the initial policy π_0 of always choosing action "L" — $\pi_0(S_1) = \pi_0(S_2) = \pi_0(S_3) = \pi_0(S_4) = L$eft. Write down four linear equations interrelating the values of $\{ U^{\pi_0}(S_i) \}_i$, then solve these equations.

c [5]: *Policy Improvement*
A new policy π_1 is computed by policy improvement, *i.e.*,

$$\pi_1(S_i) \quad = \quad \underset{a \in L,R}{\text{argmax}} \left\{ \ R(S_i) \ + \ \gamma \sum_j p_{ij}^a \times U^{\pi_0}(S_j) \ \right\} \tag{1}$$

What is the new policy — *i.e.*, what are the values for $\pi_1(S_i)$ for each $i = 1..4$?

For this new policy, compute the values, $U^{\pi_1}(S_i)$ for each $i = 1..4$.

d [5]: Let π_2 be the policy computed using Equation 1 *mutatis mutandis*; show the values of $\pi_2(S_i)$ and $U^{\pi_2}(S_i)$.

e [5]: Follow this iterative process one more time, to compute the π_3 policy with associated U^{π_3} values. Show these values.

f [3]: Is this policy the optimal policy? Why or why not?

g [5]: *Value Iteration*
Start with initial values $U_0(S_1) = U_0(S_2) = U_0(S_3) = U_0(S_4) = 0$. Show the values $U_1(S_i)$ after one value iteration; then the values $U_2(S_i)$ after a second value iteration. Then show the policy corresponding to U_2.

Problem 3 *[20 points] 2-player MDP*
Consider a two-player MDP that corresponds to a zero-sum turn-taking game (Chapter 6), with player X and Y. Let $R(s)$ be the reward for X. (Note the reward for Y is $-R(s)$.) Let $U_X(s)$ be the utility of state s when it is X's turn to move in s, and let $U_Y(s)$ be the utility of state s when it is Y's turn to move in s. Like rewards, all utilities are calculuted wrt X (just as we did for minimax game trees).

a [5]: Write down Bellman's euqations defining $U_X(s)$ and $U_Y(s)$.

b [5]: Explain how to do 2-player value iteration with these equations, and define a suitable stopping criterion.

c [5]: Now consider the following game. X and Y start as shown below:

X			Y
1	2	3	4

The players alternative moving, with player X moving first. On each move, the player moves his token to an open adjacent square, *in either direction.* If the opponent occupies an adjacent square, the player may jump over to the next open space, if any. (*E.g.*, if X is on 3 and Y is on 2, then X may jump back to 1.) The game ends when on e player reaches the opposite site — *i.e.*, when X reaches 4 or Y reaches 1. The value of the game (to X) is 1 if X reaches 4 first, and it is -1 if Y reaches 1 first.

Draw the state space (not the game tree), showing the moves by A as solid lines and moves by B as dashed lines. Mark each state s with $R(s)$. You will find it helpful to arrange the states (s_X, s_Y) on a 2-dimensional grid, using s_X and s_Y as coordinates.

d [5]: Now apply 2-player value iteration to solve this game, and derive the optimal policy.

Problem 4 *[15 points] Game Theory*
Solve 3-finger Morra: Each player, O and E, simultaneously holds out 1, 2 or 3 fingers — call these actions $s_O \in \{1, 2, 3\}$ and $s_E \in \{1, 2, 3\}$. If the sum $s_O + s_E$ is even, then player

E will win $s_O + s_E$ (and player O will loose that amount). If that sum is odd, then O will win $s_O + s_E$ and E will lose that amount.

Problem 5 *[10 points] Decision Tree Learning*
When constructing a decision tree, we may decide to stop at a node that has p positive examples and n negative examples. — perhaps because all of the attributes have been used.

a [5]: The obvious algorithm here would return $+$ if $p > n$, and $-$ otherwise. Show that this minimizes the total number of errors, over the set of examples that have reached this leaf.

b [5]: Alternatively, suppose the decision tree can return some probability $q = q(p, n) \in [0, 1]$. at this leaf. What value of q minimizes the sum of squared errors, over the examples. (Here, the tree should have returned "1" for each positive instance, which means the squared error for that instance is $(1 - q)^2$. Similarly, its squared error is $(0 - q)^2$ for each negative example.

Problem 6 *[10 points] Universal Set; tools from PAC learning*
A set $S = \{\mathbf{x}_1, \ \ldots, \ \mathbf{x}_m\}$ of binary d-tuples (*i.e.*, each $\mathbf{x}_k = \langle\, x_1^{(k)}, \ldots, x_d^{(k)} \,\rangle \ \in \{0,1\}^d$) is a (d, k)-*universal set* if, for every assignment to any subset of k variables, S includes an element that agrees with that assignment. That is, pick any of the $\binom{d}{k}$ size-k subsets of the d variables — call them $\{X_{i_1}, \ldots, X_{i_k}\}$ where each $i_j \in \{1, \ldots, d\}$ — and then pick any one of the 2^k assignments to these variables, say $t_{i_j} \in \{0, 1\}$ for each j. Then there is (at least) one element $\mathbf{x} \in S$ such that $x_{i_j} = t_{i_j}$ for all $j = 1..d$.

As an example, consider the set of $d = 4$ tuples:

$$S \quad = \quad \left[\begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{array} \right]$$

To be a $(4, 2)$-universal set, it would have to include all $2^2 = 4$ assignments to each of the $\binom{4}{2} = 6$ pairs, $\langle\, x_i, \ x_j \,\rangle$. Note that S does include all $2^2 = 4$ assignments to $\langle\, x_1, \ x_2 \,\rangle$ — *i.e.*, it includes $\langle\, x_1, \ x_2 \,\rangle = \langle\, 0, 0 \,\rangle$, $\langle\, 0, 1 \,\rangle$, $\langle\, 1, 0 \,\rangle$ and $\langle\, 1, 1 \,\rangle$. It also includes all 4 assignments to $\langle\, x_1, \ x_3 \,\rangle$, $\langle\, x_1, \ x_4 \,\rangle$, $\langle\, x_2, \ x_3 \,\rangle$, and $\langle\, x_3, \ x_4 \,\rangle$. However, this S is *not* a $(4, 2)$-universal set as it does not include every possible assignment to $\langle\, x_2, x_4 \,\rangle$: while it includes $\langle\, x_2, \ x_4 \,\rangle = \langle\, 0, 0 \,\rangle$ and $\langle\, 1, 1 \,\rangle$, it does *not* include either $\langle\, 0, 1 \,\rangle$ or $\langle\, 1, 0 \,\rangle$.

Now consider

$$S' \quad = \quad \left[\begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{array} \right]$$

and notice this S' is a $(4,2)$-universal set.

There are elaborate algorithms that are guaranteed to produce such (d,k)-universal sets. But how hard is it, really?

Suppose you just generate a set of $m(d,k)$ binary d-tuples, RANDOMLY — *i.e.*, each $x_i^{(k)}$ is drawn uniformly from $\{0,1\}$. How large does $m(d,k)$ have to be, to be $1-\delta$ confident that this set is a (d,k)-universal set?

[Hint: Just use Hoeffding's Bound, and don't worry too much about the constant :-) Also, you should expect this to be at least 2^k, for obvious reasons.]

Problem 7 *[10 points]* *Parsing*
[Russell/Norvig:Exercise 22.9, page 832]