HW04: Utility and Markov Decision Processes [*]

Alex Clemmer, u0458675

February 20, 2012

1 Utility and Expectation

1. Coin toss lottery. Let P be the random variable representing the amount of money won from the coin toss. Let $\mathbf{Pr}(F_H)$ stand for the probability of flipping heads, and $\mathbf{Pr}(G_H)$ stand for the probability of guessing heads.

   (a) If we assume that guessing heads or tails is equally likely, the probability of correctly guessing an already-flipped coin is simply 0.5; that is, $\mathbf{Pr}(G_H \cap F_H) + \mathbf{Pr}(G_T \cap F_T) = 0.25 + 0.25$.

   Thus, the expectation of payoff P is $\mathbf{E}[P] = \$3 \cdot \mathbf{Pr}(G_H \cap F_H) + \$3 \cdot \mathbf{Pr}(G_T \cap F_T) + \$0 \cdot \mathbf{Pr}(G_H \cap F_T) + \$0 \cdot \mathbf{Pr}(G_T \cap F_T) = \$\mathbf{1.5}$

   (b) The probabilities don't change, but the values do: $\mathbf{E}[P] = \$3 \cdot \mathbf{Pr}(G_H \cap F_H) + \$8 \cdot \mathbf{Pr}(G_T \cap F_T) + \$0 \cdot \mathbf{Pr}(G_H \cap F_T) + \$0 \cdot \mathbf{Pr}(G_T \cap F_T) = \$0 \cdot 0.25 + \$0 \cdot 0.25 + \$3 \cdot 0.25 + \$8 \cdot 0.25 = \$\mathbf{2.75}$

   (c) I'll assume the payouts are the same as in (b), though this is not explicitly stated. Then: $\mathbf{E}[P] = \$0 \cdot \mathbf{Pr}(G_T \cap F_H) + \$8 \cdot \mathbf{Pr}(G_T \cap F_T) + \$3 \cdot \mathbf{Pr}(G_H \cap F_H) + \$0 \cdot \mathbf{Pr}(G_T \cap F_H) = \$0 \cdot 0.5 + \$8 \cdot 0.5 + \$3 \cdot 0.0 + \$0 \cdot 0.0 = \$\ \mathbf{4}$.

2. The problem is a bit unclear, so I'll just assume that Tom pays Bob ahead of time, and then if Tom wins, Bob will pay back *double* what Tom initially paid — that is, Tom pays 100% of the wager ahead of time and then, if he is right, he recieves 200% of what he wagered, for a net total of 100% of the wager (and *not* 100% of what he wagered plus 200% of that wager).

   Further, I assume both that $\mathbf{Pr}(Roll_1) = \mathbf{Pr}(Roll_3)$, and that $\mathbf{Pr}(Roll_2) = \mathbf{Pr}(Roll_4)$ — that is to say, I assume that the probability of a number being even is equally distributed among even numbers (and the same for odd). Also note that $\mathbf{Pr}(A \cap B) = \mathbf{Pr}(A|B)P(B)$.

   Then the expectations are as follows:

   $\mathbf{E}[Guess_1] = 1 \cdot \mathbf{Pr}(Roll_1) \cdot \mathbf{Pr}(Guess_1|Roll_1) + -2 \cdot \mathbf{Pr}(Roll_2) \cdot \mathbf{Pr}(Guess_1|Roll_2) + -3 \cdot \mathbf{Pr}(Roll_3) \cdot \mathbf{Pr}(Guess_1|Roll_3) + -4 \cdot \mathbf{Pr}(Roll_4) \cdot \mathbf{Pr}(Guess_1|Roll_4) = 1(1/6)(1/4) + -2(1/3)(1/4) + -3(1/6)(1/4) + -4(1/3)(1/4) =$ **-7/12 dollars**

   $\mathbf{E}[Guess_2] = -1 \cdot \mathbf{Pr}(Roll_1) \cdot \mathbf{Pr}(Guess_1|Roll_1) + 2 \cdot \mathbf{Pr}(Roll_2) \cdot \mathbf{Pr}(Guess_1|Roll_2) + -3 \cdot \mathbf{Pr}(Roll_3) \cdot \mathbf{Pr}(Guess_1|Roll_3) + -4 \cdot \mathbf{Pr}(Roll_4) \cdot \mathbf{Pr}(Guess_1|Roll_4) = -1(1/6)(1/4) + 2(1/3)(1/4) + -3(1/6)(1/4) + -4(1/3)(1/4) =$ **-1/3 dollars**

[*] CS 5300 AI; Spring 2012 Instructor: Jur van den Berg, University of Utah

$\mathbf{E}[Guess_3] = -1 \cdot \mathbf{Pr}(Roll_1) \cdot \mathbf{Pr}(Guess_1|Roll_1) + -2 \cdot \mathbf{Pr}(Roll_2) \cdot \mathbf{Pr}(Guess_1|Roll_2) + 3 \cdot \mathbf{Pr}(Roll_3) \cdot \mathbf{Pr}(Guess_1|Roll_3) + -4 \cdot \mathbf{Pr}(Roll_4) \cdot \mathbf{Pr}(Guess_1|Roll_4) = -1(1/6)(1/4) + -2(1/3)(1/4) + 3(1/6)(1/4) + -4(1/3)(1/4) =$ **-5/12 dollars**

$\mathbf{E}[Guess_4] = -1 \cdot \mathbf{Pr}(Roll_1) \cdot \mathbf{Pr}(Guess_1|Roll_1) + -2 \cdot \mathbf{Pr}(Roll_2) \cdot \mathbf{Pr}(Guess_1|Roll_2) + -3 \cdot \mathbf{Pr}(Roll_3) \cdot \mathbf{Pr}(Guess_1|Roll_3) + 4 \cdot \mathbf{Pr}(Roll_4) \cdot \mathbf{Pr}(Guess_1|Roll_4) = -1(1/6)(1/4) + -2(1/3)(1/4) + -3(1/6)(1/4) + 4(1/3)(1/4) =$ **0 dollars**

3. N/A

4. Given all this, the actual calculation involves a fairly annoying application of the products of probabilities. Note that because the probability that the guess is some number and the probability of the roll is some number are not independent, we thus have to total all the cases somewhat tediously; consider, for example, the probability that Bob gains \$1 is totaled as $\mathbf{Pr}(Roll_3) \cdot \mathbf{Pr}(Guess_1|Roll_3) + \mathbf{Pr}(Roll_{even}) \cdot \mathbf{Pr}(Guess_1|Roll_{even})$, since when we roll 1 and guess 1, Bob pays Tom \$1. Also, remember, given the assumptions above, $\mathbf{Pr}(Roll_1) = \mathbf{Pr}(Roll_3) = 1/6$ and $\mathbf{Pr}(Roll_2) = \mathbf{Pr}(Roll_4) = 1/3$. I also assume that all guesses are equiprobable.

   The explicit equation is thus this monstrosity: $\mathbf{E}[P] = -\$1 \cdot \mathbf{Pr}(Roll_1) \cdot \mathbf{Pr}(Guess_1|Roll_1) + -\$2 \cdot \mathbf{Pr}(Roll_2) \cdot \mathbf{Pr}(Guess_2|Roll_2) + -\$3 \cdot \mathbf{Pr}(Roll_3) \cdot \mathbf{Pr}(Guess_3|Roll_3) + -\$4 \cdot \mathbf{Pr}(Roll_4) \cdot \mathbf{Pr}(Guess_4|Roll_4) + \$1 \cdot \mathbf{Pr}(Roll_3) \cdot \mathbf{Pr}(Guess_1|Roll_3) + \$1 \cdot \mathbf{Pr}(Roll_{even}) \cdot \mathbf{Pr}(Guess_1|Roll_{even}) + \$2 \cdot \mathbf{Pr}(Roll_{odd}) \cdot \mathbf{Pr}(Guess_2|Roll_{odd}) + \$2 \cdot \mathbf{Pr}(Roll_4) \cdot \mathbf{Pr}(Guess_2|Roll_4) + \$3 \cdot \mathbf{Pr}(Roll_1) \cdot \mathbf{Pr}(Guess_3|Roll_1) + \$3 \cdot \mathbf{Pr}(Roll_{even}) \cdot \mathbf{Pr}(Guess_3|Roll_{even}) + \$4 \cdot \mathbf{Pr}(\neg Roll_{odd}) \cdot \mathbf{Pr}(Guess_4|Roll_{odd}) + \$4 \cdot \mathbf{Pr}(\neg Roll_2) \cdot \mathbf{Pr}(Guess_4|Roll_2)$

   This all translates to this awful expected value calculation: $\mathbf{E}[P] = (1/6)(1/4)(-\$1) + (1/6)(1/4)(-\$3) + (1/3)(1/4)(-\$2) + (1/3)(1/4)(-\$4) + (1/6)(1/4)\$1 + 2(1/3)(1/4)\$1 + 2(1/6)(1/4)\$2 + (1/3)(1/4)\$2 + (1/6)(1/4)\$3 + 2(1/3)(1/4)\$3 + 2(1/6)(1/4)\$4 + (1/3)(1/4)\$4 = \$\mathbf{7/6} \approx \$\ \mathbf{1.16667}$

2 AIMA Problem 17.8

NOTE: I am assuming That r is not an exit node, since in the figure, it does not have a box around it.

1. Case: $r = 300$. The agent in this case really wants to move to the r cell because the reward is so incredibly high. Although the actor will not explicitly avoid going to the exit simply because it is an exit, in this case, the center cell is next to *two* nodes that lead directly to r, and so typically its utility will be higher than the exit node even though its reward is positive. Thus, even the node directly beneath it will not point directly at it. (For reference, I actually did run the algorithm to make sure.)

↑	←	←
↑	↑	←
↑	↑	↑

2. Case: $r = -3$. Unlike the first problem, the agent here wants to exit as quickly as possible, since the r cell provides no incentive to stick around. Again the center cell will be right next

to paths that lead directly to the exit (which is the highest-value cell), and therefore the cell beneath the r cell will point to it, because ultimately the incentive to point away from the r cell is not great enough to cause it to point away from the center cell. (I also ran this to make sure.)

→	→	↑
→	→	↑
→	↑	↑

3. Case: $r = 0$. Rhe cell directly beneath the r cell will actually point towards the r cell. It seems like the center cell, which is connected to *two* cells that point directly at the exit cell, would take precedence, but it does not provide the path of least resistance, and when you actually run it, the path generated goes through the r node. The rest is as one would expect it.

→	→	↑
↑	↑	↑
↑	↑	↑

4. Case: $r = 3$. Although an agent could theoretically stick around and try to stay inside the r cell, forever obtaining increase in score, it will not. If it has the oppotunity, it will go through the r cell, but it will then proceed towards the exit, which has the greatest payoff. This is because of the fact that it is an exit cell does not actually factor into the algorithm.

→	→	↑
↑	↑	↑
↑	↑	↑

3 Mission to Mars

1. There are 4 deterministic policies: (1) the *all-slow* policy, which goes slow no matter what; (2) the *all-fast* policy, which goes fast no matter what; and (3) the *alternating* policy, which starts out going fast in the cool state, and then slows down once it reaches the warm state.

   Note that the last policy is somewhat silly: (4) is an alternating policy as well, but instead of going fast in the cool state and slow in the warm state, it goes slow in the cool state and fast in the warm state. This is meaningless because going slow in the cool state transitions back to the cool state 100% of the time, so we never actually get to the warm state. But I treat this as a distinct policy because there is no other policy that gives the same recommendations over all the possible states.

2. *Much* higher than the warm state.

3. I chose to use the standard **Bellman Equation**-derived Value Iteration update rule (as noted in the book), rather than the equation in Dustin's notes, which comes from Wikipedia. They are totally equivalent, but for reference, the Bellman-derived rule looks like this: $U_{i+1} \leftarrow R(s) + \gamma \max\limits_{a \in A(s)} \sum_{s'} P(s'|a,s) U_i(s')$.

s	$V_0(s)$	$V_1(s)$	$V_2(s)$
cool	0	**10**	**19**
warm	0	**10**	**17.875**
off	0	0	0

4. Technically speaking, without a finite horizon, this will converge as part of an infinite series. We usually ameliorate this concern by picking some ϵ and stopping the iteration at that point.

5. After 7 iterations or so, the optimal policy begins to recommend *alternation*: if you're in the cool state, you should go fast, and if you're in the warm state, you should go slow.

s	$\pi(s)$
cool	fast
warm	slow

6. After 7 iterations, I get:

s	$V^*(s)$
cool	43.2834
warm	38.3154
off	0