CS 181 Assignment 5: Markov Decision Processes and Reinforcement Learning

Mark VanMiddlesworth & Shuang Wu

4/19/11

Problem 1.

(a) Given a probability function $P(\text{points} \mid \text{target})$ and utility function $U(\text{points}, \text{score})$, the optimal action maximizes the expected utility:

$$t_{\text{opt}} = \arg \max_{t \in \text{target}} \sum_{p \in \text{points}} P(p \mid t) U(p, \text{score}).$$

(b) While a player's score is still high, this greedy utility function does well in selecting the optimal strategy of getting high points in order to decrease the player's score as fast as possible since it prefers targets with the most expected points. However, when a player's move could place him within winning distance in the next move, this utility function might not result in optimal actions since it will still tell the player to try to get the maximum number of points this turn, which may limit the players options in the next turn and make future moves more difficult (i.e., forcing the player to score exactly 1 point in the next turn after hitting 60 this turn with a score of 61).

Problem 2.

(a) The states for the MDP are the possible scores at each throw, which is the set $\{s \mid s \in [0, \texttt{START\_SCORE}]\}$. The actions of the model consist of the choice of target, denoted by the wedge number and the region.

(b) The reward occurs only when we win the game (getting to state $s = 0$), so we define the reward

$$R(s) = \begin{cases} 1 & \text{if } s = 0, \\ 0 & \text{otherwise.} \end{cases}$$

The discount factor automatically makes winning sooner rather than later more preferable.

(d) We choose to use an infinite horizon policy since the problem does not have a finite horizon: we are not guaranteed to win in a finite number of steps since it is possible for our score to remain the same indefinitely due to repeated misses or hitting targets larger than the current score. The discount factor as mentioned before takes into account our preference of winning sooner rather than later, so we don't have to track the number of turns (like we would in the finite horizon case).

(e) The optimal policy resulting from value iteration with the small game is as follows

Current score	Target ring	Target wedge	Points for hitting target
9	3	3	9
8	5	4	8
7	4	2	2
6	5	3	6
5	1	1	5
4	4	4	4
3	4	3	3
2	4	2	2
1	4	1	1

These results are very intuitive, since we note that all scores from 1 through 9 except for 7 are possible with a single dart throw, and the optimal policy from value iteration always picks the target that yields exactly the current score to win whenever possible. In the case of the score being 7, value iteration has selected a target to score 2 points to move the state of the game away from 7, after which the player may win in one throw.

(f) When varying the discount factor γ from 0.1 to 0.9, the optimal policy essentially stays the same; at $\gamma = 0.8$ and higher, the policy for a score of 2 changes to ring 5 and wedge 1 instead of ring 4 wedge 2, and at $\gamma = 0.9$, the policy for a score of 7 changes to ring 4 wedge 1 from ring 4 wedge 2. However, across optimal policies, the point value of the target wedge remains equal to the current score value whenever possible, and in the case of a score of 7, chooses some wedge that will yield a score less than 7 in the next turn.

Problem 3.

(a) We know that an optimal policy π^* needs to satisfy the Bellman equations:

$$\pi^*(s) = \arg\max_{a \in A} \left[R(s,a) + \gamma \sum_{s'} P(s' \mid s, a) V^{\pi^*}(s') \right]$$

for every state s. The policy iteration algorithm updates the policy at every iteration by

$$\pi^{\text{new}}(s) = \arg\max_{a \in A} \left[R(s,a) + \gamma \sum_{s'} P(s' \mid s, a) V^{\text{old}}(s') \right],$$

where V^{old} is the value of each state calculated with the old policy π^{old} from the previous iteration. The algorithm terminates when the policy stops changing,

$$\pi^* = \pi^{\text{new}} = \pi^{\text{old}}.$$

Therefore, if policy iteration terminates, then we also have that $V^{\text{old}} = V^{\text{new}} = V^*$ (since the value function is deterministic with respect to the policy function), and we have exactly satisfied the Bellman conditions for an optimal stationary policy.

(b) Suppose we start with policy π^1 and after one iteration we have an updated policy π^2. Consider a change

in action which is improved in state $\hat{s}$, such that $V^{\pi^1}(\hat{s}) \leq Q^{\pi^1}(\hat{s}, \pi^2(\hat{s}))$. We will prove $V^{\pi^1}(\hat{s}) \leq V^{\pi^2}(\hat{s})$. We have

$$\begin{aligned}
V^{\pi^1}(\hat{s}) &\leq Q^{\pi^1}(\hat{s}, \pi^2(\hat{s})) \\
&= R(\hat{s}, \pi^2(\hat{s})) + \gamma \sum_{s'} P(s' \mid \hat{s}, \pi^2(\hat{s})) V^{\pi^1}(s') \\
&\leq R(\hat{s}, \pi^2(\hat{s})) + \gamma \sum_{s'} P(s' \mid \hat{s}, \pi^2(\hat{s})) Q^{\pi^1}(s', \pi^2(s')) \\
&= R(\hat{s}, \pi^2(\hat{s})) + \gamma \sum_{s'} P(s' \mid \hat{s}, \pi^2(\hat{s})) \left[R(s', \pi^2(s')) + \gamma \sum_{s''} P(s'' \mid s', \pi^2(s')) V^{\pi^1}(s'') \right] \\
&\leq R(\hat{s}, \pi^2(\hat{s})) + \gamma \sum_{s'} P(s' \mid \hat{s}, \pi^2(\hat{s})) \left[R(s', \pi^2(s')) + \gamma \sum_{s''} P(s'' \mid s', \pi^2(s')) Q^{\pi^1}(s'', \pi^2(s'')) \right] \\
&= R(\hat{s}, \pi^2(\hat{s})) + \gamma \sum_{s'} P(s' \mid \hat{s}, \pi^2(\hat{s})) R(s', \pi^2(s')) \\
&\qquad + \gamma^2 \sum_{s'} P(s' \mid \hat{s}, \pi^2(\hat{s})) \sum_{s''} P(s'' \mid s', \pi^2(s')) Q^{\pi^1}(s'', \pi^2(s'')) \\
&\cdots
\end{aligned}$$

where we repeatedly replace V^{π^1} with Q^{π^1} so that the resulting expression is always greater or equal to the previous. Each time we do the replacement, we get a R term and an expected value term with V^{π^1}; we can move the R term out of the sum as we did in the last line above. The first term is just the reward if we follow the new policy π^2 in state $\hat{s}$, and the second term is the discounted expected reward in the next time step from following π^2, so the right hand expression converges to $V^{\pi^2}(\hat{s})$, and giving us the desired inequality,

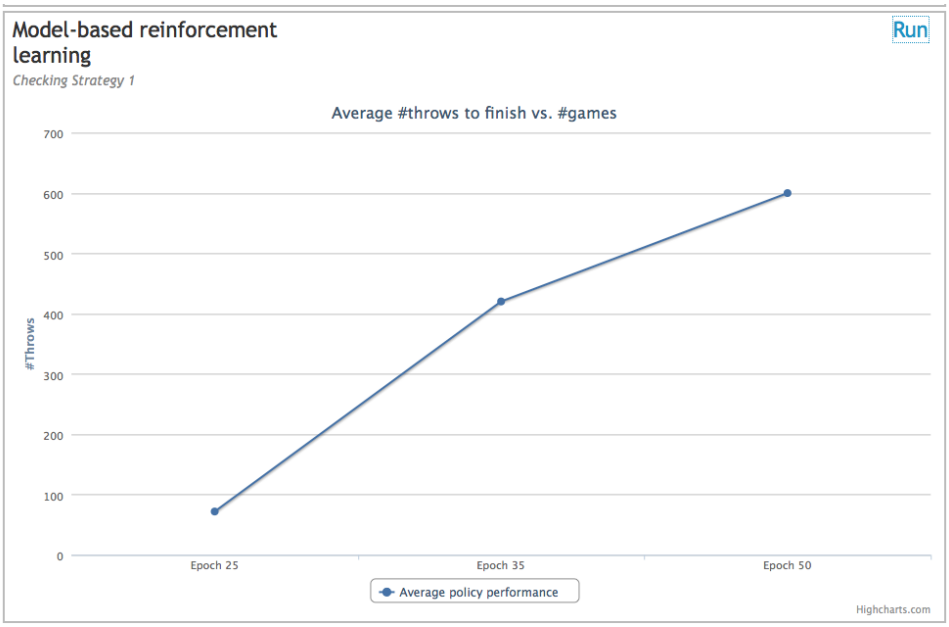
$$V^{\pi^1}(\hat{s}) \leq V^{\pi^2}(\hat{s}).$$

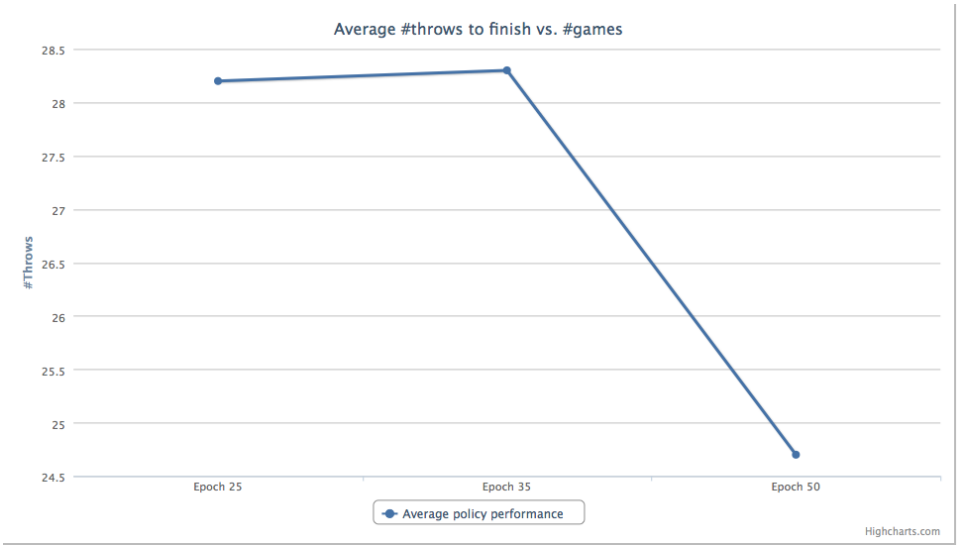
Since this derivation does not depend on Q^{π^1} only improving in state $\hat{s}$, and holds even if the new policy has improving actions in multiple states. Thus we see that it generalizes to V^{π^1} for all states $\hat{s}$, giving us the desired result.

(c) We have that $V^{\pi^{\text{new}}} = V^{\pi^{\text{old}}}$ if and only if $\pi^{\text{new}} = \pi^{\text{old}}$, and the algorithm stops when this condition is satisfied. The algorithm continues while $V^{\pi^{\text{new}}} \neq V^{\pi^{\text{old}}}$, and from part (b), we know that the value must monotonically improve at every step (if we have equality, then by part (a) we would have reached an optimal stationary policy by the terminating condition). Since there are finite actions and states, and hence a finite number of policies, the algorithm must terminate in a finite number of iterations with the optimal policy since the monotonicity guarantees that there are no cycles of policies.

Problem 4.

(a) We have the following charts:





Policy 1, where

$$P(\texttt{explore}) = \frac{\texttt{max\_score} - \texttt{current\_score}}{\texttt{max\_score}},$$

was terrible. However, we chose to include it because it was an interesting result from a relatively standard epsilon-greedy policy: becoming greedier towards the end of a game. This policy performs poorly in the darts game because the agent's early-game exploration tells it nothing about late-game behavior. Specifically, the agent will be greedy despite not have visited late-game states (i.e. scores close to zero), in other words, it will be acting greedily with no information. Even if the agent is learning over many games, it will never explore in the possible state-action pairs of the late game, so it will never learn about late-game states. Compare this, for example, to a soccer game, in which early-game exploration can be used to inform late game behavior. In a soccer game, late-game states will have been visited in the early

game, which is not the case in darts.

Policy 1 was so terrible that we should be wary of extrapolating from the negative correlation between epoch size and performance. However, a possible explanation is that the late-game agent will repeat the "best" action until the Q values are updated, because its probability of exploring is near zero. Because the "best" action is usually uninformed, and therefore terrible, it makes sense that a smaller epoch size would result in less repetition of undesirable late-game actions.

Policy 2,

$$P(\texttt{explore}) = \frac{\texttt{num\_games\_total} - \texttt{num\_games\_completed}}{\texttt{num\_games\_completed}},$$

was much better. In this strategy, the agent will explore both early- and late-game strategy during the first games, and will become greedier during the last games.

(b) A static learning rate, unless impractically small, does not guarantee convergence of the Q-learning algorithm, which requires that the learning rate be "eventually small enough." Therefore, we chose

$$\alpha(s, a) = \frac{1}{\#(s, a)}$$

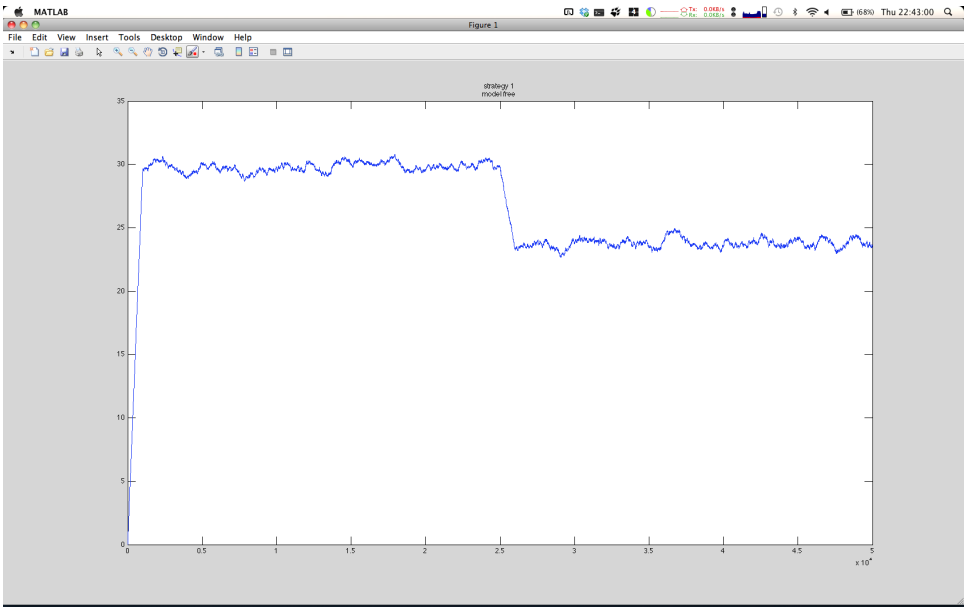
where $\#(s, a)$ represents the number of times this state/action pair has been encountered.

Exploration strategy 1 was a pure exploration phase followed by a pure exploitation phase.

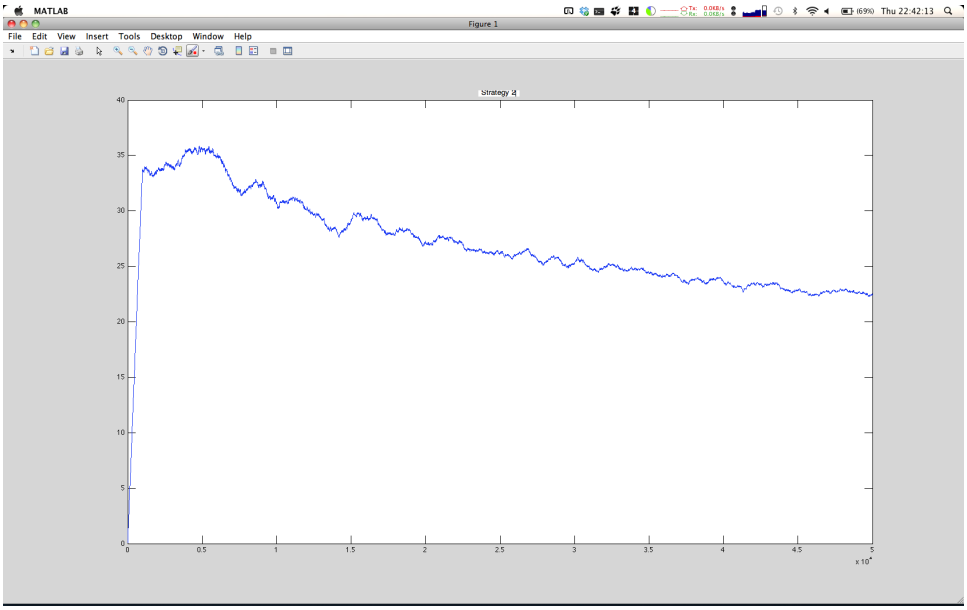
Exploration strategy 2 decayed from 100% exploration to 10% exploration over the number of games played. The first 1/10th of games were pure exploration, with the remainder of games decaying linearly from 100% to 10% exploration.

We also tested a pure exploration strategy and a pure exploitation strategy.

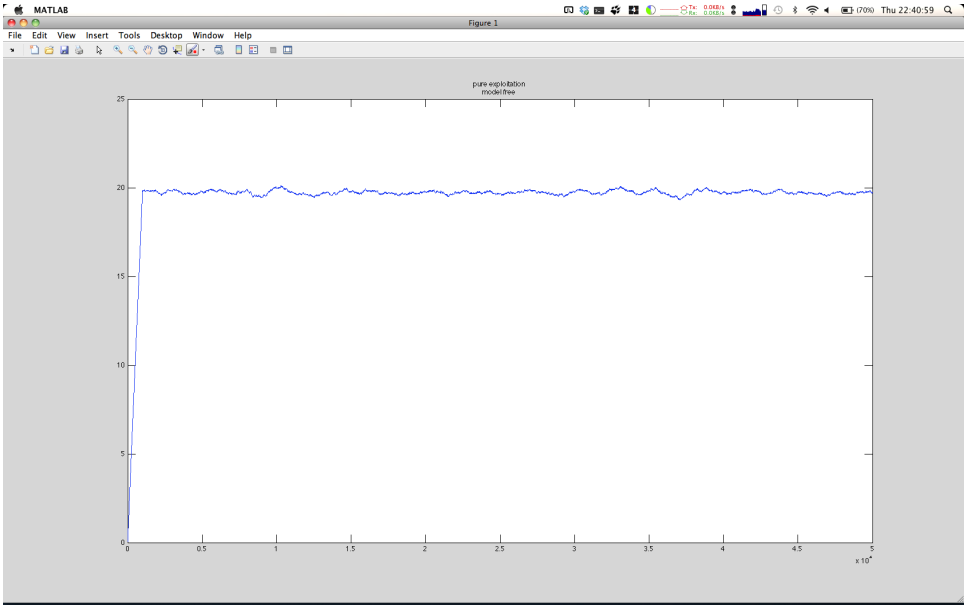
We have the following plots of average number of throws over 50,000 games, for model-free learner with policy 1:



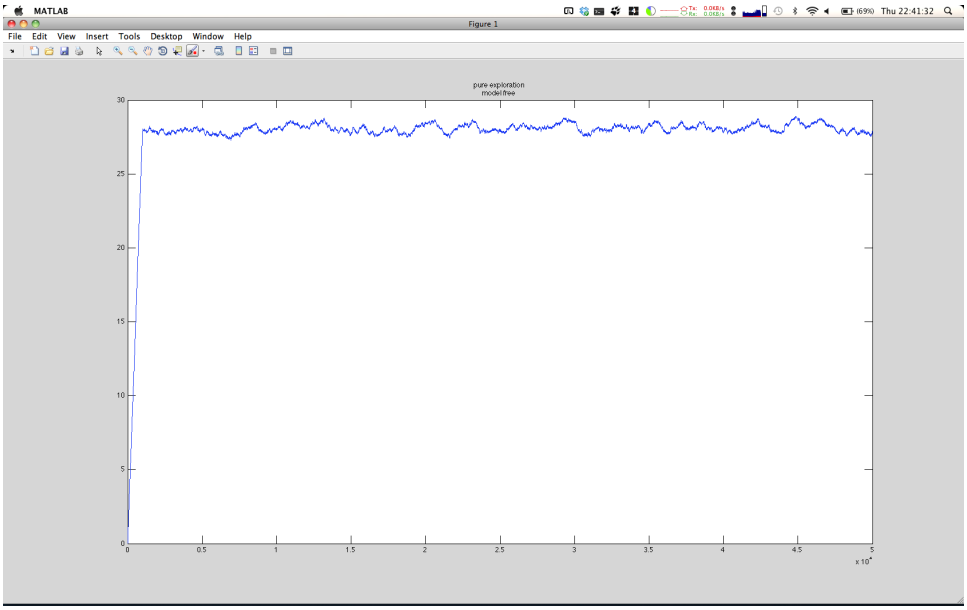
using policy 2:



with pure exploitation:



and with pure exploration:



(c) The most obvious difference between model-based and model-free learning was CPU speed. Whereas the model-based method took minutes to train on only 10 games, the model-free method could churn through hundreds of games per second. 10 games is not enough training data. The likelihood of even visiting all states, much less all state-action pairs, in 10 games is very low.

Running our model-free learner for 10 games produced average behavior between 25 and 32 throws per game. This is slightly worse than model-based learning (using the more reasonable of the exploration policies, #2). This is expected, as model-free learning generally requires more periods of experience to achieve the same policy as model-based learning.

However, in terms of CPU time, model-free learning was much faster, and achieved much better results given the same constraints. Our model-free learner was able to achieve very good results in under 30 seconds of CPU time. The plots below represent 50,000 training games on a medium-sized board. The results have been low-pass filtered with a window size of 1,000 games.

As expected, policy 1 obtains better results in the exploitation phase than in the exploration phase. Policy 2 gradually decays from exploration to exploitation. Pure exploitation performs the best out of all of the strategies, pure exploration, the worst. This is likely due to the long training time, over which even a greedy agent will likely access all relevant state-action pairs (i.e. all state-action pairs that occur in the optimal policy). It is interesting that policy 1 and policy 2 both converge to a policy with worse performance than the pure exploitation policy. However, this may be a result of the throwing model being randomly initialized.