Homework 1

Due: September 16, 2025 at 11:59pm

Instructions

- See `https://rpmml.github.io/homework_instructions/` for instructions on accessing the accompanying code and submitting homework via GradeScope.
- See `https://rpmml.github.io/policies` for additional policies regarding collaboration, LLMs, and late submissions.
- Use Piazza or attend office hours to ask questions about the homework.

Part 1: Written Exercises

Practice with Expectimax Search (15 points)

Consider the following MDP:

- The state space is $\mathcal{S} = \{1, 2, 3, \ldots, 1000\}$.
- The action space is $\mathcal{A} = \{\texttt{left}, \texttt{right}\}$.
- The reward function is $R(s, a, s') = -(s' - 531)^2$.
- The transition distribution is

s	a	$P(s' = s - 1)$	$P(s' = s)$	$P(s' = s + 1)$
$2 \leq s \leq 999$	`left`	0.8	0.1	0.1
$2 \leq s \leq 999$	`right`	0.1	0.1	0.8
$s = 1$	`left`	0.0	0.9	0.1
$s = 1$	`right`	0.0	0.1	0.9
$s = 1000$	`left`	0.9	0.1	0.0
$s = 1000$	`right`	0.1	0.9	0.0

- The horizon is finite: $H = 2$.

Draw a complete expectimax search tree that starts at $s = 530$. Make sure that all values are shown in the tree. Also annotate the optimal actions at each node. You can hand-draw the figure or use any software to produce it. *Hint: H counts actions, not states. So your tree should have depth 2 in actions.*

Pause to Ponder (No Points & No Submission Required)

Would anything have changed if the state space was 10x or 100x larger? What if we were using value iteration or policy iteration instead?

Pruning Low Probability Transitions (20 points)

We have seen *sparse sampling* as a way to deal with "large" transition distributions. As an alternative approach, your colleague Casey proposes to prune all low-probability transitions and then run expectimax search. Given an MDP with transition distribution $P(s' \mid s, a)$, their algorithm uses a hyperparameter $\epsilon \in (0, 1)$ to create a new transition distribution

$$\tilde{P}(s' \mid s, a) = \begin{cases} \frac{P(s' \mid s,a)}{Z(s,a)} & \text{if } P(s' \mid s, a) \geq \epsilon \\ 0 & \text{otherwise} \end{cases}$$

where $Z(s, a)$ is a normalization constant ensuring that $\sum_{s'} \tilde{P}(s' \mid s, a) = 1$. Discuss the advantages and disadvantages of Casey's algorithm versus sparse sampling. For full credit, provide at least one concrete MDP to support your arguments.

Pause to Ponder (No Points & No Submission Required)

How would Casey's algorithm change if we wanted to specify a maximum number of s' to consider for each (s, a), rather than specifying a probability threshold ϵ?

Part 2: Coding Exercises

Receding Horizon Control (10 points)

Show that receding horizon control (RHC) can be arbitrarily bad. Do this by implementing a general procedure that, given any lookahead horizon H and error threshold $\alpha \in [0, \infty)$, constructs a "pathological MDP" and initial state such that the optimal value of the initial state is at least α better than the value of the action selected by RHC. Complete the function `create_pathological_mdp_for_rhc` in `rhc_bad.py`. Examine the unit tests in `test_rhc_bad.py` to make sure you understand the question. No written answer required.

Pause to Ponder (No Points & No Submission Required)

What are some general strategies for overcoming the limitations of receding horizon control that you illustrated in this problem?

Expectimax Search: Another Implementation (20 points)

One implementation of expectimax search is provided in the `rpmml` codebase. In this exercise, you will complete an alternative implementation that produces the same output, but does so in a different way. Finish the function `iter_states` in `expectimax_alt.py`. Examine the unit tests in `test_expectimax_alt.py` to make sure you understand the question. No written answer required.

Ms. PacMan (20 points)

Implement the `MsPacmanAgent` in `pacman.py`. You can write any code at all—it can be very specific to Pacman (and specific to the unit tests), and it can use any or none of the code in the `rpmml` codebase. Examine the unit tests in `test_pacman.py` to make sure you understand the question. No written answer required. **Note: this is a challenging and very open-ended problem. Do the best that you can within a reasonable amount of time. You will get full credit for passing 7/11 tests.**

Pause to Ponder (No Points & No Submission Required)

What else might you have tried if you had more time?

Part 3: Your Turn to Teach

Create a Question (10-15 points)

Create a new question of any kind (writing, coding, etc.) that pertains to the material covered in this problem set. To receive full credit, you only need to write the question. To receive up to 5 points of extra credit, you should also provide a solution. Extra credit points will also be awarded based on creativity and effort. Please indicate in your answer (1) whether your question can be used (with modifications) in future versions of this course; and (2) whether you would like to be credited by name if your question is used.

Part 4: Feedback

General Course Feedback (5 points)

How can we improve the course in future years? Any and all feedback is welcome. Please comment on this problem set, lectures, and anything else. Some feedback is required for full credit. If you prefer to additionally submit anonymous feedback, please do so through the course website.