1 Pacman with Feature-Based Q-Learning

We would like to use a Q-learning agent for Pacman, but the size of the state space for a large grid is too massive to hold in memory. To solve this, we will switch to feature-based representation of Pacman’s state.

(a) We will have two features, $F_g$ and $F_p$, defined as follows:

\[
F_g(s, a) = A(s) + B(s, a) + C(s, a)
\]
\[
F_p(s, a) = D(s) + 2E(s, a)
\]

where

- $A(s) =$ number of ghosts within 1 step of state $s$
- $B(s, a) =$ number of ghosts Pacman touches after taking action $a$ from state $s$
- $C(s, a) =$ number of ghosts within 1 step of the state Pacman ends up in after taking action $a$
- $D(s) =$ number of food pellets within 1 step of state $s$
- $E(s, a) =$ number of food pellets eaten after taking action $a$ from state $s$

For this pacman board, the ghosts will always be stationary, and the action space is \{left, right, up, down, stay\}.

calculate the features for the actions $\in \{left, right, up, stay\}$

(b) After a few episodes of Q-learning, the weights are $w_g = -10$ and $w_p = 100$. Calculate the Q value for each action $\in \{left, right, up, stay\}$ from the current state shown in the figure.

calculate the features for the actions $\in \{left, right, up, stay\}$

(c) We observe a transition that starts from the state above, $s$, takes action up, ends in state $s'$ (the state with the food pellet above) and receives a reward $R(s, a, s') = 250$. The available actions from state $s'$ are down and stay. Assuming a discount of $\gamma = 0.5$, calculate the new estimate of the Q value for $s$ based on this episode.

(d) With this new estimate and a learning rate ($\alpha$) of 0.5, update the weights for each feature.
Consider the following gridworld (rewards shown on left, state names shown on right).

From state A, the possible actions are right(→) and down(↓). From state B, the possible actions are left(←) and down(↓). For a numbered state (G1, G2), the only action is to exit. Upon exiting from a numbered square we collect the reward specified by the number on the square and enter the end-of-game absorbing state X. We also know that the discount factor $\gamma = 1$, and in this MDP all actions are deterministic and always succeed.

Consider the following episodes:

<table>
<thead>
<tr>
<th>Episode 1 ($E_1$)</th>
<th>Episode 2 ($E_2$)</th>
<th>Episode 3 ($E_3$)</th>
<th>Episode 4 ($E_4$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s$</td>
<td>$a$</td>
<td>$s'$</td>
<td>$r$</td>
</tr>
<tr>
<td>$A$</td>
<td>↓</td>
<td>$G_1$</td>
<td>0</td>
</tr>
<tr>
<td>$G_1$ exit</td>
<td>$X$</td>
<td>10</td>
<td>$G_2$ exit</td>
</tr>
</tbody>
</table>

(a) Consider using temporal-difference learning to learn $V(s)$. When running TD-learning, all values are initialized to zero.

For which sequences of episodes, if repeated infinitely often, does $V(s)$ converge to $V^*(s)$ for all states $s$?

(Assume appropriate learning rates such that all values converge.)

Write the correct sequence under "Other" if no correct sequences of episodes are listed.

- $E_1, E_2, E_3, E_4$
- $E_1, E_2, E_1, E_2$
- $E_1, E_2, E_3, E_1$
- $E_4, E_4, E_4, E_4$
- $E_4, E_3, E_2, E_1$
- $E_3, E_4, E_3, E_4$
- $E_1, E_2, E_4, E_1$
- $E_1, E_2, E_4, E_4$

□ Other
(b) Consider using Q-learning to learn $Q(s, a)$. When running Q-learning, all values are initialized to zero. For which sequences of episodes, if repeated infinitely often, does $Q(s, a)$ converge to $Q^*(s, a)$ for all state-action pairs $(s, a)$?

(Assume appropriate learning rates such that all Q-values converge.)
Write the correct sequence under “Other” if no correct sequences of episodes are listed.

- $E_1, E_2, E_3, E_4$
- $E_1, E_2, E_1, E_2$
- $E_1, E_2, E_3, E_1$
- $E_4, E_4, E_4, E_4$
- $E_4, E_3, E_2, E_1$
- $E_3, E_4, E_3, E_4$
- $E_1, E_2, E_4, E_1$
- Other

□ Other