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 $\in \{left, right, up, stay\}$

\[
egin{align*}
F_p(s, up) &= 1 + 2(1) = 3 \\
F_p(s, left) &= 1 + 2(0) = 1 \\
F_p(s, right) &= 1 + 2(0) = 1 \\
F_p(s, stay) &= 1 + 2(0) = 1 \\
F_g(s, up) &= 2 + 0 + 0 = 2 \\
F_g(s, left) &= 2 + 1 + 1 = 4 \\
F_g(s, right) &= 2 + 1 + 1 = 4 \\
F_g(s, stay) &= 2 + 0 + 2 = 4
\end{align*}
\]
(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 $\in \{\text{left, right, up, stay}\}$ from the current state shown in the figure.

\[
Q(s, \text{up}) = w_p F_p(s, \text{up}) + w_g F_g(s, \text{up}) = 100(3) + (-10)(2) = 280
\]
\[
Q(s, \text{left}) = w_p F_p(s, \text{left}) + w_g F_g(s, \text{left}) = 100(1) + (-10)(4) = 60
\]
\[
Q(s, \text{right}) = w_p F_p(s, \text{right}) + w_g F_g(s, \text{right}) = 100(1) + (-10)(4) = 60
\]
\[
Q(s, \text{stay}) = w_p F_p(s, \text{stay}) + w_g F_g(s, \text{stay}) = 100(1) + (-10)(4) = 60
\]

(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.

\[
Q_{\text{new}}(s, a) = R(s, a, s') + \gamma \max_{a'} Q(s', a')
\]
\[
= 250 + 0.5 \max\{Q(s', \text{down}), Q(s', \text{stay})\}
\]
\[
= 250 + 0.5 \cdot 0
\]
\[
= 250
\]

where

\[
Q(s', \text{down}) = w_p F_p(s', \text{down}) + w_g F_g(s', \text{down}) = 100(0) + (-10)(2) = -20
\]
\[
Q(s', \text{stay}) = w_p F_p(s', \text{stay}) + w_g F_g(s', \text{stay}) = 100(0) + (-10)(0) = 0
\]

(d) With this new estimate and a learning rate ($\alpha$) of 0.5, update the weights for each feature.

\[
w_p = w_p + \alpha \cdot (Q_{\text{new}}(s, a) - Q(s, a)) \cdot F_p(s, a) = 100 + 0.5 \cdot (250 - 280) \cdot 3 = 55
\]
\[
w_g = w_g + \alpha \cdot (Q_{\text{new}}(s, a) - Q(s, a)) \cdot F_g(s, a) = -10 + 0.5 \cdot (250 - 280) \cdot 2 = -40
\]
2 Q-learning

Consider the following gridworld (rewards shown on left, state names shown on right).

<table>
<thead>
<tr>
<th>Rewards</th>
<th>State names</th>
</tr>
</thead>
<tbody>
<tr>
<td>+10</td>
<td>A</td>
</tr>
<tr>
<td>+1</td>
<td>B</td>
</tr>
<tr>
<td></td>
<td>G1</td>
</tr>
<tr>
<td></td>
<td>G2</td>
</tr>
</tbody>
</table>

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:

- **Episode 1 (E1)**
  - Transition: \( A \xrightarrow{↓} G1 \) 0
  - Reward: +10

- **Episode 2 (E2)**
  - Transition: \( B \xrightarrow{↓} G2 \) 0
  - Reward: +1

- **Episode 3 (E3)**
  - Transition: \( A \xrightarrow{→} B \) 0
  - Transition: \( B \xrightarrow{↓} G2 \) 0

- **Episode 4 (E4)**
  - Transition: \( B \xleftarrow{←} A \) 0

(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 \( s \)?

(Write the correct sequence under “Other” if no correct sequences of episodes are listed.)

- \( E1, E2, E3, E4 \)
- \( E4, E3, E2, E1 \)
- \( E1, E2, E1, E2 \)
- \( E1, E2, E3, E1 \)
- \( E1, E2, E4, E4 \)
- \( E4, E4, E4, E4 \)
- \( E1, E1, E2, E4, E1 \)

Other: See explanation below

TD learning learns the value of the executed policy, which is \( V^\pi(s) \). Therefore for \( V^\pi(s) \) to converge to \( V^*(s) \), it is necessary that the executing policy \( \pi(s) = \pi^*(s) \).

Because there is no discounting since \( \gamma = 1 \), the optimal deterministic policy is \( \pi^*(A) = \downarrow \) and \( \pi^*(B) = \leftarrow \) (\( \pi^*(G1) \) and \( \pi^*(G2) \) are trivially exit because that is the only available action). Therefore episodes \( E1 \) and \( E4 \) act according to \( \pi^*(s) \) while episodes \( E2 \) and \( E3 \) are sampled from a suboptimal policy.

From the above, TD learning using episode \( E4 \) (and optionally \( E1 \)) will converge to \( V^*(s) = V^*(s) \) for states \( A, B, G1 \). However, then we never visit \( G2 \), so \( V(G2) \) will never converge. If we add either episode \( E2 \) or \( E3 \) to ensure that \( V(G2) \) converges, then we are executing a suboptimal policy, which will then cause \( V(B) \) to not converge. Therefore none of the listed sequences will learn a value function \( V^\pi(s) \) that converges to \( V^*(s) \) for all states \( s \). An example of a correct sequence would be \( E2, E4, E4, E4, \ldots \); sampling
$E_2$ first with the learning rate $\alpha = 1$ ensures $V^\pi(G_2) = V^*(G_2)$, and then executing $E_4$ infinitely after ensures the values for states $A$, $B$, and $G_1$ converge to the optimal values.

We also accepted the answer such that the value function $V(s)$ converges to $V^*(s)$ for states $A$ and $B$ (ignoring $G_1$ and $G_2$). TD learning using only episode $E_4$ (and optionally $E_1$) will converge to $V^\pi(s) = V^*(s)$ for states $A$ and $B$, therefore the only correct listed option is $E_4, E_4, E_4, E_4$. 
(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 ________

For $Q(s, a)$ to converge, we must visit all state action pairs for non-zero $Q^*(s, a)$ infinitely often. Therefore we must take the exit action in states $G_1$ and $G_2$, must take the down and right action in state $A$, and must take the left and down action in state $B$. Therefore the answers must include $E_3$ and $E_4$. 