Q1. 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 $\{\text{left, right, up, down, stay}\}$.

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

- $F_p(s, up) = 1 + 2(1) = 3$
- $F_p(s, left) = 1 + 2(0) = 1$
- $F_p(s, right) = 1 + 2(0) = 1$
- $F_p(s, stay) = 1 + 2(0) = 1$
- $F_g(s, up) = 2 + 0 + 0 = 2$
- $F_g(s, left) = 2 + 1 + 1 = 4$
- $F_g(s, right) = 2 + 1 + 1 = 4$
- $F_g(s, stay) = 2 + 0 + 2 = 4$
(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.

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

(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.

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

where

\[
Q(s', down) = w_p F_p(s, down) + w_g F_g(s, down) = 100(0) + (-10)(2) = -20
\]
\[
Q(s', stay) = w_p F_p(s, stay) + w_g F_g(s, stay) = 100(0) + (-10)(0) = 0
\]
(d) With this new estimate and a learning rate ($\alpha$) of 0.5, update the weights for each feature.

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

2 Feature-Based Q-Learning

1. When using features to represent the Q-function is it guaranteed that the feature-based Q-learning finds the same optimal $Q^*$ as would be found when using a tabular representation for the Q-function?

No, if the optimal Q-function $Q^*$ cannot be represented as a weighted combination of features, then the feature-based representation would not have the expressive power to find it. For example, consider the following MDP with deterministic transitions:

```
A  1  B  1  G
```

With discount $\gamma = 1$, the optimal $Q$ values are $Q(A, right) = 2, Q(B, right) = 1, Q(G, stay) = 0$.

Suppose we have just one feature $f$, which depends only on states, with value $f(A) = 1, f(B) = 2, f(G) = 0$. There’s no linear function that can map the feature values for the states to the optimal $Q$ values above, so it’s not possible for feature-based Q-learning to find the optimal values.
Q3. MDPs and RL

Recall that in approximate Q-learning, the Q-value is a weighted sum of features: $Q(s, a) = \sum_i w_i f_i(s, a)$. To derive a weight update equation, we first defined the loss function $L_2 = \frac{1}{2}(y - \sum_k w_k f_k(x))^2$ and found $dL_2/dw_m = -(y - \sum_k w_k f_k(x)) f_m(x)$. Our label $y$ in this set up is $r + \gamma \max_a Q(s', a')$. Putting this all together, we derived the gradient descent update rule for $w_m$ as $w_m \leftarrow w_m + \alpha (r + \gamma \max_a Q(s', a') - Q(s, a)) f_m(s, a)$.

In the following question, you will derive the gradient descent update rule for $w_m$ using a different loss function:

$$L_1 = \left| y - \sum_k w_k f_k(x) \right|$$

(a) Find $dL_1/dw_m$. Ignore the non-differentiable point.

Note that the derivative of $|x|$ is $-1$ if $x < 0$ and $1$ if $x > 0$. So for $L_1$, we have:

$$\frac{\partial L_1}{\partial w_m} = \begin{cases} -f_m(x) & \text{if } y - \sum_k w_k f_k(x) > 0 \\ f_m(x) & \text{if } y - \sum_k w_k f_k(x) < 0 \end{cases}$$

(b) Write the gradient descent update rule for $w_m$, using the $L_1$ loss function.

$$w_m \leftarrow w_m - \alpha \frac{\partial L_1}{\partial w_m} \leftarrow \begin{cases} w_m + \alpha f_m(x) & \text{if } y - \sum_k w_k f_k(x) > 0 \\ w_m - \alpha f_m(x) & \text{if } y - \sum_k w_k f_k(x) < 0 \end{cases}$$

4 Probability

Use the probability table to calculate the following values:

<table>
<thead>
<tr>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$X_3$</th>
<th>$P(X_1, X_2, X_3)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.05</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0.4</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0.1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0.05</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0.2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.0</td>
</tr>
</tbody>
</table>

1. $P(X_1 = 1, X_2 = 0) = 0.15$
2. $P(X_3 = 0) = 0.65$
3. $P(X_2 = 1 | X_3 = 1) = 0.2/0.35$
4. $P(X_1 = 0 | X_2 = 1, X_3 = 1) = 1$
5. $P(X_1 = 0, X_2 = 1 | X_3 = 1) = 0.2/0.35$