Sample Quiz on Function Approximation and Approximate Dynamic Programming

Instructor: Prof. XYZ

Instructions

Answer the following questions. For computational questions, show all steps clearly.

Problems

Q1. **Conceptual: Function Approximation in RL**
What is the role of function approximation in reinforcement learning, and why is it necessary for large-scale problems? Provide examples of situations where exact solutions are impractical.

Solution:
Function approximation in reinforcement learning is used to generalize from limited data, allowing the agent to handle large state and action spaces where exact solutions (like tabular methods) are impractical. It enables the agent to approximate value functions or policies when the state-action space is too large to store or compute exactly. Examples include robotics, where there are continuous states (e.g., positions, velocities) and games like Go, which have an enormous number of possible states.

Q2. **Computational: Linear Value Function Approximation**
Consider an environment with three states $S = \{s_1, s_2, s_3\}$ and a feature vector $\phi(s) = [\phi_1(s), \phi_2(s)]$. Assume that for state s_1, the feature vector is $\phi(s_1) = [1, 0]$, for s_2 it is $\phi(s_2) = [0, 1]$, and for s_3 it is $\phi(s_3) = [1, 1]$. The value function is approximated as a linear function $V(s; \theta) = \theta^T \phi(s)$, where $\theta = [\theta_1, \theta_2]$. Given $\theta_1 = 2$ and $\theta_2 = 3$, compute the approximate values of $V(s_1)$, $V(s_2)$, and $V(s_3)$.

Solution:
The approximate value function is given by:

$$V(s; \theta) = \theta_1 \phi_1(s) + \theta_2 \phi_2(s)$$

For s_1:

$$V(s_1) = 2 \cdot 1 + 3 \cdot 0 = 2$$

For s_2:

$$V(s_2) = 2 \cdot 0 + 3 \cdot 1 = 3$$

For s_3:

$$V(s_3) = 2 \cdot 1 + 3 \cdot 1 = 5$$

Q3. **Conceptual: Stochastic Gradient Descent in RL**
Explain how stochastic gradient descent (SGD) is used for value function approximation in reinforcement learning. What are the key advantages and challenges of using SGD?

Solution:
Stochastic gradient descent (SGD) is used to update the parameters of the value function approximation by minimizing the error between the predicted value and the target (e.g., the return or TD target). The update rule for SGD is:

$$\theta \leftarrow \theta + \alpha \left(\text{Target} - V(s;\theta)\right) \nabla_\theta V(s;\theta)$$

Advantages of SGD include its efficiency for large datasets and ability to handle online learning. However, it can be noisy (since updates are based on individual samples) and may converge slowly or get stuck in local minima.

Q4. **Computational: Gradient Descent Update for Value Function Approximation**
Using the same feature vectors and parameters from Problem 2, assume that the true value for state s_1 is 3. Perform one step of gradient descent with learning rate $\alpha = 0.1$ to update the parameter θ. The current approximation is $V(s_1) = 2$.

Solution:
The gradient descent update rule is:

$$\theta_i \leftarrow \theta_i + \alpha \left(\text{Target} - V(s;\theta)\right) \frac{\partial V(s;\theta)}{\partial \theta_i}$$

For s_1, the target value is 3, and the current approximation is $V(s_1) = 2$. Thus, the error is:

$$\delta = 3 - 2 = 1$$

For θ_1, the derivative of $V(s_1)$ with respect to θ_1 is $\frac{\partial V(s_1)}{\partial \theta_1} = \phi_1(s_1) = 1$. So, the update for θ_1 is:

$$\theta_1 \leftarrow 2 + 0.1 \cdot 1 \cdot 1 = 2.1$$

For θ_2, the derivative of $V(s_1)$ with respect to θ_2 is $\frac{\partial V(s_1)}{\partial \theta_2} = \phi_2(s_1) = 0$. Therefore, θ_2 remains unchanged:

$$\theta_2 \leftarrow 3$$

Thus, the updated parameters are $\theta_1 = 2.1$ and $\theta_2 = 3$.

Q5. **Conceptual: Batch vs. Incremental Methods for Function Approximation**
Compare batch methods and incremental methods in the context of value function approximation in reinforcement learning. When would you prefer one over the other?

Solution:
Batch methods update the value function approximation by processing a large set of experiences all at once (e.g., using least squares), whereas incremental methods update the value function after each new experience (e.g., using stochastic gradient descent). Batch methods can produce more stable and accurate solutions, but they require storing and processing large amounts of data. Incremental methods are more efficient for online learning and can be used in real-time settings, but may converge more slowly and be more prone to noise.

Q6. **Computational: TD Learning with Function Approximation**
Given the following feature vector for states $\phi(s_1) = [1, 0]$, $\phi(s_2) = [0, 1]$, and $\phi(s_3) = [1, 1]$, and the current parameters $\theta = [2, 3]$, the agent experiences the transition $s_1 \xrightarrow{r=1} s_2$. Update the parameter vector θ using TD(0) learning with learning rate $\alpha = 0.1$ and discount factor $\gamma = 0.9$.

Solution:
The TD(0) update rule is:

$$\theta_i \leftarrow \theta_i + \alpha \left[r + \gamma V(s'; \theta) - V(s; \theta) \right] \frac{\partial V(s; \theta)}{\partial \theta_i}$$

Step 1: Compute the current values $V(s_1)$ and $V(s_2)$:

$$V(s_1) = 2 \cdot 1 + 3 \cdot 0 = 2$$

$$V(s_2) = 2 \cdot 0 + 3 \cdot 1 = 3$$

Step 2: Calculate the TD error:

$$\delta = r + \gamma V(s_2) - V(s_1) = 1 + 0.9 \cdot 3 - 2 = 1.7$$

Step 3: Update θ_1 and θ_2: For θ_1:

$$\theta_1 \leftarrow 2 + 0.1 \cdot 1.7 \cdot 1 = 2.17$$

For θ_2:

$$\theta_2 \leftarrow 3 + 0.1 \cdot 1.7 \cdot 0 = 3$$

Thus, the updated parameters are $\theta_1 = 2.17$ and $\theta_2 = 3$.

Q7. **Conceptual: Experience Replay in Deep Q-Networks (DQN)**
What is experience replay in Deep Q-Networks (DQN), and why is it important for improving the learning process?

Solution:
Experience replay is a technique where an agent stores its experiences (state, action, reward, next state) in a replay buffer and samples from this buffer to update the Q-network. It helps break the correlation between consecutive experiences, improving the stability of learning. By reusing past experiences, experience replay also increases the data efficiency and reduces the risk of overfitting to recent experiences.

Q8. **Computational: Least Squares Prediction for Value Function Approximation**
Given a set of feature vectors $\phi(s_1) = [1, 0]$, $\phi(s_2) = [0, 1]$, $\phi(s_3) = [1, 1]$ and the corresponding target values $V_{\text{target}}(s_1) = 2$, $V_{\text{target}}(s_2) = 3$, $V_{\text{target}}(s_3) = 4$, perform one step of the least squares update to find the best-fit parameters $\theta = [\theta_1, \theta_2]$.

Solution:
The least squares solution minimizes the error between the target values and the predicted values. We want to minimize:

$$\min_{\theta} \sum_{s} \left(V_{\text{target}}(s) - \theta^T \phi(s)\right)^2$$

We can solve this using matrix algebra, but for this step, we can set up the normal equations:

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \\ 4 \end{bmatrix}$$

By solving this system of linear equations, we find that $\theta_1 = 2$ and $\theta_2 = 2$.

Q9. **Conceptual: Bias-Variance Tradeoff in Function Approximation**
Explain the bias-variance tradeoff in the context of function approximation in reinforcement learning. How does the choice of model complexity affect this tradeoff?

Solution:
The bias-variance tradeoff describes the balance between the error due to bias (error from approximating a complex function with a simpler model) and the error due to variance (sensitivity to fluctuations in the training data). A more complex model may have low bias but high variance, while a simpler model may have high bias but low variance. The key is to find a model with an appropriate level of complexity to minimize the total error.

Q10. **Computational: Feature-Based Action-Value Function Approximation**
Consider a Q-function approximated as $Q(s, a; \theta) = \theta_1\phi_1(s, a)+\theta_2\phi_2(s, a)$. Given the following feature vectors for state-action pairs: $\phi(s_1, a_1) = [1, 0]$, $\phi(s_2, a_1) = [0, 1]$, and $\phi(s_1, a_2) = [1, 1]$, and parameters $\theta = [2, 3]$, compute the approximate action-value $Q(s_1, a_1)$, $Q(s_2, a_1)$, and $Q(s_1, a_2)$.

Solution:
The action-value function is approximated as:

$$Q(s, a; \theta) = \theta_1 \phi_1(s, a) + \theta_2 \phi_2(s, a)$$

For $Q(s_1, a_1)$:

$$Q(s_1, a_1) = 2 \cdot 1 + 3 \cdot 0 = 2$$

For $Q(s_2, a_1)$:

$$Q(s_2, a_1) = 2 \cdot 0 + 3 \cdot 1 = 3$$

For $Q(s_1, a_2)$:

$$Q(s_1, a_2) = 2 \cdot 1 + 3 \cdot 1 = 5$$