Reinforcement Learning: A Comprehensive Question Bank

Based on the Exposition by Taha Majlesi

July 12, 2025

Contents

Part I

Multiple Choice Questions

1. What is the fundamental paradigm of Reinforcement Learning?

   A. Learning from labeled input-output pairs.

   B. Finding structure in unlabeled data.

   C. Goal-directed learning from experience through trial and error.

   D. Cleaning and optimizing static datasets.

2. In the agent-environment interface, what does the agent receive from the environment at each time step?

   A. A new policy and a new state.

   B. A new state and a scalar reward.

   C. Only a new state.

   D. A batch of labeled data.

3. What is the ultimate objective of an RL agent?

   A. To maximize the immediate reward.

   B. To learn a policy that maximizes the total accumulated reward over the long run.

   C. To perfectly model the environment's dynamics.

   D. To visit as many states as possible.

4. The process of using RL to align LLMs with human preferences is called:

   A. Supervised Fine-Tuning (SFT).

   B. Reward Model Training (RMT).

   C. Reinforcement Learning from Human Feedback (RLHF).

   D. Proximal Policy Optimization (PPO).

5. In the RLHF pipeline for training ChatGPT, what is the purpose of the Reward Model (RM)?

   A. To generate initial responses to prompts.

   B. To serve as a scalable, automated proxy for human judgment.

   C. To directly update the policy using gradient descent.

   D. To fine-tune the model on a high-quality dataset of demonstrations.

6. Which algorithm is most commonly used for the final optimization step in the RLHF pipeline?

   A. Deep Q-Network (DQN).

B. Trust Region Policy Optimization (TRPO).

C. Supervised Fine-Tuning (SFT).

D. Proximal Policy Optimization (PPO).

7. According to the text, what is the key difference in learning style between SFT and RL?

   A. SFT generalizes better, while RL memorizes.

   B. SFT memorizes, while RL generalizes.

   C. Both are equally effective at generalization.

   D. SFT is for continuous tasks, while RL is for discrete tasks.

8. The formal mathematical framework for reinforcement learning is known as:

   A. The Bellman Equation.

   B. The Markov Decision Process (MDP).

   C. The Agent-Environment Interface.

   D. The Value Function.

9. Which of the following is NOT one of the five components of an MDP tuple?

   A. S (State Space).

   B. A (Action Space).

   C. V (Value Function).

   D. γ (Discount Factor).

10. What does the transition function $P(s'|s, a)$ specify?

    A. The reward for taking action a in state s.

    B. The probability of transitioning to state s' from state s after taking action a.

    C. The best action to take in state s.

    D. The value of being in state s.

11. A discount factor (γ) close to 1 creates what kind of agent?

    A. A "myopic" agent that prioritizes immediate rewards.

    B. A "farsighted" agent that values long-term gains.

    C. An agent that ignores rewards completely.

    D. An agent that acts randomly.

12. The effective planning horizon of an agent can be approximated as:

    A. $1/\gamma$.

    B. $1 - \gamma$.

    C. $1/(1 - \gamma)$.

    D. γ^2.

13. The Markov Property states that:

    A. The future is dependent on the entire past history.

    B. The future is independent of the past, given the present.

    C. All states are equally likely.

    D. The reward is always the same for a given action.

14. What is a policy (π) in reinforcement learning?

    A. The total accumulated reward.

    B. The agent's behavioral strategy, mapping states to actions.

    C. The probability of the environment changing.

    D. A measure of how good a state is.

15. A policy that outputs a probability distribution over actions is called:

    A. A deterministic policy.

    B. A stochastic policy.

    C. An optimal policy.

    D. A value-based policy.

16. What does the state-value function $V^{\pi}(s)$ quantify?

    A. The immediate reward for being in state s.

    B. The probability of reaching state s.

    C. The expected return an agent can achieve by starting in state s and following policy π.

    D. The best action to take in state s.

17. The action-value function $Q^{\pi}(s, a)$ is also known as the:

    A. Q-function.

    B. Advantage function.

    C. Reward function.

    D. Policy function.

18. The Bellman equation provides a:

    A. Way to calculate the immediate reward.

    B. Method for choosing the learning rate.

    C. Recursive relationship between the value of a state and its successors.

    D. Proof of the Markov property.

19. In the Grid World case study, what is the effect of a higher discount factor (γ)?

    A. It makes the agent prioritize closer rewards.

B. It makes the agent value distant rewards more, encouraging long-term planning.

C. It increases the environmental noise.

D. It makes the agent ignore all rewards.

20. An optimal policy π^* is a policy that:

A. Achieves a state-value function greater than or equal to that of any other policy for all states.

B. Always chooses the action with the highest immediate reward.

C. Explores the environment randomly.

D. Is always deterministic.

21. The Bellman optimality equation is different from the Bellman expectation equation because it includes a:

A. Summation over all actions.

B. max operator over actions.

C. Discount factor.

D. Transition probability.

22. Which algorithm is based on the Bellman optimality equation for $Q^*(s, a)$?

A. Value Iteration.

B. Policy Iteration.

C. Q-learning.

D. Sarsa.

23. In the Grid World exercise, which parameter setting would lead to a farsighted but risky strategy?

A. $\gamma = 0.1$, noise=0.

B. $\gamma = 0.99$, noise=0.

C. $\gamma = 0.1$, noise=0.5.

D. $\gamma = 0.99$, noise=0.5.

24. What is the key difference between an MDP and a POMDP?

A. POMDPs do not have rewards.

B. In POMDPs, the agent's state is hidden and only partially observable.

C. MDPs are for continuous action spaces, POMDPs are for discrete.

D. POMDPs do not follow the Markov property.

25. In a POMDP, what does the agent maintain to handle uncertainty about the true state?

A. A Q-table.

B. A belief state (a probability distribution over all possible states).

C. A target network.

D. A replay buffer.

26. A policy in a POMDP maps:

A. States to actions.

B. Observations to rewards.

C. Belief states to actions.

D. States to observations.

27. Which of the following is a real-world application of POMDPs?

A. Playing chess.

B. Medical diagnosis based on symptoms.

C. Sorting a list of numbers.

D. Calculating mortgage payments.

28. What was the key achievement of the Deep Q-Network (DQN) in 2013?

A. It defeated a human champion in Go.

B. It was the first algorithm to successfully train a deep neural network to learn control policies from raw pixel data.

C. It introduced the concept of policy gradients.

D. It solved the Rubik's Cube.

29. How does DQN overcome the problem of astronomically large state spaces in games like Atari?

A. By using a lookup table.

B. By using a deep convolutional neural network (CNN) as a function approximator for the Q-function.

C. By simplifying the game.

D. By using a POMDP framework.

30. What is the purpose of "Experience Replay" in DQN?

A. To increase the learning rate.

B. To break temporal correlations in training data and increase data efficiency.

C. To make the policy deterministic.

D. To directly model the environment.

31. The "target network" in DQN is used to:

A. Provide a stable target for the Q-learning updates, preventing divergence.

B. Store past experiences.

C. Approximate the policy.

D. Calculate the immediate reward.

32. Policy gradient methods are particularly well-suited for:

    A. Problems with small, discrete state spaces.
    B. Problems with continuous action spaces, like robotics.
    C. Problems where the environment is fully known.
    D. Problems that can be solved with a lookup table.

33. What is the main problem with standard policy gradient methods that TRPO addresses?

    A. They are too slow.
    B. They only work for discrete actions.
    C. They are unstable and highly sensitive to step size.
    D. They require a perfect model of the environment.

34. TRPO constrains the policy update using the:

    A. Mean Squared Error.
    B. Euclidean distance in the parameter space.
    C. Kullback-Leibler (KL) divergence between policy distributions.
    D. Huber loss.

35. What is Generalized Advantage Estimation (GAE) used for?

    A. To provide a better trade-off between bias and variance when estimating the advantage function.
    B. To speed up the MCTS algorithm.
    C. To replace the need for a discount factor.
    D. To make DQN more stable.

36. AlphaGo's success was due to a synthesis of deep neural networks and:

    A. Q-learning.
    B. Trust Region Policy Optimization (TRPO).
    C. Monte Carlo Tree Search (MCTS).
    D. Experience Replay.

37. In AlphaGo, what is the role of the "Value Network"?

    A. To predict the most promising moves to consider.
    B. To estimate the probability of winning from a given board position.
    C. To store a database of human expert games.
    D. To execute the final move.

38. What is the role of the "Policy Network" in AlphaGo?

    A. To evaluate the "goodness" of a board position.
    B. To predict the next move by outputting a probability distribution over legal moves.
    C. To perform rollouts to the end of the game.

D. To update the MCTS tree.

39. How was AlphaGo's initial policy network trained?

    A. Entirely through self-play.
    B. Via supervised learning on a massive database of human expert games.
    C. Using the Q-learning algorithm.
    D. By playing against a random opponent.

40. What was the major innovation of AlphaGo Zero over the original AlphaGo?

    A. It used a deeper neural network.
    B. It learned entirely from self-play with no human data.
    C. It used PPO instead of MCTS.
    D. It could play multiple games, not just Go.

41. Proximal Policy Optimization (PPO) was developed to be simpler and more practical than:

    A. DQN.
    B. Q-learning.
    C. TRPO.
    D. MCTS.

42. What is the central innovation of PPO?

    A. The use of a target network.
    B. The use of a clipped surrogate objective function.
    C. The use of Monte Carlo Tree Search.
    D. The use of a replay buffer.

43. The 'clip' operator in the PPO objective function serves to:

    A. Increase the learning rate.
    B. Encourage the new policy to be as different as possible from the old one.
    C. Discourage the new policy from moving too far from the old one, ensuring stability.
    D. Calculate the advantage function.

44. PPO is considered a _____ algorithm.

    A. Second-order.
    B. First-order.
    C. Model-based.
    D. Value-based.

45. Besides its use in RLHF, PPO was famously used to train the OpenAI Five agent for which game?

    A. Go.
    B. Atari's Pong.

C. Chess.

D. Dota 2.

46. In the RLHF pipeline, what is often included in the PPO objective function to prevent the LLM from straying too far from its original knowledge?

    A. A reward bonus.

    B. A penalty term based on the KL-divergence between the current and initial policies.

    C. A larger batch size.

    D. A higher learning rate.

47. The historical progression of Deep RL shows a virtuous cycle between theory, algorithms, and:

    A. Human feedback.

    B. Supervised learning.

    C. Computational power.

    D. Data availability.

48. What remains a primary open challenge for reinforcement learning?

    A. Inability to solve games.

    B. Sample inefficiency (requiring millions of interactions).

    C. Incompatibility with deep learning.

    D. Lack of a formal mathematical framework.

49. A policy $\pi : S \to A$ is a:

    A. Stochastic policy.

    B. Deterministic policy.

    C. Value function.

    D. Reward function.

50. In the MDP tuple (S, A, P, R, γ), what does R represent?

    A. The return.

    B. The replay buffer.

    C. The reward function.

    D. The policy.

51. The value $Q^*(s, a)$ represents the expected return from taking action a in state s and then:

    A. Acting randomly thereafter.

    B. Following the current policy π.

    C. Following an optimal policy π^* thereafter.

    D. Terminating the episode.

52. The first step in the RLHF pipeline is:

A. Reward Model Training.
B. PPO Optimization.
C. Supervised Fine-Tuning (SFT).
D. Data Collection.

53. What does a belief state $b(s)$ in a POMDP represent?

A. The agent's certainty about the current state.
B. The probability distribution over all possible states.
C. The agent's desired state.
D. The observation received by the agent.

54. DQN's architecture is efficient because it computes Q-values for all actions in a single:

A. Backward pass.
B. Training epoch.
C. Forward pass through the network.
D. Experience replay sample.

55. The "moving target" problem in Q-learning is caused by:

A. Using a fixed policy.
B. The environment being stochastic.
C. The target Q-value changing with every weight update of the same network.
D. The learning rate being too small.

56. TRPO was a landmark algorithm for stabilizing:

A. Value-based methods.
B. Policy gradient methods.
C. Model-based methods.
D. Monte Carlo Tree Search.

57. The "trust region" in TRPO is a constraint on the:

A. Size of the neural network.
B. Change in the policy's output distribution.
C. Amount of memory used.
D. Number of steps in an episode.

58. The MCTS algorithm in AlphaGo uses neural networks to:

A. Randomly explore the game tree.
B. Guide its search to focus on the most promising lines of play.
C. Replace the need for a search tree entirely.
D. Play against human opponents.

59. The complexity of Go, with a state space larger than the number of atoms in the universe, makes what infeasible?

A. Supervised learning.
B. Brute-force search.
C. Using a discount factor.
D. Reinforcement learning.

60. PPO's simplicity comes from replacing TRPO's complex second-order optimization with a:

A. Simple gradient ascent.
B. First-order mechanism (the clipped objective).
C. Lookup table.
D. Heuristic search.

61. The hyperparameter ϵ in the PPO objective function defines the:

A. Learning rate.
B. Discount factor.
C. Clipping range.
D. Number of network layers.

62. The "credit assignment problem" in RL refers to:

A. Determining which actions in a sequence are responsible for a future reward.
B. Assigning credit to the developers of an algorithm.
C. Deciding how much memory to allocate to the replay buffer.
D. Choosing the correct learning rate.

63. In the context of RL, "exploration" refers to:

A. Always choosing the action with the highest known value.
B. Trying out different actions to discover their outcomes.
C. Reducing the dimensionality of the state space.
D. Using a pre-trained model.

64. In the context of RL, "exploitation" refers to:

A. Always choosing the action with the highest known value to maximize immediate return.
B. Trying out new, unknown actions.
C. Updating the policy based on human feedback.
D. Storing experiences in a replay buffer.

65. A stochastic policy is useful for encouraging:

A. Fast convergence.
B. Exploitation.
C. Exploration.
D. Memorization.

66. The Bellman equations are named after:

A. Richard Bellman.
B. Andrew Barto.
C. Richard Sutton.
D. Geoffrey Hinton.

67. If an environment is deterministic, the transition probability $P(s'|s, a)$ for a specific outcome s' is:

A. 0.
B. 0.5.
C. 1.
D. Dependent on the policy.

68. The main challenge in a POMDP that is not present in an MDP is:

A. Credit assignment over time.
B. The curse of dimensionality.
C. State estimation (belief tracking).
D. Choosing a discount factor.

69. The value function in a finite-horizon POMDP is known to be:

A. Linear and convex.
B. Piecewise-linear and convex (PWLC).
C. Quadratic and concave.
D. Non-convex.

70. The 2013 Atari paper was published by researchers at:

A. OpenAI.
B. Berkeley.
C. Google.
D. DeepMind.

71. TRPO was primarily developed at:

A. DeepMind.
B. OpenAI.
C. Berkeley.
D. Stanford.

72. The "rollout" step in AlphaGo's MCTS involves:

A. Using the value network to get a score.
B. Using a fast policy to simulate the game to the end.
C. Asking a human expert for the best move.
D. Expanding the tree with all possible moves.

73. The final move selection in AlphaGo is based on:

A. The path with the highest action-value.

B. The most visited path from the root of the MCTS tree.
C. The recommendation of the policy network.
D. A random choice among the top moves.

74. What is a significant bottleneck for applying RL to real-world robotics?

A. The difficulty of defining a reward function.
B. The high sample complexity of many algorithms.
C. The lack of suitable hardware.
D. The inability to handle continuous action spaces.

75. The term "function approximator" in Deep RL refers to:

A. A lookup table.
B. A heuristic function.
C. A deep neural network used to represent a policy or value function.
D. The MCTS algorithm.

76. In the Grid World exercise, a myopic and safe strategy corresponds to:

A. High γ, high noise.
B. Low γ, zero noise.
C. High γ, zero noise.
D. Low γ, high noise.

Answer Key for Multiple Choice Questions

1. C
2. B
3. B
4. C
5. B
6. D
7. B
8. B
9. C
10. B
11. B
12. C
13. B
14. B
15. B
16. C
17. A
18. C
19. B
20. A
21. B
22. C
23. D
24. B
25. B
26. C
27. B
28. B
29. B
30. B
31. A
32. B
33. C
34. C
35. A
36. C
37. B
38. B
39. B
40. B
41. C
42. B
43. C
44. B
45. D
46. B
47. C
48. B
49. B
50. C
51. C
52. B
53. C
54. B
55. C
56. B
57. C
58. B
59. B
60. B
61. A
62. C
63. A
64. C
65. A
66. C
67. B
68. D
69. C
70. B
71. B
72. C
73. B
74. B

Part II

Explanatory Questions and Answers

1. **Explain the agent-environment interface in Reinforcement Learning and the role of each component in the interaction loop.**

   Answer

   The agent-environment interface is the core framework of RL. It consists of two main components:

   - **The Agent:** The learner and decision-maker.
   - **The Environment:** The world in which the agent operates.

   The interaction loop proceeds in discrete time steps. At each step t:

   (a) The agent observes the current **state** (s_t) of the environment.

   (b) Based on this state, the agent selects an **action** (a_t).

   (c) The environment receives the action, transitions to a new **state** (s_{t+1}), and provides the agent with a scalar **reward** (r_{t+1}).

   This loop continues, with the agent's goal being to learn a policy (a strategy for choosing actions) that maximizes the cumulative reward over time, not just the immediate reward.

2. **Describe the three-step pipeline of Reinforcement Learning from Human Feedback (RLHF) as used in training models like ChatGPT.**

Answer

RLHF is a process to align large language models with complex human preferences. The pipeline consists of three steps:

(a) **Supervised Fine-Tuning (SFT):** A pre-trained language model is first fine-tuned on a high-quality, curated dataset of prompts and desired responses created by human labelers. This step teaches the model the expected style and format for responses, giving it a strong starting point.

(b) **Reward Model (RM) Training:** To go beyond the limitations of SFT, a separate reward model is trained. For a given prompt, the SFT model generates several responses. Human labelers then rank these responses from best to worst. This comparison data is used to train the RM, which learns to output a scalar score that reflects the degree of human preference for any given response. It acts as an automated proxy for human judgment.

(c) **Reinforcement Learning Optimization:** The SFT model (now acting as the policy) is further optimized using an RL algorithm, typically PPO. The policy generates a response to a prompt, the RM evaluates it and provides a reward, and this reward signal is used to update the policy's parameters. This iteratively refines the model's behavior to maximize the rewards from the learned preference model, effectively aligning it with human values.

3. **What is the Markov Property and why is it a crucial assumption for the MDP framework?**

Answer

The Markov Property states that **the future is independent of the past, given the present**. In the context of an MDP, this means that the next state s_{t+1} and the expected reward r_{t+1} depend only on the current state s_t and the current action a_t, not on the entire history of states and actions that came before.

This assumption is crucial because it makes the reinforcement learning problem tractable. If the agent needed to consider the entire history to make a decision, the policy would have to map histories to actions. The space of possible histories grows exponentially with time, making it computationally infeasible to learn or store such a policy. The Markov Property allows the agent's policy to be a function of only the current state, $\pi(a|s)$, which is a much simpler and more manageable problem.

4. **Explain the role of the discount factor (γ) and how its value shapes an agent's behavior.**

Answer

The discount factor, γ, is a value between 0 and 1 that determines the present value of future rewards. A reward received k time steps in the future is discounted by a factor of γ^k. Its role is to balance the importance of immediate versus future rewards.
The value of γ fundamentally shapes the agent's behavior and strategy:

- **When γ is close to 0:** The agent is **"myopic"** or short-sighted. Future rewards are heavily discounted, so the agent will prioritize actions that yield high immediate rewards, even if they lead to poor long-term outcomes.
- **When γ is close to 1:** The agent is **"farsighted"**. Future rewards retain most of their value, so the agent is willing to endure short-term costs or small rewards to achieve a much larger reward in the distant future. It encourages long-term planning.

The effective planning horizon of an agent can be approximated as $1/(1-\gamma)$, so a γ of 0.99 implies the agent is planning over a horizon of about 100 steps.

5. **What is the difference between a state-value function ($V^\pi(s)$) and an action-value function ($Q^\pi(s,a)$)?**

Answer

Both are value functions that measure the "goodness" of a situation under a given policy π, but they do so at different levels of granularity.

- **State-Value Function ($V^\pi(s)$)**: This function represents the expected total future reward an agent will receive starting from a particular **state** s and then following policy π thereafter. It answers the question: "How good is it to be in this state?"
- **Action-Value Function ($Q^\pi(s,a)$)**: This function represents the expected total future reward an agent will receive after taking a specific **action** a in state s and then following policy π thereafter. It answers the question: "How good is it to take this action in this state?"

The Q-function is often more useful for control because it allows an agent to directly compare the quality of different actions within a state without needing a model of the environment. An agent can simply choose the action with the highest Q-value for the current state.

6. **Explain the Bellman Optimality Equation for $V^*(s)$ and what each part of the equation represents.**

Answer

The Bellman Optimality Equation for the optimal state-value function $V^*(s)$ is:

$$V^*(s) = \max_{a \in \mathcal{A}} \sum_{s',r} p(s', r|s, a)[r + \gamma V^*(s')]$$

This equation states that the value of a state under an optimal policy must equal the expected return for the best possible action from that state. Let's break it down:

- $V^*(s)$: The value of being in state s, assuming the agent acts optimally from this point forward.
- $\max_{a \in \mathcal{A}}$: This is the key difference from the expectation equation. It signifies that the agent must choose the action a that **maximizes** the expected future return. This is the principle of optimality.
- $\sum_{s',r} p(s', r|s, a)[...]$: This is the expectation over all possible next states s' and rewards r, given the current state s and the chosen action a.
- $[r + \gamma V^*(s')]$: This is the core of the recursive definition. It's the immediate reward r plus the discounted value of the next state s', where $V^*(s')$ is itself the optimal value of that successor state.

7. **Compare and contrast Markov Decision Processes (MDPs) and Partially Observable Markov Decision Processes (POMDPs).**

Answer

Markov Decision Process (MDP)	**Partially Observable MDP (POMDP)**
The environment's state is **fully observable**. The agent knows its true state s_t at all times.	The environment's state is **hidden** or partially observable.
The agent's input is the true state s_t.	The agent's input is an **observation** o_t, which is probabilistically related to the true state.
The policy $\pi(s)$ maps states to actions.	The policy $\pi(b)$ maps **belief states** (probability distributions over states) to actions.
The key challenge is temporal credit assignment (figuring out which past actions led to future rewards).	The key challenges are **state estimation** (maintaining the belief state) in addition to planning and credit assignment.
The state space is the environment's (often discrete) state space $\mathcal{S}$.	The "state space" for the policy is the continuous space of all possible belief distributions, which is much more complex.

8. **What were the two core innovations of the Deep Q-Network (DQN), and**

why were they necessary for stable training?

Answer

The two core innovations of DQN were **Experience Replay** and the use of a **Target Network**. They were necessary to address the instability that arises when combining Q-learning with non-linear function approximators like deep neural networks.

(a) **Experience Replay:**

- **What it is:** Instead of training on experiences as they occur, the agent stores them in a large buffer (replay memory). During training, mini-batches of experiences are sampled randomly from this buffer.
- **Why it's necessary:** RL data is sequential and highly correlated. Training on consecutive samples violates the i.i.d. (independent and identically distributed) assumption of many optimization algorithms, leading to high variance and instability. Replay breaks these temporal correlations. It also increases data efficiency, as each experience can be used for multiple updates.

(b) **Target Network:**

- **What it is:** A separate neural network, a clone of the main Q-network, is used to calculate the target value for the Q-learning update. Its weights are held fixed for a period and only periodically updated to match the main network's weights.
- **Why it's necessary:** The Q-learning target is $y_t = r_t + \gamma \max_{a'} Q(s_{t+1}, a'; \theta)$. If the same network θ is used for both the prediction and the target, the target value shifts with every single weight update. This "moving target" problem can cause oscillations and training divergence. The fixed target network provides a stable, consistent target for the updates, significantly improving training stability.

9. **Why are standard policy gradient methods often unstable, and how does Trust Region Policy Optimization (TRPO) address this problem?**

Answer

Standard policy gradient methods are often unstable because they are highly sensitive to the choice of step size (learning rate). They update the policy parameters by taking a step in the direction of the gradient.

- If the step size is too small, learning is very slow.
- If the step size is too large, a single update can lead to a catastrophic drop in performance. A small change in the parameter space can cause a very large, detrimental change in the policy's behavior, from which it may never recover.

TRPO addresses this by reformulating the optimization problem. Instead of taking an unconstrained step, TRPO aims to maximize the policy's performance subject to a **constraint** that the new policy does not deviate too far from the old one. Crucially, this constraint is not on the distance in the parameter space, but on the distance between the policies' output distributions, measured by the **Kullback-Leibler (KL) divergence**. By enforcing this "trust region," TRPO ensures that policy updates are kept small and controlled, preventing catastrophic performance drops and leading to more stable, monotonic improvements.

10. **Describe the roles of the Policy Network and the Value Network in the AlphaGo architecture.**

Answer

AlphaGo's architecture masterfully combines deep learning with Monte Carlo Tree Search (MCTS). The two neural networks act as "intuition" to guide the search, making it feasible to navigate Go's enormous game tree.

(a) **The Policy Network:**

- **Role:** To predict the next move. It answers the question: "What are the most promising moves to consider from this position?"
- **Function:** It takes the current board state as input and outputs a probability distribution over all legal moves. This is used during the "Expansion" phase of MCTS to prune the search, focusing only on a small number of high-probability moves instead of all possible moves.

(b) **The Value Network:**

- **Role:** To evaluate a board position. It answers the question: "Given this board configuration, who is likely to win?"
- **Function:** It takes the current board state as input and outputs a single scalar value estimating the probability of winning from that state. This allows AlphaGo to truncate its search. Instead of simulating a game to the very end (which is slow), it can evaluate a position with the value network and use that score to update the MCTS tree.

Together, the policy network provides breadth (which moves to look at) and the value network provides depth (how good a position is without searching further).

11. **What is the main advantage of Proximal Policy Optimization (PPO) over TRPO, and how does its clipped surrogate objective function work?**

Answer

The main advantage of PPO over TRPO is its **simplicity and ease of implementation**. While TRPO provides strong theoretical guarantees of stable improvement, its practical implementation is complex, requiring a computationally intensive second-order optimization step (the conjugate gradient algorithm). PPO achieves similar stability benefits with a much simpler, first-order optimization method.
PPO's central innovation is the **clipped surrogate objective function**:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t) \right]$$

Here's how it works:

- $r_t(\theta)$ is the probability ratio between the new and old policies. If $r_t > 1$, the action is more likely under the new policy.
- $\hat{A}_t$ is the estimated advantage. If it's positive, the action was good; if negative, it was bad.
- The 'clip' function constrains the ratio $r_t(\theta)$ to be within the interval $[1-\epsilon, 1+\epsilon]$.
- The 'min' operator takes the minimum of the normal objective and the clipped objective.

The effect is that if an update would change the policy too much (i.e., $r_t(\theta)$ is outside the clipping interval), the objective function is clipped, removing the incentive for the policy to change so drastically. This simple mechanism effectively enforces a trust region without the computational complexity of TRPO.

12. **Explain the concept of "SFT Memorizes, RL Generalizes" in the context of training foundation models.**

Answer

This phrase highlights a key distinction in how Supervised Fine-Tuning (SFT) and Reinforcement Learning (RL) affect a model's capabilities.

- **SFT Memorizes:** SFT trains a model on a fixed dataset of high-quality input-output pairs. The model learns to mimic these examples very well. As a result, it achieves high performance on tasks that are similar to its training data (in-distribution). However, it often struggles with novel scenarios (out-of-distribution) because it has learned to rely on surface-level statistical patterns and has effectively "memorized" the correct responses for specific types of inputs.
- **RL Generalizes:** RL, particularly through exploration and feedback, encourages the model to learn a more robust, underlying decision-making process rather than just input-output pairings. By receiving rewards for the consequences of its actions across a wide range of situations, the RL-trained agent develops a policy that is less brittle and more adaptable. It learns a general strategy for achieving a goal, which can be applied more effectively in unfamiliar situations.

This is why RLHF is a critical step after SFT; it pushes the model beyond simple mimicry towards a more generalized understanding of how to produce desirable outputs.

13. **What is a belief state in a POMDP and how is it updated?**

Answer

A **belief state**, denoted $b(s)$, is a probability distribution over the set of all possible environment states $\mathcal{S}$. It is the agent's internal representation of its uncertainty about the true, hidden state of the world. Since the agent cannot observe the state s directly in a POMDP, the belief state $b(s)$ serves as a sufficient statistic for the history, meaning it summarizes all the information from past actions and observations needed to make an optimal decision.
The belief state is updated at each time step using a two-step process that incorporates the agent's action and the new observation, based on Bayes' rule:

(a) **Prediction Step:** After taking action a from belief state $b(s)$, the agent predicts the next belief state before seeing the new observation. This is done by considering all possible next states s' that could be reached from each original state s, weighted by the transition probabilities $P(s'|s, a)$ and the prior belief $b(s)$.

(b) **Update Step:** After receiving the new observation o, the agent updates its belief. It uses the observation probability function $O(o|s', a)$ to adjust the predicted belief, increasing the probability of states s' that are likely to have produced the observation o, and decreasing the probability of others. The result is the new belief state $b'(s')$.

14. **How did AlphaGo's MCTS use both the policy and value networks during**

a single simulation (selection, expansion, evaluation, backup)?

Answer

During a game, AlphaGo runs thousands of simulations using MCTS to choose its next move. Each simulation involves four steps where the networks play a crucial role:

(a) **Selection:** The algorithm starts at the root of the search tree (the current board state) and traverses down the tree by selecting moves that have high action-values (exploitation) plus an exploration bonus. The networks are not directly used here, but the values they helped compute in previous simulations guide this step.

(b) **Expansion:** When the traversal reaches a leaf node (a position not yet explored), the tree is expanded. The **policy network** is called on this leaf node's state. It provides a probability distribution over promising next moves. The tree is then expanded by adding these new moves as children of the leaf node. This prunes the search, focusing it on moves the policy network deems worthy.

(c) **Evaluation:** The newly expanded leaf node is evaluated in two ways:

- The **value network** is called to provide a quick, deep evaluation of the position, estimating the win probability from that state.
- A "rollout" is performed using a separate, fast policy to play the game to the end, providing a second, less accurate but complete evaluation. These two evaluations are combined to produce a final score for the node.

(d) **Backup:** The outcome of the evaluation (the win/loss score) is propagated back up the tree along the path taken during the selection phase. The action-values of all parent nodes along this path are updated with this new information.

15. **What is the "credit assignment problem" in reinforcement learning?**

Answer

The credit assignment problem is a fundamental challenge in RL that arises from the delay between actions and their resulting rewards. In many tasks, an agent performs a long sequence of actions, and a reward (either positive or negative) is only received much later.
The problem is then: **how to determine which specific actions in that sequence were responsible for the final outcome?**
For example, in a game of chess, a single brilliant move early in the game might be the primary cause of a victory 50 moves later. Conversely, a subtle mistake might lead to a loss much later on. The credit assignment problem is the challenge of correctly attributing the final reward back to the crucial actions in the sequence that caused it, and not incorrectly rewarding or penalizing irrelevant or neutral actions. Solving this is essential for learning an effective policy. Value functions and advantage estimators are key mechanisms designed to help solve this problem.

16. **Explain the exploration-exploitation trade-off.**

Answer

The exploration-exploitation trade-off is a central dilemma in reinforcement learning. An agent must choose between two competing goals at each step:

- **Exploitation:** The agent can choose to take the action that it currently believes will yield the highest cumulative reward, based on its past experience. This means exploiting its current knowledge to maximize its performance.
- **Exploration:** The agent can choose to take a different, perhaps seemingly suboptimal, action to gather more information about the environment. This means exploring new possibilities with the hope of discovering an even better action or strategy for the future.

The trade-off is that excessive exploitation might cause the agent to get stuck in a locally optimal strategy, never discovering a globally optimal one. On the other hand, excessive exploration means the agent spends too much time taking random or poor actions and never capitalizes on its knowledge to actually perform the task well. A successful RL agent must carefully balance these two needs: exploring enough to find good strategies but exploiting that knowledge to achieve a high reward.

17. **Why is a stochastic policy often more advantageous than a deterministic one, especially during learning?**

Answer

A stochastic policy, which outputs a probability distribution over actions ($\pi(a|s)$), has several advantages over a deterministic policy ($\pi(s) = a$), particularly during the learning process:

(a) **Built-in Exploration:** A stochastic policy naturally encourages exploration. By sampling actions from a distribution, the agent can try different actions in a given state instead of always choosing the one that currently seems best. This is crucial for discovering new, potentially better strategies and avoiding getting stuck in a suboptimal loop.

(b) **Handling Uncertainty:** In stochastic environments, where the same action in the same state can lead to different outcomes, a deterministic policy can be brittle. A stochastic policy can be more robust by hedging its bets and maintaining a distribution over potentially good actions.

(c) **Breaking Symmetries:** In situations where multiple actions are equally good, a deterministic policy might arbitrarily always choose one, which could be suboptimal if the environment has hidden dynamics. A stochastic policy can break this symmetry by trying all good actions.

(d) **Smoother Updates in Policy Gradient Methods:** Policy gradient algorithms work by adjusting the probabilities of actions. The continuous nature of these probabilities allows for smooth, gradual policy updates, which is often more stable than the discrete, all-or-nothing changes of a deterministic policy.

18. **What is function approximation in Deep RL and why is it necessary?**

Answer

Function approximation is the use of a parameterized function, such as a deep neural network, to represent the value function ($V(s)$ or $Q(s, a)$) or the policy ($\pi(a|s)$). Instead of storing the value for every single state-action pair in a table (a Q-table), the agent learns the weights (θ) of the network that approximates the true function.
It is necessary for solving any non-trivial problem due to the **curse of dimensionality**. In problems with large or continuous state/action spaces (like controlling a robot from sensor inputs or playing Atari from pixels), the number of possible states is astronomically large or infinite. It is impossible to store a lookup table for all these states or to visit every state to learn its value.
Function approximation solves this by:

- **Generalization:** It allows the agent to generalize from states it has seen to similar, unseen states. The network can estimate the value of a new state based on its features, without ever having visited it before.
- **Memory Efficiency:** It requires storing only the network weights, which is vastly more memory-efficient than storing a giant table.

The Deep Q-Network (DQN) is a prime example, using a CNN to approximate the Q-function for playing Atari games directly from pixel inputs.

19. **What was the significance of AlphaGo Zero's achievement compared to the original AlphaGo?**

Answer

The significance of AlphaGo Zero's achievement was that it demonstrated that a machine could achieve superhuman performance in a highly complex domain **without any human knowledge or data**.
The original AlphaGo was a landmark achievement, but its training process was bootstrapped from human expertise:

- Its policy network was first trained via supervised learning on a massive database of 30 million moves from human expert games. It learned to "mimic" human play.
- It was then refined through RL, but its foundation was human strategy.

AlphaGo Zero started from scratch. It was given only the basic rules of Go. It learned entirely through **self-play reinforcement learning**. It played millions of games against itself, starting from random moves and gradually discovering the principles, strategies, and tactics of Go on its own.
The key significance is that AlphaGo Zero not only rediscovered human strategies but also developed novel, creative, and superior strategies that were beyond the scope of human knowledge. This proved that an AI could surpass the limits of human understanding in a complex domain, a major step towards more general and powerful artificial intelligence.

20. **Explain the symbiotic relationship between theory, algorithms, and computation in the history of Deep RL.**

Answer

The history of Deep Reinforcement Learning is a compelling example of a virtuous cycle where theory, algorithms, and computation propel each other forward:

(a) **Theory Enables Algorithms:** The foundational mathematical theory of RL, such as the concepts of MDPs and the Bellman equations, provided the principles upon which algorithms could be built. For example, the Bellman equation directly gave rise to algorithms like Q-learning and Value Iteration.

(b) **Algorithms Leverage Computation:** For decades, these algorithms were limited to small, "toy" problems. The advent of deep learning and powerful GPUs provided the **computational** substrate needed to scale these algorithms to high-dimensional problems. The Deep Q-Network (DQN) was born from combining the Q-learning **algorithm** with the **computational** power of CNNs.

(c) **Computation Enables New Frontiers, Driving New Theory/Algorithms:** The ability to train massive neural networks allowed RL to tackle previously impossible problems (e.g., Go, robotics, LLM alignment). These new, complex applications revealed the limitations of existing algorithms (e.g., the instability of policy gradients). This, in turn, drove the development of new **theory** and more robust **algorithms** like TRPO and PPO to address these new challenges.

This symbiotic relationship continues to be the primary engine of progress in the field.

21. **What is Generalized Advantage Estimation (GAE) and what problem does it help solve?**

Answer

Generalized Advantage Estimation (GAE) is a technique used in policy gradient methods (like TRPO and PPO) to estimate the advantage function, $\hat{A}_t$. The advantage function, $A(s, a) = Q(s, a) - V(s)$, represents how much better taking action a is compared to the average action in state s. A good estimate of the advantage is crucial for effective policy updates.
The problem GAE helps solve is the **bias-variance trade-off** in this estimation. There are several ways to estimate the advantage, each with its own trade-off:

- A simple one-step estimate has low variance but can be highly biased (inaccurate).
- A Monte Carlo estimate (using the full return of an episode) is unbiased but can have very high variance, making learning unstable.

GAE provides a way to balance this trade-off. It computes a weighted average of advantage estimates over multiple time steps. It uses a parameter, λ (similar to γ), to control this balance.

- When $\lambda = 0$, GAE reduces to the simple, low-variance, high-bias one-step estimate.
- When $\lambda = 1$, GAE is equivalent to the high-variance, unbiased Monte Carlo estimate.

By choosing a λ between 0 and 1, GAE can find a sweet spot that significantly reduces the variance of the advantage estimate without introducing too much bias, leading to more stable and efficient learning.

22. **Why is sample inefficiency a major bottleneck for real-world RL applications, especially in robotics?**

Answer

Sample inefficiency refers to the fact that many state-of-the-art Deep RL algorithms (like DQN and PPO) require a massive number of interactions with the environment—often millions or even billions of time steps—to learn an effective policy.
This is a major bottleneck for real-world applications, especially in robotics, for several reasons:

(a) **Time and Cost:** Collecting millions of samples on a physical robot is incredibly time-consuming and expensive. An experiment that takes a few hours in a fast simulation could take weeks or months on a real robot.

(b) **Wear and Tear:** Physical hardware degrades over time. Running a robot for millions of cycles can lead to mechanical failure, requiring costly repairs and maintenance.

(c) **Safety:** During the initial exploration phase, an RL agent will inevitably try many random and potentially dangerous actions. In a simulation, this is harmless. On a physical robot, this could lead to the robot damaging itself, its surroundings, or even harming people.

(d) **Resetting the Environment:** Many RL training paradigms rely on being able to easily reset the environment to a starting state after each episode. This is trivial in simulation but can be difficult, slow, or impossible in the real world.

Because of these challenges, much research is focused on developing more sample-efficient algorithms, including model-based RL (where the agent learns a model of the world to "practice" in) and transfer learning from simulation to the real world (sim-to-real).

23. **In the context of the Grid World exercise, explain why a low discount factor ($\gamma = 0.1$) and zero noise leads to a "myopic but safe" strategy.**

Answer

This strategy arises from the combination of the two parameters:

(a) **Low Discount Factor ($\gamma = 0.1$) $\rightarrow$ Myopic Behavior:** A discount factor of 0.1 means the agent is extremely short-sighted. A reward received one step in the future is only worth 10% of its value, and a reward two steps away is worth only 1%. The distant, large reward of +10 becomes almost worthless compared to the immediate, small reward of +1. For example, if the +10 reward is 5 steps away, its discounted value is $0.1^4 \times 10 = 0.001$. The agent's value function will be dominated by the closest reward, so its policy will be to go for the +1 exit.

(b) **Zero Noise $\rightarrow$ Safe Behavior:** Zero noise means the environment is deterministic. When the agent chooses an action (e.g., move RIGHT), it moves RIGHT with 100% certainty. There is no risk of slipping and accidentally falling into the cliff. The agent has perfect control over its movements.

Combining these two, the agent is myopic and prioritizes the +1 reward. Since it has perfect control, it can execute a plan to get there without any risk of falling off the cliff. Therefore, the optimal policy is to take the safe, short path to the close exit, completely avoiding any risk associated with the cliff.

24. **What is the "moving target" problem in Q-learning and how does the target network in DQN solve it?**

Answer

The "moving target" problem is a source of instability when training a Q-network. The update rule for Q-learning involves minimizing the difference between the current Q-value prediction, $Q(s_t, a_t; \theta)$, and a "target" value, y_t. This target is calculated using the Bellman equation:

$$y_t = r_t + \gamma \max_{a'} Q(s_{t+1}, a'; \theta)$$

The problem arises when the **same network** (with weights θ) is used to compute both the current prediction and the target value. Every time the weights θ are updated via gradient descent, the function used to calculate the target y_t also changes.
This means the policy is chasing a "moving target." It's like trying to hit a target that moves every time you adjust your aim. This coupling can lead to feedback loops, oscillations, and divergence in the training process.
The **target network** in DQN solves this by decoupling the prediction from the target. It introduces a second network, the target network, with weights θ^-. This network is a clone of the main network, but its weights are held fixed. The target is now calculated as:

$$y_t = r_t + \gamma \max_{a'} Q(s_{t+1}, a'; \theta^-)$$

Since θ^- is fixed, the target remains stable for a period of time. The main network's weights θ can be updated multiple times towards this stable target. Periodically (e.g., every 10,000 steps), the weights of the target network are updated to match the main network ($\theta^- \leftarrow \theta$). This provides the stability needed for the learning process to converge reliably.

25. **How does the reward function in RLHF differ from the reward function in classic RL problems like games?**

Answer

The reward function in RLHF represents a fundamental evolution from the reward functions used in classic RL problems like Atari games or chess.

- **Classic RL Reward Function:** In classic problems, the reward function is typically **simple, sparse, and externally defined** by the rules of the environment. For example:
  - In Pong: +1 for scoring a point, -1 for being scored on, 0 otherwise.
  - In Chess: +1 for a win, -1 for a loss, 0 for a draw.
  - In a maze: +1 for reaching the exit, -0.1 for every step to encourage efficiency.

  These rewards are objective, easily programmable, and an inherent part of the problem definition.

- **RLHF Reward Function:** In RLHF, the goal is to align a model with complex, nuanced, and subjective human values like "helpfulness," "truthfulness," or "harmlessness." These concepts are not easily programmable into a simple reward function.

  Therefore, the reward function in RLHF is itself a **learned model**. It is a separate neural network (the Reward Model) that has been trained on human preference data (rankings of different model outputs) to **internalize and represent complex human values**. Instead of being an external signal from the environment, the reward is an internal judgment from a learned proxy of a human. This abstraction allows the powerful optimization machinery of RL to be applied to domains where the objectives are subjective and not easily defined with simple rules.

26. **What is the difference between a deterministic policy and a stochastic policy? Provide an example of each.**

Answer

The difference lies in how they map states to actions.

- **Deterministic Policy ($\pi : S \to A$)**: A deterministic policy provides a direct mapping from a state to a single, specific action. For any given state, the action is always the same.
  - **Formalism:** $a = \pi(s)$
  - **Example:** In a maze, a deterministic policy might be: "If you are at coordinate (3,4), **always** move NORTH." There is no ambiguity.
- **Stochastic Policy ($\pi : S \times A \to [0, 1]$)**: A stochastic policy maps a state to a probability distribution over the entire action space. For any given state, it specifies the probability of taking each possible action.
  - **Formalism:** $\pi(a|s) = \Pr(A_t = a|S_t = s)$
  - **Example:** In the same maze at coordinate (3,4), a stochastic policy might be: "Move NORTH with 80% probability, move EAST with 10% probability, and move WEST with 10% probability." The agent then samples an action from this distribution.

Stochastic policies are crucial for exploration during learning and can be more robust in uncertain environments.

27. **Explain why PPO is considered a "first-order" algorithm while TRPO is more complex.**

Answer

The distinction between "first-order" and "second-order" relates to the type of derivative information used in the optimization process.

- **TRPO (More Complex, Second-Order Nature):** TRPO enforces its trust region constraint on the KL-divergence. To solve this constrained optimization problem efficiently, it uses the **conjugate gradient algorithm**. This algorithm requires computing matrix-vector products with the Hessian matrix (the matrix of second derivatives) of the constraint. While it doesn't compute the full Hessian, this reliance on second-derivative information makes it a second-order method. These methods are powerful but computationally expensive and more complex to implement.
- **PPO (Simpler, First-Order):** PPO's main goal was to capture the stability of TRPO without its complexity. It achieves this by replacing the hard KL constraint with the **clipped surrogate objective**. This new objective function can be optimized using standard gradient ascent methods (like Adam), which only require computing the **gradient** (the first derivative) of the objective function. Because it only relies on first-derivative information, it is a **first-order** algorithm. This makes it much simpler to code, more computationally efficient, and more compatible with standard deep learning software libraries.

28. **What is the "curse of dimensionality" and how does function approximation help mitigate it in RL?**

Answer

The "curse of dimensionality" is a term that describes the exponential growth in volume associated with adding extra dimensions to a space. In reinforcement learning, this applies to the state and action spaces.
As the number of features describing a state (or the number of joints in a robot) increases, the size of the state space grows exponentially. For example:

- A simple grid of 10x10 has 100 states.
- A grid of 10x10x10 has 1,000 states.
- An Atari screen of 84x84 pixels with 4 frames has a state space so large ($256^{84\times84\times4}$) that it's impossible to visit every state, let alone store a value for each one in a table.

This makes traditional tabular methods (like Q-tables) completely infeasible for any interesting problem.
Function approximation, particularly with deep neural networks, helps mitigate this curse in two ways:

(a) **Memory Efficiency:** Instead of storing a value for every state, we only need to store the weights of the neural network, which is a fixed and much smaller amount of memory.

(b) **Generalization:** This is the most important contribution. A neural network can learn underlying patterns in the state features. This allows it to **generalize** from states it has visited to new, unseen states that are similar. It can estimate the value of a state it has never encountered before, breaking the need to visit every single state to learn a good policy. This ability to generalize is what makes solving high-dimensional RL problems possible.

29. **Describe the key challenge that POMDPs introduce and why they are important for real-world applications like medical diagnosis.**

Answer

The key challenge introduced by Partially Observable Markov Decision Processes (POMDPs) is **state estimation under uncertainty**. Unlike in an MDP where the agent knows the true state of the world, in a POMDP the agent must infer the state from a stream of noisy or incomplete observations. This adds a significant layer of complexity, as the agent must simultaneously:

(a) **Track Beliefs:** Maintain and update a probability distribution over possible hidden states (the belief state).

(b) **Plan Actions:** Make decisions based on this uncertain belief, where actions might be taken not just to gain reward, but also to gain information and reduce uncertainty (e.g., performing a test).

This makes the problem much harder, as the policy must operate over a continuous, high-dimensional belief space.
POMDPs are crucial for real-world applications like **medical diagnosis** because these problems are inherently partially observable:

- **Hidden State:** A patient's true underlying health condition or disease is a hidden state.
- **Observations:** A doctor receives only partial observations, such as symptoms, patient history, and lab test results. These observations are only probabilistically related to the true disease.
- **Actions and Information Gathering:** The doctor's actions are tests or treatments. A key part of the process is choosing actions (like ordering an MRI) specifically to gather information and update their "belief" about the patient's true condition, before committing to a more invasive treatment.

The POMDP framework provides a principled way to model this decision-making process under uncertainty, balancing the cost and risk of procedures against the information they provide to derive optimal diagnostic and treatment policies.