# Questions and Explanations on Modern Policy Gradient Methods

Taha Majlesi

July 17, 2025

**Abstract**

This document contains a comprehensive set of questions based on the text ”A Theoretical Exposition of Modern Policy Gradient Methods: From REINFORCE to Soft Actor-Critic”. It is divided into two parts: a multiple-choice section with 80 questions to test foundational knowledge, and a section with 30 detailed explanatory questions designed to deepen understanding of the core theoretical concepts. Complete answers are provided for all questions.

## Contents

Part I

# Multiple-Choice Questions

## 1 Questions

1. What is the primary objective function, $J(\theta)$, that policy gradient methods aim to maximize?

   A. The immediate reward at a single time step.
   B. The expected total discounted reward over trajectories.
   C. The probability of reaching a goal state.
   D. The entropy of the policy.

2. What mathematical identity is known as the "log-derivative trick"?

   A. $\nabla_x f(x) = \log(f(x))$
   B. $\nabla_x \log f(x) = f(x)\nabla_x f(x)$
   C. $\nabla_x f(x) = f(x)\nabla_x \log f(x)$
   D. $\log(\nabla_x f(x)) = \nabla_x \log f(x)$

3. Why is the Policy Gradient Theorem significant for reinforcement learning?

   A. It requires full knowledge of the environment's dynamics.
   B. It eliminates the need for a discount factor.
   C. It allows computing the policy gradient without knowing the environment's transition model.
   D. It guarantees that the policy will always improve.

4. The REINFORCE algorithm is also known as:

   A. Q-Learning
   B. Monte Carlo Policy Gradients
   C. Temporal Difference Learning
   D. Dynamic Programming

5. What is the most significant practical drawback of the vanilla REINFORCE algorithm?

   A. It is computationally expensive.
   B. It has extremely high variance.
   C. It is biased.
   D. It only works in discrete action spaces.

6. The principle of "causality" in policy gradients implies that:

A. The initial state determines the entire trajectory.
B. An action can only influence future rewards, not past ones.
C. All actions in a trajectory should receive the same credit.
D. The policy must be deterministic.

7. What is the "reward-to-go", $G_t$?

   A. The sum of all rewards in a trajectory.
   B. The sum of rewards from the beginning of the episode up to time $t$.
   C. The sum of rewards from time step $t$ to the end of the episode.
   D. The maximum possible reward from state $s_t$.

8. How does using reward-to-go improve upon the vanilla REINFORCE algorithm?

   A. It introduces bias to speed up learning.
   B. It makes the algorithm off-policy.
   C. It reduces variance by improving credit assignment.
   D. It eliminates the need for a learning rate.

9. What is the purpose of subtracting a baseline $b(s_t)$ from the reward-to-go?

   A. To introduce bias.
   B. To change the objective function.
   C. To further reduce the variance of the gradient estimate.
   D. To guarantee convergence.

10. What is the optimal choice for a baseline $b(s_t)$ to minimize variance?

    A. A constant value, $c$.
    B. The average reward of the episode.
    C. The state-value function, $V^\pi(s_t)$.
    D. The action-value function, $Q^\pi(s_t, a_t)$.

11. The Advantage Function, $A^\pi(s_t, a_t)$, is formally defined as:

    A. $Q^\pi(s_t, a_t) + V^\pi(s_t)$
    B. $R(\tau) - V^\pi(s_t)$
    C. $G_t$
    D. $Q^\pi(s_t, a_t) - V^\pi(s_t)$

12. Does subtracting a state-dependent baseline $b(s_t)$ introduce bias into the policy gradient estimate?

    A. Yes, always.
    B. Only if the baseline is negative.
    C. No, the expectation of the subtracted term is zero.

D. Yes, but it is negligible.

13. The term $\nabla_\theta \log \pi_\theta(a_t|s_t)$ is often called the:

    A. Value function
    B. Score function
    C. Reward function
    D. Policy function

14. In the REINFORCE algorithm, how are actions that lead to a high total reward treated?

    A. Their probabilities are decreased.
    B. Their probabilities are increased.
    C. Their probabilities are unchanged.
    D. They are removed from the policy.

15. The policy $\pi_\theta(a_t|s_t)$ is typically represented by:

    A. A lookup table
    B. A linear function
    C. A neural network
    D. A set of rules

16. The discount factor $\gamma$ is used to:

    A. Make future rewards less valuable than immediate rewards.
    B. Increase the variance of the returns.
    C. Ensure the policy is stochastic.
    D. Normalize the rewards to be between 0 and 1.

17. A trajectory $\tau$ is a sequence of:

    A. States and rewards
    B. Actions and rewards
    C. States and actions
    D. States, actions, and rewards

18. The REINFORCE update rule $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$ is an instance of:

    A. Gradient descent
    B. Newton's method
    C. Gradient ascent
    D. The Bellman equation

19. Why does vanilla REINFORCE have flawed credit assignment?

A. It only rewards the last action.

B. It assigns the same total return to every action in an episode.

C. It ignores negative rewards.

D. It cannot handle discounted rewards.

20. Algorithms that learn a value function (critic) to reduce variance in a policy gradient method (actor) are called:

A. Q-Learning methods

B. Actor-Critic methods

C. Monte Carlo methods

D. Sarsa methods

21. How can policy gradient methods be viewed from the perspective of classic Policy Iteration (PI)?

A. As a completely unrelated method.

B. As a method that only performs policy evaluation.

C. As a stochastic, approximate version of PI.

D. As a method that only performs policy improvement.

22. What does the Performance Difference Lemma express?

A. The difference in performance between two states.

B. The difference in performance between two actions.

C. The exact change in expected return when switching to a new policy.

D. The error bound of the value function approximation.

23. According to the Performance Difference Lemma, the performance difference $J(\theta') - J(\theta)$ is an expectation over trajectories sampled from which policy?

A. The old policy, $\pi_\theta$.

B. The new policy, $\pi_{\theta'}$.

C. A uniform random policy.

D. An optimal policy, $\pi^*$.

24. What is the "state distribution mismatch" problem?

A. The initial state distribution is unknown.

B. The states are not distributed normally.

C. The advantage function is calculated using the old policy's data, but the performance difference depends on the new policy's state distribution.

D. The policy parameters do not match the state representation.

25. What statistical technique is used to handle the mismatch between sampling and target distributions?

A. Bootstrapping

B. Regularization

C. Importance Sampling (IS)

D. Principal Component Analysis (PCA)

26. In the context of policy gradients, the importance sampling ratio is:

    A. $\frac{p(s_t)}{p'(s_t)}$

    B. $\frac{A^{\pi_\theta}(s_t,a_t)}{A^{\pi_{\theta'}}(s_t,a_t)}$

    C. $\frac{\pi_{\theta'}(a_t|s_t)}{\pi_\theta(a_t|s_t)}$

    D. $\frac{r(s_t,a_t)}{V^\pi(s_t)}$

27. What critical approximation is made to create a tractable surrogate objective function $L_{\pi_\theta}(\pi_{\theta'})$?

    A. The rewards are assumed to be constant.

    B. The advantage function is assumed to be zero.

    C. The state visitation distributions of the old and new policies are assumed to be approximately equal.

    D. The discount factor is assumed to be 1.

28. Taking a very large optimization step on the surrogate objective can be harmful because:

    A. It is computationally expensive.

    B. It can violate the assumption that the new and old policies are close, leading to a performance drop.

    C. It always leads to overfitting.

    D. It introduces bias into the estimate.

29. Trust region methods like TRPO guarantee monotonic policy improvement by:

    A. Using a very small learning rate.

    B. Constraining the change between the old and new policy, often using KL-divergence.

    C. Only updating the policy if the reward is positive.

    D. Averaging the policy parameters over many episodes.

30. The lower bound on performance improvement in trust region methods involves the surrogate objective and a penalty term based on:

    A. The L2 norm of the policy parameters.

    B. The maximum reward in the environment.

    C. The KL-divergence between the old and new policies.

    D. The number of steps in the episode.

31. The "moving target" problem in the Performance Difference Lemma refers to the fact that:

A. The environment's dynamics can change over time.
B. The reward function is non-stationary.
C. The state distribution we need to evaluate depends on the policy we are trying to find.
D. The optimal policy is constantly changing.

32. On-policy algorithms like REINFORCE are sample-inefficient because:

A. They require a model of the environment.
B. They can only learn from the most recent trajectory.
C. They discard old data and require new samples for every gradient update.
D. They use a very small learning rate.

33. The surrogate objective $L_{\pi_\theta}(\pi_{\theta'})$ is useful because it can be estimated using data from which policy?

A. The new policy $\pi_{\theta'}$.
B. The old policy $\pi_\theta$.
C. A random policy.
D. The optimal policy $\pi^*$.

34. The gradient of the surrogate objective $L_{\pi_\theta}(\pi_{\theta'})$ at $\theta' = \theta$ is equal to:

A. Zero.
B. The standard policy gradient.
C. The advantage function.
D. The learning rate $\alpha$.

35. The term $D_{\mathrm{KL}}^{\max}(\pi_\theta, \pi_{\theta'})$ represents:

A. The minimum KL-divergence between the policies.
B. The average KL-divergence over a trajectory.
C. The maximum KL-divergence between the policies over all states.
D. The KL-divergence at the initial state.

36. The policy improvement step in classic Policy Iteration involves acting greedily with respect to what?

A. The reward function.
B. The policy parameters.
C. The state-visitation frequency.
D. The value function of the current policy.

37. The Performance Difference Lemma provides a formal link between policy gradient methods and:

A. Supervised learning.
B. Dynamic programming and policy iteration.
C. Unsupervised clustering.
D. Bayesian inference.

38. Off-policy methods are generally more sample-efficient because they can:
    - A. Learn from data generated by older policies (e.g., from a replay buffer).
    - B. Use larger neural networks.
    - C. Converge in a single step.
    - D. Ignore the discount factor.

39. The error bound on the state distribution mismatch $|p_{\theta'}(s_t) - p_\theta(s_t)|$ grows with:
    - A. The discount factor $\gamma$.
    - B. The time step $t$.
    - C. The number of states.
    - D. The size of the action space.

40. The core idea of TRPO is to maximize the surrogate objective subject to a:
    - A. Constraint on the policy update size.
    - B. Constraint on the value function error.
    - C. Penalty on reward variance.
    - D. Penalty on episode length.

41. What is the core modification to the standard RL objective in the maximum entropy framework?
    - A. Subtracting an entropy term.
    - B. Adding a policy entropy term.
    - C. Squaring the reward.
    - D. Taking the logarithm of the reward.

42. What does the temperature parameter $\alpha$ control in the maximum entropy objective?
    - A. The learning rate of the actor.
    - B. The discount factor.
    - C. The relative importance of the entropy bonus versus the reward.
    - D. The size of the replay buffer.

43. A higher value of $\alpha$ in SAC encourages the policy to be more:
    - A. Deterministic and exploitative.
    - B. Stochastic and exploratory.
    - C. Biased.
    - D. Dependent on the initial state.

44. Which of the following is NOT a benefit of the maximum entropy framework?
    - A. Enhanced exploration.
    - B. Guaranteed convergence in a single policy update.
    - C. Improved robustness and transferability.
    - D. Smoother and more stable training dynamics.

45. The value functions in the maximum entropy framework are referred to as:

A. Hard value functions.
B. Deterministic value functions.
C. Optimal value functions.
D. Soft value functions.

46. The soft state-value function $V^{\text{soft}}(s_t)$ is defined as the expectation of:

A. $Q^{\text{soft}}(s_t, a_t) + \alpha \log \pi(a_t|s_t)$
B. $Q^{\text{soft}}(s_t, a_t) - \alpha \log \pi(a_t|s_t)$
C. $r(s_t, a_t)$
D. $\gamma V^{\text{soft}}(s_{t+1})$

47. Soft Actor-Critic (SAC) is what type of algorithm?

A. On-policy, model-based
B. Off-policy, model-free
C. On-policy, model-free
D. Off-policy, model-based

48. Why does SAC use two Q-function networks (critics)?

A. To double the learning speed.
B. To handle two different reward signals.
C. To counteract the tendency of Q-learning to overestimate values.
D. To learn two different policies simultaneously.

49. What is the purpose of the target value network $V_{\bar{\psi}}(s)$ in SAC?

A. To provide a stable, slowly changing target for Q-function updates.
B. To directly represent the policy.
C. To estimate the immediate reward.
D. To calculate the entropy of the policy.

50. How are the parameters of the target network $\bar{\psi}$ updated?

A. By direct gradient descent on the policy loss.
B. They are copied directly from the main network every step.
C. Using an exponential moving average (EMA) of the main network's parameters.
D. They are kept fixed throughout training.

51. The target for the Q-function loss $J_Q(\theta)$ in SAC is:

A. $r(s_t, a_t) + \gamma V_{\psi}(s_{t+1})$
B. $r(s_t, a_t) + \gamma V_{\bar{\psi}}(s_{t+1})$
C. $r(s_t, a_t)$
D. $V_{\bar{\psi}}(s_t)$

52. The policy loss $J_\pi(\phi)$ in SAC aims to make the policy choose actions that have a high:

A. Soft Q-value, as estimated by the critics.

B. Immediate reward.
C. State value.
D. Probability, regardless of value.

53. What technique does SAC use to get a low-variance gradient estimate for the policy update?

    A. The log-derivative trick.
    B. Importance sampling.
    C. The reparameterization trick.
    D. Using a baseline.

54. The use of a replay buffer in SAC is possible because it is:

    A. An on-policy algorithm.
    B. A model-based algorithm.
    C. An off-policy algorithm.
    D. A tabular algorithm.

55. The technique of using the minimum of two Q-networks is inspired by which algorithm?

    A. REINFORCE
    B. A2C (Advantage Actor-Critic)
    C. TRPO (Trust Region Policy Optimization)
    D. Clipped Double Q-Learning

56. The entropy term $\mathcal{H}(\pi(\cdot|s_t))$ acts as a form of:

    A. Objective function
    B. Regularizer
    C. Discount factor
    D. Reward signal

57. In the SAC architecture, which component is the "actor"?

    A. The Q-function networks $Q_{\theta_i}$.
    B. The value function network $V_\psi$.
    C. The policy network $\pi_\phi$.
    D. The target value network $V_{\bar{\psi}}$.

58. The target for the value function loss $J_V(\psi)$ is the expected value of:

    A. The immediate reward.
    B. The discounted next state value.
    C. The soft Q-value minus the entropy term.
    D. The policy's log-probability.

59. The reparameterization trick works by expressing the action as:

    A. A sample from the policy distribution.

B. A deterministic function of the state and independent noise.
C. The action with the highest Q-value.
D. A random action from the action space.

60. What is the theoretical framework used to prove the convergence of SAC?

A. Standard Policy Iteration
B. Monte Carlo Tree Search
C. Soft Policy Iteration
D. Gradient Descent Analysis

61. The proof of convergence for Soft Policy Evaluation (Lemma 1) relies on showing that the soft Bellman backup operator is a:

A. Linear function
B. Contraction mapping
C. Convex function
D. Non-expansive mapping

62. To prove that the soft Bellman operator is a contraction, the proof recasts it into a standard Bellman operator by defining a:

A. Entropy-augmented reward
B. Discounted policy
C. Deterministic environment
D. Bounded action space

63. The convergence of an operator that is a contraction mapping is guaranteed by which theorem?

A. The Central Limit Theorem
B. The Banach Fixed-Point Theorem
C. Bayes' Theorem
D. The Policy Improvement Theorem

64. The Soft Policy Improvement step (Lemma 2) finds a new policy that minimizes the KL-divergence to what distribution?

A. A uniform distribution.
B. The previous policy.
C. A Gaussian distribution.
D. The exponentiated soft Q-function of the old policy.

65. The proof of Soft Policy Improvement shows that for the new policy $\pi_{\text{new}}$, its soft Q-function is:

A. Pointwise less than or equal to the old one.
B. Pointwise greater than or equal to the old one.
C. Completely independent of the old one.
D. Equal to the old one.

66. The overall convergence theorem for Soft Policy Iteration guarantees convergence to:

    A. A locally optimal policy.
    B. A deterministic policy.
    C. The optimal policy under the maximum entropy objective.
    D. A policy that is optimal for the standard RL objective.

67. The theoretical guarantees for SAC are primarily established for which case?

    A. Continuous action spaces with function approximation.
    B. The tabular case (finite states and actions).
    C. Environments with non-stationary dynamics.
    D. Model-based reinforcement learning.

68. The proof of Lemma 2 (Soft Policy Improvement) relies on the operator $\mathcal{T}^{\pi_{\text{new}}}$ being:

    A. A contraction
    B. Invertible
    C. A monotone operator
    D. A linear operator

69. The sequence of soft Q-functions $Q^{\pi_i}$ generated by Soft Policy Iteration is guaranteed to converge because it is:

    A. Strictly increasing and unbounded.
    B. A random walk.
    C. Monotonically non-decreasing and bounded from above.
    D. A geometric series.

70. The assumption of a finite action space in the proofs is important to ensure that:

    A. The policy is always stochastic.
    B. The reward function is bounded.
    C. The policy entropy is bounded.
    D. The state space is finite.

71. At convergence, the optimal policy $\pi^*$ is the one that is the fixed point of which process?

    A. The Bellman backup.
    B. The policy evaluation step.
    C. The policy improvement step.
    D. The gradient ascent update.

72. The proof of Lemma 1 shows that the soft Bellman operator is specifically a:

    A. 1-contraction
    B. $\alpha$-contraction
    C. $\gamma$-contraction

D. $\epsilon$-contraction

73. The key inequality established in the proof of Lemma 2 is $\mathbb{E}_{a_t \sim \pi_{\text{new}}}[\dots] \geq V^{\pi_{\text{old}}}(s_t)$. What does this inequality relate?

    A. The value of the new policy to the value of the old policy.
    B. The expected soft value under the new policy to the state value of the old policy.
    C. The reward of the new policy to the reward of the old policy.
    D. The entropy of the new policy to the entropy of the old policy.

74. How does the practical SAC algorithm relate to the theoretical Soft Policy Iteration?

    A. It is an exact implementation.
    B. It is an approximate version that uses function approximators and does not run the steps to convergence.
    C. It is a completely different algorithm that shares the same name.
    D. It only implements the Soft Policy Evaluation step.

75. The final step in the proof of Theorem 1 shows the converged policy $\pi^*$ is optimal by arguing that:

    A. Its value function is the largest possible.
    B. Any other policy would have a strictly lower soft Q-value.
    C. It satisfies the Bellman optimality equation.
    D. The gradient of its objective function is zero.

## 2 Answer Key

| **Q** | **Ans** | **Q** | **Ans** | **Q** | **Ans** | **Q** | **Ans** |
|---|---|---|---|---|---|---|---|
| 1 | B | 21 | C | 41 | B | 61 | C |
| 2 | C | 22 | C | 42 | C | 62 | B |
| 3 | C | 23 | B | 43 | B | 63 | A |
| 4 | B | 24 | C | 44 | B | 64 | B |
| 5 | B | 25 | C | 45 | D | 65 | D |
| 6 | B | 26 | C | 46 | B | 66 | B |
| 7 | C | 27 | C | 47 | B | 67 | C |
| 8 | C | 28 | B | 48 | C | 68 | B |
| 9 | C | 29 | B | 49 | A | 69 | C |
| 10 | C | 30 | C | 50 | C | 70 | C |
| 11 | D | 31 | C | 51 | B | 71 | C |
| 12 | C | 32 | C | 52 | A | 72 | C |
| 13 | B | 33 | B | 53 | C | 73 | C |
| 14 | B | 34 | B | 54 | C | 74 | B |
| 15 | C | 35 | C | 55 | D | 75 | B |
| 16 | A | 36 | D | 56 | B | 76 | B |
| 17 | C | 37 | B | 57 | C | 77 | *N/A* |
| 18 | C | 38 | A | 58 | C | 78 | *N/A* |
| 19 | B | 39 | B | 59 | B | 79 | *N/A* |
| 20 | B | 40 | A | 60 | B | 80 | *N/A* |

Table 1: Answer Key for Multiple-Choice Questions

Part II

# Explanatory Questions

1. **Explain the log-derivative trick and why it is essential for deriving the Policy Gradient Theorem.**

**Answer**

The log-derivative trick is a simple identity from calculus based on the derivative of the natural logarithm: $\nabla_x \log f(x) = \frac{\nabla_x f(x)}{f(x)}$. By rearranging this, we get the form used in policy gradients: $\nabla_x f(x) = f(x)\nabla_x \log f(x)$.
Its essential role in deriving the Policy Gradient Theorem is to solve the problem of differentiating an expectation where the distribution itself depends on the parameters we are differentiating with respect to. The derivation starts with the gradient of the objective function:

$$\nabla_\theta J(\theta) = \nabla_\theta \mathbb{E}_{\tau \sim p_\theta(\tau)}[R(\tau)] = \nabla_\theta \int p_\theta(\tau) R(\tau) d\tau$$

This leads to $\int (\nabla_\theta p_\theta(\tau)) R(\tau) d\tau$. The term $\nabla_\theta p_\theta(\tau)$ is problematic because we cannot estimate it from samples.
By applying the log-derivative trick, we transform this term:

$$\nabla_\theta p_\theta(\tau) = p_\theta(\tau) \nabla_\theta \log p_\theta(\tau)$$

Substituting this back into the integral gives:

$$\nabla_\theta J(\theta) = \int p_\theta(\tau)(\nabla_\theta \log p_\theta(\tau)) R(\tau) d\tau$$

This expression is now an expectation under the original distribution $p_\theta(\tau)$:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)}[(\nabla_\theta \log p_\theta(\tau)) R(\tau)]$$

This final form is crucial because it can be estimated using Monte Carlo sampling. We can run the policy $\pi_\theta$ to generate trajectories $\tau$ and then compute the sample mean of $(\nabla_\theta \log p_\theta(\tau))R(\tau)$. This allows us to estimate the gradient without needing to know or differentiate the environment dynamics, which is the core breakthrough of the theorem.

2. **Describe the problem of high variance in the REINFORCE algorithm and explain how using a reward-to-go and a baseline help to mitigate it.**

**Answer**

The high variance in REINFORCE stems from its gradient estimator: $\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}(\sum_{t=0}^{T-1}\nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}))R(\tau_i)$. The total return $R(\tau_i)$ is a sum of many stochastic rewards and is therefore a high-variance random variable. A single lucky or unlucky trajectory can result in a very large or small $R(\tau_i)$, causing the gradient estimate to fluctuate wildly between updates. This makes training unstable and slow.

Two main techniques mitigate this:

(a) **Reward-to-Go:** The vanilla estimator assigns the same credit, $R(\tau)$, to every action in the trajectory. This violates the principle of causality, as an action $a_t$ cannot affect rewards received before it $(r_0, \ldots, r_{t-1})$. These past rewards add noise to the credit assigned to $a_t$. By replacing the total return $R(\tau)$ with the reward-to-go $G_t = \sum_{t'=t}^{T} r_{t'}$, we only credit an action with the rewards that followed it. This removes the irrelevant noise from past rewards, reducing the variance of the estimator without introducing any bias.

(b) **Baseline:** Even with reward-to-go, the gradient estimate can still be high variance. For example, if all rewards are large and positive, every action will be reinforced, even if some were suboptimal. A baseline $b(s_t)$ shifts the values, centering them around a reference point. The gradient estimator becomes $\nabla_\theta J(\theta) \approx \mathbb{E}[\sum_t \nabla_\theta \log \pi_\theta(a_t|s_t)(G_t - b(s_t))]$. The learning signal now depends on whether the action led to a return that was *better or worse than average* (the baseline). The optimal baseline for variance reduction is the state-value function $V^\pi(s_t)$. The resulting term, $G_t - V^\pi(s_t)$, is an estimate of the advantage function $A^\pi(s_t, a_t)$, which has significantly lower variance than $G_t$ alone.

3. **Explain the state distribution mismatch problem as described in Chapter 2. Why does it pose a challenge for policy improvement?**

**Answer**

The state distribution mismatch problem is a core challenge in off-policy and policy iteration-style reinforcement learning. It arises from the Performance Difference Lemma:

$$J(\theta') - J(\theta) = \mathbb{E}_{\tau \sim p_{\theta'}(\tau)} \left[ \sum_{t=0}^{\infty} \gamma^t A^{\pi_\theta}(s_t, a_t) \right]$$

This equation tells us that the improvement in performance from switching to a new policy $\pi_{\theta'}$ depends on the advantage of the old policy, $A^{\pi_\theta}$, evaluated over states and actions visited by the *new* policy $\pi_{\theta'}$.

The challenge is that we want to find the best $\theta'$ to maximize this quantity, but the distribution of trajectories $p_{\theta'}(\tau)$ itself depends on $\theta'$. We cannot simply generate samples from $\pi_{\theta'}$ to compute the expectation, because $\pi_{\theta'}$ is the very policy we are trying to find. This creates a "moving target" problem: the optimization landscape changes as the policy changes.

To make this tractable, algorithms make a key approximation: they assume that for a small change in policy parameters (from $\theta$ to $\theta'$), the state visitation distribution does not change much ($p_{\theta'}(s_t) \approx p_\theta(s_t)$). This allows the creation of a *surrogate objective* that can be evaluated using samples from the *old* policy $\pi_\theta$. However, if the policy update is too large, this approximation breaks down, and maximizing the surrogate objective might actually lead to a decrease in true performance. This is why trust region methods, which explicitly limit the size of the policy update, are necessary for guaranteed improvement.

4. **What is the maximum entropy RL objective, and what is the role of the temperature parameter $\alpha$?**

**Answer**

The maximum entropy reinforcement learning objective modifies the standard RL objective by adding a term that rewards the policy for being more random, or having higher entropy. The objective is to find a policy $\pi$ that maximizes the expected sum of rewards and policy entropy:

$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(s_t,a_t)\sim\rho_\pi}[r(s_t, a_t) + \alpha\mathcal{H}(\pi(\cdot|s_t))]$$

Here, $\mathcal{H}(\pi(\cdot|s_t))$ is the entropy of the policy at state $s_t$.
The **temperature parameter** $\alpha$ is a non-negative scalar that controls the trade-off between the standard reward-seeking behavior and the entropy-seeking behavior.

- If $\alpha = 0$, the objective reduces to the standard RL objective of maximizing only the expected cumulative reward.
- As $\alpha \to \infty$, the reward term becomes negligible, and the agent's primary goal becomes maximizing its own randomness, leading to a uniform random policy.
- For intermediate values of $\alpha > 0$, the agent learns to find a balance. It seeks to maximize rewards while staying as random as possible. This encourages it to find all possible solutions to a task if multiple exist and to explore more broadly during training.

In essence, $\alpha$ determines how much the agent should prioritize exploration and stochasticity over pure exploitation of the learned rewards.

5. **Describe the four main neural network components of the Soft Actor-Critic (SAC) architecture and the role of each.**

**Answer**

The SAC algorithm uses several neural networks that are trained concurrently:

(a) **Policy Network (Actor) $\pi_\phi(a|s)$:** This is the actor. It takes a state $s$ as input and outputs the parameters of a stochastic policy distribution (e.g., the mean and standard deviation of a Gaussian). Its goal is to learn a policy that maximizes the soft (entropy-regularized) value function. It is trained to produce actions that have high soft Q-values.

(b) **Two Soft Q-Function Networks (Critics) $Q_{\theta_1}(s,a)$ and $Q_{\theta_2}(s,a)$:** These are the critics. Each network takes a state-action pair $(s,a)$ and outputs an estimate of the soft action-value function, $Q^{\text{soft}}(s,a)$. SAC uses two Q-networks and takes the minimum of their predictions during the policy update. This technique, known as clipped double Q-learning, helps to mitigate the problem of Q-value overestimation, which is a common source of instability in Q-learning-based methods.

(c) **A Soft Value Function Network $V_\psi(s)$:** This network takes a state $s$ and estimates the soft state-value function, $V^{\text{soft}}(s)$. While some versions of SAC omit this network and calculate its target directly, the version described in the text uses it. Its loss function is designed to make its output match the expected soft Q-value minus the entropy term, providing a target for the policy update.

(d) **A Target Value Function Network $V_{\bar{\psi}}(s)$:** This is a time-delayed copy of the main value network $V_\psi$. Its parameters $\bar{\psi}$ are not trained via gradient descent but are instead updated slowly as an exponential moving average (EMA) of the main value network's parameters ($\bar{\psi} \leftarrow \tau\psi + (1-\tau)\bar{\psi}$). Its purpose is to provide a stable target for the Q-function updates. By using a slowly changing target, it prevents the Q-learning updates from becoming unstable, which is a critical component for successful off-policy learning.

6. **Explain the proof of Lemma 1 (Soft Policy Evaluation). How is the problem transformed to prove convergence?**

**Answer**

Lemma 1 states that repeatedly applying the soft Bellman backup operator, $\mathcal{T}^\pi$, to any Q-function will cause it to converge to the true soft Q-function, $Q^\pi$. The proof relies on the Banach fixed-point theorem, which requires showing that $\mathcal{T}^\pi$ is a contraction mapping.
The soft Bellman backup operator is:

$$\mathcal{T}^\pi Q(s_t, a_t) = r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1}\sim p, a_{t+1}\sim\pi}[Q(s_{t+1}, a_{t+1}) - \alpha \log \pi(a_{t+1}|s_{t+1})]$$

The proof elegantly transforms this into a familiar form.

(a) **Isolate the Entropy Term:** The expectation can be split:

$$\mathcal{T}^\pi Q(s_t, a_t) = r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1}\sim p}\left[\mathbb{E}_{a_{t+1}\sim\pi}[Q(s_{t+1}, a_{t+1})] - \alpha \mathbb{E}_{a_{t+1}\sim\pi}[\log \pi(a_{t+1}|s_{t+1})]\right]$$

Recognizing that $\mathcal{H}(\pi(\cdot|s_{t+1})) = \mathbb{E}_{a_{t+1}\sim\pi}[-\log \pi(a_{t+1}|s_{t+1})]$, this becomes:

$$\mathcal{T}^\pi Q(s_t, a_t) = r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1}\sim p}[\alpha \mathcal{H}(\pi(\cdot|s_{t+1}))] + \gamma \mathbb{E}_{s_{t+1}\sim p, a_{t+1}\sim\pi}[Q(s_{t+1}, a_{t+1})]$$

(b) **Define an Entropy-Augmented Reward:** The first two terms depend only on the current state-action pair and the next state distribution, not on the Q-function being evaluated. They can be grouped into a new, "soft" reward function:

$$r_{\text{soft}}(s_t, a_t) \triangleq r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1}\sim p}[\alpha \mathcal{H}(\pi(\cdot|s_{t+1}))]$$

(c) **Apply Standard Convergence Results:** With this new reward, the operator simplifies to:

$$\mathcal{T}^\pi Q(s_t, a_t) = r_{\text{soft}}(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1}\sim p, a_{t+1}\sim\pi}[Q(s_{t+1}, a_{t+1})]$$

This is now identical in form to the standard Bellman evaluation operator for a policy $\pi$ with a reward function $r_{\text{soft}}$. It is a well-known result that the standard Bellman operator is a $\gamma$-contraction in the infinity norm (provided rewards are bounded). Since the action space is assumed finite, the entropy $\mathcal{H}$ is bounded, making $r_{\text{soft}}$ bounded. Therefore, $\mathcal{T}^\pi$ is also a $\gamma$-contraction, and by the Banach fixed-point theorem, it is guaranteed to converge to a unique fixed point, which is $Q^\pi$.

7. **Walk through the proof of Lemma 2 (Soft Policy Improvement), explaining the key inequality and how it leads to the final result.**

**Answer**

Lemma 2 proves that the policy improvement step in Soft Policy Iteration yields a new policy $\pi_{\text{new}}$ whose soft Q-value is pointwise greater than or equal to that of the old policy $\pi_{\text{old}}$. The new policy is defined as the one that minimizes the KL-divergence to the exponentiated soft Q-function of the old policy:

$$\pi_{\text{new}} = \arg\min_{\pi' \in \Pi} D_{\text{KL}}\left(\pi'(\cdot|s_t) \middle\| \frac{\exp(\frac{1}{\alpha}Q^{\pi_{\text{old}}}(s_t, \cdot))}{Z^{\pi_{\text{old}}}(s_t)}\right)$$

The proof has two main stages:

(a) **Establish a Key Inequality:** Because $\pi_{\text{new}}$ is the minimizer of the KL-divergence, its objective value must be less than or equal to the value for any other policy, including $\pi_{\text{old}}$. This gives:

$$D_{\text{KL}}(\pi_{\text{new}} \| \dots) \leq D_{\text{KL}}(\pi_{\text{old}} \| \dots)$$

Expanding the definition of KL-divergence and simplifying leads to the key inequality:

$$\mathbb{E}_{a_t \sim \pi_{\text{new}}}\left[Q^{\pi_{\text{old}}}(s_t, a_t) - \alpha \log \pi_{\text{new}}(a_t|s_t)\right] \geq \mathbb{E}_{a_t \sim \pi_{\text{old}}}\left[Q^{\pi_{\text{old}}}(s_t, a_t) - \alpha \log \pi_{\text{old}}(a_t|s_t)\right]$$

The right-hand side is exactly the definition of the soft state-value function $V^{\pi_{\text{old}}}(s_t)$. So, we have:

$$\mathbb{E}_{a_t \sim \pi_{\text{new}}}\left[Q^{\pi_{\text{old}}}(s_t, a_t) - \alpha \log \pi_{\text{new}}(a_t|s_t)\right] \geq V^{\pi_{\text{old}}}(s_t)$$

(b) **Apply the Inequality Iteratively:** We start with the soft Bellman equation for $Q^{\pi_{\text{old}}}$:

$$Q^{\pi_{\text{old}}}(s_t, a_t) = r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim p}[V^{\pi_{\text{old}}}(s_{t+1})]$$

We can substitute our key inequality into the $V^{\pi_{\text{old}}}(s_{t+1})$ term:

$$Q^{\pi_{\text{old}}}(s_t, a_t) \leq r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim p}\left[\mathbb{E}_{a_{t+1} \sim \pi_{\text{new}}}[Q^{\pi_{\text{old}}}(s_{t+1}, a_{t+1}) - \alpha \log \pi_{\text{new}}(a_{t+1}|s_{t+1})]\right]$$

The right-hand side is exactly the definition of the soft Bellman backup operator for the *new* policy, $\mathcal{T}^{\pi_{\text{new}}}$, applied to the *old* Q-function, $Q^{\pi_{\text{old}}}$. This gives us $Q^{\pi_{\text{old}}} \leq \mathcal{T}^{\pi_{\text{new}}} Q^{\pi_{\text{old}}}$.

Since $\mathcal{T}^{\pi_{\text{new}}}$ is a monotone operator, we can apply it repeatedly to both sides of the inequality. From Lemma 1, we know that this repeated application $(\mathcal{T}^{\pi_{\text{new}}})^k Q^{\pi_{\text{old}}}$ converges to $Q^{\pi_{\text{new}}}$. Therefore, we conclude that $Q^{\pi_{\text{old}}} \leq Q^{\pi_{\text{new}}}$, proving that the policy improvement step leads to a monotonically non-decreasing Q-function.

8. **What is the reparameterization trick and why is it important for training the actor in SAC?**

**Answer**

The reparameterization trick is a method for computing low-variance gradients of a stochastic node in a computation graph. In SAC, the actor network $\pi_\phi$ outputs a distribution (e.g., a Gaussian with mean $\mu_\phi(s)$ and standard deviation $\sigma_\phi(s)$). To get an action, we would normally sample from this distribution: $a \sim \mathcal{N}(\mu_\phi(s), \sigma_\phi(s))$. This sampling operation is stochastic and non-differentiable, which means we cannot backpropagate gradients through it to update the actor's parameters $\phi$.

The reparameterization trick bypasses this by reframing the sampling process. Instead of sampling the action directly, we sample a random noise variable from a fixed, simple distribution (e.g., $\epsilon \sim \mathcal{N}(0, I)$) and then transform this noise into an action using a deterministic function that depends on the policy parameters. For a Gaussian policy, this would be:

$$a_t = \mu_\phi(s_t) + \sigma_\phi(s_t) \odot \epsilon$$

Here, the stochasticity is injected by $\epsilon$, which is external to the computation graph involving $\phi$. The path from the parameters $\phi$ to the final action $a_t$ is now fully deterministic.

This is crucial for SAC's policy loss:

$$J_\pi(\phi) = \mathbb{E}_{s_t,\epsilon}\left[\alpha \log(\pi_\phi(a_t(\epsilon, s_t)|s_t)) - Q_\theta(s_t, a_t(\epsilon, s_t))\right]$$

Because $a_t$ is now a deterministic function of $\phi$, we can easily backpropagate the gradient from the Q-function (critic) and the log-probability term all the way back to the actor's parameters $\phi$. This provides a direct, low-variance path for the critic to "tell" the actor how to adjust its output to produce higher-value actions.

9. **Compare and contrast on-policy and off-policy learning. Why is SAC being off-policy a major advantage?**

Answer

**On-Policy Learning:**

- **Definition:** The agent learns a policy and improves it using data collected from that *same* policy.
- **Example:** REINFORCE, A2C.
- **Mechanism:** After each policy update, all previously collected data is discarded. New data must be generated by executing the newly updated policy in the environment.
- **Pros:** Simpler to implement and analyze, as the behavior and target policies are the same.
- **Cons:** Extremely sample-inefficient. The agent can only learn from the experience it is currently gathering, which can be very slow, especially in complex environments.

**Off-Policy Learning:**

- **Definition:** The agent learns a target policy using data collected from a different policy, known as the behavior policy.
- **Example:** Q-Learning, SAC.
- **Mechanism:** Experience can be stored in a large dataset called a replay buffer. The behavior policy might be an older version of the target policy or a more exploratory policy. The learning algorithm can then sample mini-batches from this buffer to update the target policy.
- **Pros:** Highly sample-efficient. The agent can reuse old experiences many times, learning from a diverse set of past behaviors. This dramatically reduces the number of environment interactions needed.
- **Cons:** Can be more complex and less stable due to the mismatch between the behavior and target distributions (requiring techniques like importance sampling or stable target networks).

**SAC's Advantage:** SAC being an off-policy algorithm is a major advantage primarily due to its **sample efficiency**. By using a replay buffer, SAC can learn from each piece of experience multiple times. This is especially critical in real-world applications (like robotics) where interacting with the environment is expensive or time-consuming. The ability to reuse past data allows SAC to learn effective policies with far fewer environmental steps compared to on-policy methods like REINFORCE or TRPO.

10. **Explain the role of the surrogate objective $L_{\pi_\theta}(\pi_{\theta'})$ in policy gradient methods and its relationship to the true performance improvement $J(\theta') - J(\theta)$.**

**Answer**

The surrogate objective $L_{\pi_\theta}(\pi_{\theta'})$ is a practical approximation of the true performance improvement that an agent gets by switching from policy $\pi_\theta$ to $\pi_{\theta'}$. The true performance improvement is given by the Performance Difference Lemma:

$$J(\theta') - J(\theta) = \mathbb{E}_{\tau\sim p_{\theta'}(\tau)}\left[\sum_t \gamma^t A^{\pi_\theta}(s_t, a_t)\right]$$

This is intractable to optimize directly because the expectation is over the unknown new policy's trajectory distribution, $p_{\theta'}(\tau)$.
To create a tractable objective, two steps are taken:

(a) **Importance Sampling:** The expectation over actions is switched from the new policy to the old policy by introducing an importance weight:

$$\mathbb{E}_{s_t\sim p_{\theta'}(s_t)}\left[\mathbb{E}_{a_t\sim\pi_\theta(a_t|s_t)}\left[\frac{\pi_{\theta'}(a_t|s_t)}{\pi_\theta(a_t|s_t)}\gamma^t A^{\pi_\theta}(s_t, a_t)\right]\right]$$

(b) **Approximation:** The state distribution mismatch is ignored by assuming $p_{\theta'}(s_t) \approx p_\theta(s_t)$.

This leads to the surrogate objective:

$$L_{\pi_\theta}(\pi_{\theta'}) = \mathbb{E}_{\tau\sim p_\theta(\tau)}\left[\sum_t \gamma^t \frac{\pi_{\theta'}(a_t|s_t)}{\pi_\theta(a_t|s_t)} A^{\pi_\theta}(s_t, a_t)\right]$$

The key relationship is that $L_{\pi_\theta}(\pi_{\theta'})$ is an approximation of $J(\theta') - J(\theta)$ that can be estimated using data collected from the old policy $\pi_\theta$. Furthermore, the gradients of both objectives are identical at the point $\theta' = \theta$. This means that a small step in the direction of the gradient of $L$ is also a step in the direction of the gradient of $J$.
However, for large steps, the approximation $p_{\theta'}(s_t) \approx p_\theta(s_t)$ breaks down. Trust region methods address this by showing that the true improvement is lower-bounded by the surrogate objective minus a penalty term for the policy change: $J(\theta') - J(\theta) \geq L_{\pi_\theta}(\pi_{\theta'}) - C \cdot D_{\text{KL}}^{\max}(\pi_\theta, \pi_{\theta'})$. This provides a principled way to maximize the surrogate objective while guaranteeing true policy improvement.

11. **What is the Advantage Function, and why is it a better learning signal than the raw reward-to-go?**

**Answer**

The Advantage Function $A^\pi(s_t, a_t)$ measures how much better a specific action $a_t$ is compared to the average action that would be taken in state $s_t$ under policy $\pi$. It is formally defined as:

$$A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t)$$

where $Q^\pi(s_t, a_t)$ is the action-value (the expected return after taking action $a_t$ in state $s_t$) and $V^\pi(s_t)$ is the state-value (the average expected return from state $s_t$).
The advantage function is a better learning signal than the raw reward-to-go ($G_t$, which is a sample-based estimate of $Q^\pi$) for a crucial reason related to variance reduction. Consider an environment where all rewards are positive and large (e.g., always between 100 and 110). Using the reward-to-go $G_t$ as the learning signal would mean that every action is reinforced (its probability is increased), because $G_t$ will always be positive. This doesn't effectively distinguish between good and bad actions; it only distinguishes between actions that lead to high returns and actions that lead to slightly less high returns.
By subtracting the state-value $V^\pi(s_t)$ (the baseline), the advantage function centers the learning signal around zero.

- If $A^\pi(s_t, a_t) > 0$, it means action $a_t$ was better than the average action for state $s_t$. The policy will be updated to increase the probability of taking $a_t$.
- If $A^\pi(s_t, a_t) < 0$, it means action $a_t$ was worse than average. The policy will be updated to decrease its probability.

This provides a much more stable and intuitive learning signal, significantly reducing the variance of the gradient estimate and leading to faster and more reliable convergence.

12. **Explain the policy improvement step in Soft Policy Iteration. Why is it defined in terms of minimizing a KL-divergence?**

**Answer**

In Soft Policy Iteration, the policy improvement step aims to find a new policy $\pi_{\text{new}}$ that is optimal with respect to the current soft Q-function, $Q^{\pi_{\text{old}}}$. This new policy is defined as:

$$\pi_{\text{new}} = \arg\min_{\pi' \in \Pi} D_{\text{KL}} \left( \pi'(\cdot|s_t) \middle\| \frac{\exp(\frac{1}{\alpha} Q^{\pi_{\text{old}}}(s_t, \cdot))}{Z^{\pi_{\text{old}}}(s_t)} \right)$$

This formulation has a deep connection to the maximum entropy objective. The objective is to find a policy that maximizes reward while also maximizing its own entropy. The expression $\frac{1}{Z}\exp(\frac{1}{\alpha}Q)$ defines a Boltzmann or softmax distribution. This distribution has a key property: it is the distribution with the highest possible entropy among all distributions that achieve a certain expected energy (in this case, expected Q-value).
Therefore, defining the policy improvement step as minimizing the KL-divergence to this Boltzmann distribution is equivalent to finding a new policy $\pi_{\text{new}}$ that perfectly balances the two goals of the maximum entropy objective:

(a) **Exploitation:** It assigns higher probability to actions with higher soft Q-values, as dictated by the $Q^{\pi_{\text{old}}}$ term in the exponent.

(b) **Exploration:** By matching the shape of a Boltzmann distribution, it retains as much entropy as possible for the given Q-values. The temperature $\alpha$ controls how "peaked" or "flat" this distribution is, directly controlling the trade-off.

This formulation is not just an arbitrary choice; it is the principled way to derive the policy that is optimal for the soft value function of the previous iteration, ensuring that the policy improvement step makes progress towards the overall maximum entropy objective.

13. **Why does SAC use target networks, and how does their slow update via EMA contribute to stability?**

**Answer**

SAC uses target networks to stabilize the training of its critic (Q-function) networks. The loss function for the Q-networks is a mean-squared error loss based on the soft Bellman equation:

$$J_Q(\theta_i) = \mathbb{E}\left[\frac{1}{2}\left(Q_{\theta_i}(s_t, a_t) - \underbrace{(r(s_t, a_t) + \gamma V_{\bar{\psi}}(s_{t+1}))}_{\text{Target Value}}\right)^2\right]$$

The "Target Value" term is what the Q-network is trying to predict. The problem is that this target depends on the value function $V$, which is also being learned and updated at the same time. If we were to use the main value network $V_\psi$ in the target, we would have a "moving target" problem. As the Q-network parameters $\theta_i$ are updated, the value network parameters $\psi$ would also be updated, which in turn changes the target for the Q-network. This can lead to oscillations and instability, where the Q-network is chasing a target that is constantly running away.
To solve this, SAC uses a **target value network** $V_{\bar{\psi}}$. This network is a copy of the main value network, but its parameters $\bar{\psi}$ are updated much more slowly using an exponential moving average (EMA):

$$\bar{\psi} \leftarrow \tau\psi + (1 - \tau)\bar{\psi} \quad \text{(where } \tau \text{ is small, e.g., 0.005)}$$

This slow update has a stabilizing effect:

- It keeps the target value fixed for several gradient updates of the Q-network.
- It makes the target change slowly and smoothly over time.

By providing a stable, slowly-drifting target, the target network breaks the feedback loop and allows the Q-function to converge more reliably towards a consistent objective. This is a crucial technique for achieving stable learning in off-policy algorithms like SAC and DQN.

14. **What is the purpose of using two Q-networks in SAC and taking the minimum of their values?**

**Answer**

The use of two Q-networks (a technique often called Clipped Double Q-Learning) is a direct solution to the problem of **Q-value overestimation** in actor-critic and Q-learning algorithms.

**The Problem:** In many function approximation settings, the Q-network can learn erroneously high values for certain state-action pairs. In a standard actor-critic setup, the actor is trained to choose actions that maximize the Q-value. If the critic has an erroneously high "spike" in its Q-value estimate for a particular action, the actor will quickly learn to exploit this error, reinforcing a suboptimal action simply because the critic is wrong. This can lead to a catastrophic feedback loop where the critic's error reinforces a bad policy, which in turn can further destabilize the critic's estimates.

**The Solution:** SAC uses two independent Q-networks, $Q_{\theta_1}$ and $Q_{\theta_2}$, which are trained on the same data but initialized differently. Due to their different initializations and the stochasticity of training, they will develop different errors. It is unlikely that both networks will overestimate the value for the same state-action pair simultaneously.

When updating the policy and the value function, SAC uses the **minimum** of the two Q-values as the target:

$$\text{Target for V-update} \sim \min(Q_{\theta_1}, Q_{\theta_2}) - \alpha \log \pi$$

$$\text{Policy Loss} \sim \alpha \log \pi - \min(Q_{\theta_1}, Q_{\theta_2})$$

By taking the minimum, SAC adopts a pessimistic or conservative estimate of the value. This makes it much harder for the actor to exploit an estimation error from a single critic, as the other critic will likely provide a lower, more reasonable estimate. This clipping action effectively smooths out the Q-value landscape that the actor sees, preventing it from latching onto spurious peaks and leading to significantly more stable and effective training.

15. **How does the theoretical framework of Soft Policy Iteration (SPI) provide a foundation for the practical SAC algorithm?**

**Answer**

Soft Policy Iteration (SPI) provides the theoretical justification for SAC by proving that its underlying algorithmic structure is sound and guaranteed to converge to the optimal policy (in the tabular case). The practical SAC algorithm can be seen as an approximate, scaled-up version of SPI.
The connection is as follows:

(a) **Soft Policy Evaluation in SPI:** This step involves iterating the soft Bellman backup operator until the Q-function converges. **SAC's Approximation:** SAC performs an approximate version of this by training the critic networks ($Q_{\theta_1}, Q_{\theta_2}$) to minimize the mean-squared Bellman error using samples from a replay buffer. It doesn't run this to convergence in each step but instead performs a few gradient steps. The use of a target network helps stabilize this approximate evaluation.

(b) **Soft Policy Improvement in SPI:** This step involves finding a new policy that minimizes the KL-divergence to the exponentiated Q-function of the current policy. **SAC's Approximation:** SAC performs an approximate version of this by updating the actor network ($\pi_\phi$) with a few gradient steps on a loss function that is derived directly from this KL-divergence objective.

(c) **Alternation:** SPI strictly alternates between full evaluation and full improvement. SAC interleaves these steps, performing a single gradient update for the actor, critics, and value function on each mini-batch of data.

The convergence proof of SPI (Theorem 1) shows that this iterative process of evaluation and improvement is guaranteed to be monotonic and will converge to the optimal policy under the maximum entropy objective. While SAC's use of function approximation and interleaved updates means these guarantees don't directly transfer, the SPI framework provides strong evidence that the algorithm is well-founded. It explains *why* the specific loss functions used in SAC are the correct ones and gives us confidence that the algorithm is making principled steps toward an optimal solution, which helps explain its empirical success and stability.

16. **Derive the policy gradient update using the reward-to-go formulation and explain why it remains an unbiased estimator of the true gradient.**

**Answer**

The standard Policy Gradient Theorem gives the gradient as:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim p_\theta(\tau)}\left[\left(\sum_{t=0}^{T-1}\nabla_\theta \log \pi_\theta(a_t|s_t)\right)\left(\sum_{t'=0}^{T-1}\gamma^{t'} r_{t'}\right)\right]$$

We can rewrite this by distributing the outer sum over the inner log-probability terms:

$$\nabla_\theta J(\theta) = \mathbb{E}_\tau\left[\sum_{t=0}^{T-1}\nabla_\theta \log \pi_\theta(a_t|s_t)\left(\sum_{t'=0}^{T-1}\gamma^{t'} r_{t'}\right)\right]$$

The reward-to-go formulation modifies this to:

$$\nabla_\theta J(\theta)_{\text{RTG}} = \mathbb{E}_\tau\left[\sum_{t=0}^{T-1}\nabla_\theta \log \pi_\theta(a_t|s_t)\left(\sum_{t'=t}^{T-1}\gamma^{t'} r_{t'}\right)\right] = \mathbb{E}_\tau\left[\sum_{t=0}^{T-1}\nabla_\theta \log \pi_\theta(a_t|s_t)G_t\right]$$

To prove this is an unbiased estimator, we must show that the expected value of the terms we removed is zero. The difference between the original gradient and the reward-to-go gradient is:

$$\text{Difference} = \mathbb{E}_\tau\left[\sum_{t=0}^{T-1}\nabla_\theta \log \pi_\theta(a_t|s_t)\left(\sum_{t'=0}^{t-1}\gamma^{t'} r_{t'}\right)\right]$$

This is the gradient contribution from rewards that occurred *before* action $a_t$ was taken. Let's look at the expectation of a single term for a specific timestep $t$ and a past timestep $t' < t$:

$$\mathbb{E}_\tau[\nabla_\theta \log \pi_\theta(a_t|s_t)\cdot r_{t'}]$$

The expectation is over all trajectories. We can condition this on the state and action history up to time $t$. Due to the Markov property, the action $a_t$ only depends on the state $s_t$. The reward $r_{t'}$ depends on the state-action pair $(s_{t'}, a_{t'})$ which occurred in the past. Given the state $s_t$, the action $a_t$ is conditionally independent of past rewards.

More formally, the expectation of the gradient of the log policy with respect to future actions is zero. The causality principle states that an action at time $t$ cannot influence rewards at time $t' < t$. Therefore, the gradient of the policy with respect to $\theta$ at time $t$ should not be correlated with past rewards. The expected value of this cross-term is zero.

Since we only subtracted terms whose expectation is zero, the resulting estimator remains unbiased. However, we have removed a significant source of noise (the sum of past rewards), which reduces the variance of the gradient estimate.

17. **Explain the telescoping sum argument used in the proof of the Performance Difference Lemma.**

**Answer**

The telescoping sum argument is a key step in the proof of the Performance Difference Lemma. It's used to relate the value function at the start of an episode, $V(s_0)$, to the advantage function summed over the entire trajectory. The proof starts with the expression for the performance difference: $J(\theta') - J(\theta)$. The core idea is to express the value of the old policy, $J(\theta) = \mathbb{E}[V^{\pi_\theta}(s_0)]$, in a way that introduces the advantage function.
The Bellman equation for the value function is:

$$V^\pi(s_t) = \mathbb{E}_{a_t \sim \pi, s_{t+1} \sim p}[r(s_t, a_t) + \gamma V^\pi(s_{t+1})]$$

We can rearrange this to express the advantage function. The advantage is $A^\pi(s_t, a_t) = r(s_t, a_t) + \gamma V^\pi(s_{t+1}) - V^\pi(s_t)$.
The telescoping sum trick comes from expressing $V^\pi(s_t)$ in terms of the advantage function. For any state $s_t$, we have:

$$V^\pi(s_t) = \mathbb{E}_{\tau_{t:\infty}}[A^\pi(s_t, a_t) + V^\pi(s_t) - \gamma V^\pi(s_{t+1})]$$

This seems circular, but when expanded over a trajectory starting from $s_0$, it creates a telescoping sum. The value at a state is the expected immediate advantage plus the discounted value of the next state.

$$V^\pi(s_0) = \mathbb{E}[A^\pi(s_0, a_0) + \gamma V^\pi(s_1)]$$

$$V^\pi(s_0) = \mathbb{E}[A^\pi(s_0, a_0) + \gamma(A^\pi(s_1, a_1) + \gamma V^\pi(s_2))]$$

$$V^\pi(s_0) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t A^\pi(s_t, a_t)\right]$$

This identity shows that the value of a state is the expected sum of all future advantages. In the proof of the Performance Difference Lemma, this identity is used to replace the value function term with a sum of advantages, which ultimately leads to the final form of the lemma:

$$J(\theta') - J(\theta) = \mathbb{E}_{\tau \sim p_{\theta'}(\tau)}\left[\sum_{t=0}^{\infty} \gamma^t A^{\pi_\theta}(s_t, a_t)\right]$$

The "telescoping" nature comes from how terms like $\gamma V^\pi(s_1)$ in one expansion are replaced by the next step's advantage and value, creating a chain that sums to infinity.

18. **Why is TRPO's constraint on KL-divergence a more principled approach than simply using a small, fixed learning rate?**

**Answer**

Using a small, fixed learning rate in a standard policy gradient algorithm is a heuristic approach to prevent destructive policy updates. While it often works in practice, it lacks theoretical grounding and can be inefficient. TRPO's constraint on KL-divergence is a more principled approach for several reasons:

(a) **Theoretical Guarantee:** The TRPO update is derived from a lower bound on policy performance: $J(\theta') \geq L_{\pi_\theta}(\pi_{\theta'}) - C \cdot D_{\text{KL}}^{\max}(\pi_\theta, \pi_{\theta'})$. By maximizing this lower bound, TRPO guarantees (under its assumptions) that the true objective function $J(\theta)$ will not decrease. A simple gradient step with a fixed learning rate offers no such guarantee; a step that seems good locally might be disastrous globally.

(b) **Adaptive Step Size:** A fixed learning rate means the step size in parameter space ($\|\Delta\theta\|$) is constant. However, the same change in parameter space can lead to vastly different changes in policy behavior. A small parameter change might cause a huge policy shift in one region of the state space, while a large parameter change might have little effect in another. KL-divergence measures the difference in the *output distributions* of the policies. By constraining the KL-divergence, TRPO takes steps of a consistent "size" in the space of policies, not parameters. This allows it to take large, safe steps in parameter space when the policy is not sensitive to changes, and small, cautious steps when it is, making it more adaptive and efficient.

(c) **Curvature Information:** The TRPO update involves a second-order optimization method (conjugate gradient) to solve the constrained problem. This implicitly uses information about the curvature of the loss landscape (via the Fisher Information Matrix, which approximates the Hessian of the KL-divergence). This allows it to find a much better update direction than a simple first-order gradient step, which only follows the steepest local direction.

In summary, constraining the KL-divergence is a more robust and adaptive way to control policy updates, grounded in a theoretical guarantee of monotonic improvement, whereas a fixed learning rate is a simpler but less reliable heuristic.

19. **Discuss the multifaceted intuition behind the maximum entropy objective. How does it encourage exploration, robustness, and stability?**

**Answer**

The maximum entropy objective, $J(\pi) = \mathbb{E}[r + \alpha\mathcal{H}]$, adds a bonus for policy randomness. The intuition is multifaceted:

(a) **Enhanced Exploration:** The most direct benefit is improved exploration. A standard RL agent, upon finding a reasonably good strategy, might quickly become deterministic and stop exploring, potentially getting stuck in a local optimum. By explicitly rewarding entropy, the agent is incentivized to try a wider variety of actions, especially in the early stages of training. It discourages the policy from collapsing prematurely to a single strategy, increasing the chances that it will discover globally optimal solutions.

(b) **Robustness and Transferability:** A policy trained with an entropy bonus is less "committed" to a single way of solving a problem. If there are multiple actions or sequences of actions that lead to similarly high rewards, a maximum entropy policy will learn to assign significant probability to all of them. This makes the policy more robust. If the environment changes slightly or the agent encounters a novel situation, a stochastic policy has more behaviors in its repertoire to fall back on, potentially adapting more quickly. This can also aid in transferring a learned policy to a new, similar task.

(c) **Improved Stability:** The entropy term acts as a powerful regularizer on the policy. In standard policy gradients, updates can sometimes be very sharp, drastically changing the policy. The entropy bonus smooths the optimization landscape. It penalizes policies that become too "peaked" or deterministic too quickly, encouraging smoother, more gradual changes to the policy distribution. This regularization can lead to more stable training dynamics and prevent the policy from oscillating or diverging.

In essence, maximum entropy RL reframes the goal from "find the single best way to solve the task" to "find all the good ways to solve the task and be prepared to use any of them," which naturally leads to better exploration and more robust final policies.

20. **How does the final convergence proof of Soft Policy Iteration (Theorem 1) bring together the results from Lemma 1 and Lemma 2?**

**Answer**

The proof of Theorem 1 (Convergence of Soft Policy Iteration) is a powerful conclusion that rests directly on the guarantees provided by the two preceding lemmas. It combines them to show that the overall process is not just stable, but converges to the optimal policy.
Here's how the pieces fit together:

(a) **Foundation from Lemma 1 (Soft Policy Evaluation):** Lemma 1 guarantees that for any given policy $\pi_i$, the evaluation step will converge to its true soft Q-function, $Q^{\pi_i}$. This ensures that the "evaluation" part of the iteration is well-defined and provides a solid target for the improvement step.

(b) **Guarantee from Lemma 2 (Soft Policy Improvement):** Lemma 2 guarantees that when we use $Q^{\pi_i}$ to find a new policy $\pi_{i+1}$, the new policy's Q-function will be pointwise greater than or equal to the old one: $Q^{\pi_{i+1}} \geq Q^{\pi_i}$. This is the engine of progress, ensuring that each full iteration of SPI results in a better (or at least, no worse) policy.

(c) **Combining for Monotonicity:** By chaining these two results, the proof of Theorem 1 establishes a monotonically non-decreasing sequence of soft Q-functions:
$$Q^{\pi_0} \leq Q^{\pi_1} \leq Q^{\pi_2} \leq \dots$$
Each step in this sequence is a valid policy improvement because of Lemma 2, and the Q-values themselves are correctly computed because of Lemma 1.

(d) **Boundedness and Convergence:** The proof then argues that this sequence must be bounded from above (since rewards and entropy are bounded). A monotonically non-decreasing sequence that is bounded from above is guaranteed by the Monotone Convergence Theorem to converge to a limit, which we call $Q^{\pi^*}$.

(e) **Optimality of the Limit:** Finally, the proof shows that this limit policy $\pi^*$ must be optimal. At convergence, the policy improvement step can no longer find a better policy. This means $\pi^*$ is a fixed point of the improvement operator, which implies it is the policy that minimizes the KL-divergence to its own exponentiated Q-function. As shown in the proof of Lemma 2, any other policy would result in a strictly lower soft Q-value, proving that $\pi^*$ is the unique optimal policy under the maximum entropy objective.

Thus, Theorem 1 elegantly weaves together the guarantees of evaluation and improvement to construct a proof of monotonic convergence to optimality.

21. **What is the difference between the score function and the advantage function in the context of policy gradients?**

**Answer**

The score function and the advantage function play distinct but complementary roles in the policy gradient update.

**Score Function:** $\nabla_\theta \log \pi_\theta(a_t|s_t)$

- **Role:** It determines the *direction* of the policy update.
- **Interpretation:** The score function points in the direction in parameter space ($\theta$) that will most increase the probability of taking action $a_t$ in state $s_t$. It tells us "how to change the policy parameters to make this specific action more likely."
- **Property:** It does not depend on the rewards or the outcome of the action. It is purely a property of the policy and its parameterization.

**Advantage Function:** $A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t)$

- **Role:** It determines the *magnitude and sign* of the policy update.
- **Interpretation:** The advantage function is a scalar value that tells us *how good* the action $a_t$ was compared to the average action in state $s_t$.
- **Property:** It is a measure of the outcome of the action. A positive advantage means the action was better than average and should be reinforced. A negative advantage means it was worse than average and should be discouraged.

**In the Update Rule:** The policy gradient update combines them:

$$\Delta\theta \propto A^\pi(s_t, a_t) \cdot \nabla_\theta \log \pi_\theta(a_t|s_t)$$

The score function provides the direction for the update (which knobs to turn in the policy network), and the advantage function provides the signal for how much and in which way (increase or decrease probability) to turn them.

22. **If a baseline must have an expected gradient contribution of zero to be unbiased, why can't we use an action-dependent baseline like $b(s_t, a_t)$?**

**Answer**

The proof that a baseline does not introduce bias relies on the fact that its expectation can be separated from the policy gradient term. For a state-dependent baseline $b(s_t)$, the expectation of the subtracted term is:

$$\mathbb{E}\left[\nabla_\theta \log \pi_\theta(a_t|s_t)b(s_t)\right] = \mathbb{E}_{s_t}\left[b(s_t)\mathbb{E}_{a_t\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(a_t|s_t)]\right]$$

The inner expectation $\mathbb{E}_{a_t}[\nabla_\theta \log \pi_\theta(a_t|s_t)]$ evaluates to $\nabla_\theta \int \pi_\theta(a_t|s_t)da_t = \nabla_\theta(1) = 0$. Because the inner term is zero, the entire expression is zero, and the estimator remains unbiased.
Now, consider an action-dependent baseline $b(s_t, a_t)$. The expectation of the subtracted term would be:

$$\mathbb{E}\left[\nabla_\theta \log \pi_\theta(a_t|s_t)b(s_t,a_t)\right] = \mathbb{E}_{s_t}\left[\mathbb{E}_{a_t\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(a_t|s_t)b(s_t,a_t)]\right]$$

In this case, we cannot pull $b(s_t, a_t)$ out of the inner expectation because it depends on $a_t$, the variable of integration. The inner expectation is now:

$$\int \pi_\theta(a_t|s_t)\left(\frac{\nabla_\theta \pi_\theta(a_t|s_t)}{\pi_\theta(a_t|s_t)}\right)b(s_t,a_t)da_t = \int (\nabla_\theta \pi_\theta(a_t|s_t))b(s_t,a_t)da_t$$

This integral is generally **not equal to zero**. By introducing a term that depends on the action $a_t$, we have created a correlation that breaks the proof of unbiasedness. Subtracting an action-dependent baseline would fundamentally change the objective function being optimized and would introduce a bias into the gradient estimate, making it incorrect.

23. **What is the "reparameterization trick" and why is it important for training the actor in SAC?**

**Answer**

The reparameterization trick is a method for computing low-variance gradients of a stochastic node in a computation graph. In SAC, the actor network $\pi_\phi$ outputs a distribution (e.g., a Gaussian with mean $\mu_\phi(s)$ and standard deviation $\sigma_\phi(s)$). To get an action, we would normally sample from this distribution: $a \sim \mathcal{N}(\mu_\phi(s), \sigma_\phi(s))$. This sampling operation is stochastic and non-differentiable, which means we cannot backpropagate gradients through it to update the actor's parameters $\phi$.

The reparameterization trick bypasses this by reframing the sampling process. Instead of sampling the action directly, we sample a random noise variable from a fixed, simple distribution (e.g., $\epsilon \sim \mathcal{N}(0, I)$) and then transform this noise into an action using a deterministic function that depends on the policy parameters. For a Gaussian policy, this would be:

$$a_t = \mu_\phi(s_t) + \sigma_\phi(s_t) \odot \epsilon$$

Here, the stochasticity is injected by $\epsilon$, which is external to the computation graph involving $\phi$. The path from the parameters $\phi$ to the final action $a_t$ is now fully deterministic.

This is crucial for SAC's policy loss:

$$J_\pi(\phi) = \mathbb{E}_{s_t,\epsilon}\left[\alpha \log(\pi_\phi(a_t(\epsilon, s_t)|s_t)) - Q_\theta(s_t, a_t(\epsilon, s_t))\right]$$

Because $a_t$ is now a deterministic function of $\phi$, we can easily backpropagate the gradient from the Q-function (critic) and the log-probability term all the way back to the actor's parameters $\phi$. This provides a direct, low-variance path for the critic to "tell" the actor how to adjust its output to produce higher-value actions.

24. **Explain the loss function for the value network, $J_V(\psi)$, in SAC. What is it trying to achieve?**

**Answer**

The loss function for the value network $V_\psi(s)$ in SAC is a mean-squared error loss designed to make the network's output match the definition of the soft state-value function. The loss is:

$$J_V(\psi) = \mathbb{E}_{s_t \sim \mathcal{D}} \left[ \frac{1}{2} \left( V_\psi(s_t) - \text{Target} \right)^2 \right]$$

where the target is:

$$\text{Target} = \mathbb{E}_{a_t \sim \pi_\phi} \left[ \min_{i=1,2} Q_{\theta_i}(s_t, a_t) - \alpha \log \pi_\phi(a_t|s_t) \right]$$

This loss function is trying to achieve the following:

(a) **Match the Soft Value Definition:** The theoretical definition of the soft state-value function is $V^{\text{soft}}(s_t) = \mathbb{E}_{a_t \sim \pi}[Q^{\text{soft}}(s_t, a_t) - \alpha \log \pi(a_t|s_t)]$. The loss function directly implements this. It trains the network $V_\psi(s_t)$ to predict the expected value of the soft Q-function (estimated by the critics) minus the policy's entropy bonus, where the expectation is over actions from the current policy.

(b) **Provide a Target for the Actor:** The primary purpose of learning this value function is to use it in the policy (actor) loss. The actor is trained to produce actions that maximize the soft Q-value. By having a separate value function, the actor's update can be more stable.

(c) **Consistency with Clipped Q-values:** The target uses the minimum of the two Q-networks ($\min Q_{\theta_i}$). This is crucial for consistency. Since the policy is also trained using this minimum Q-value to prevent overestimation, the value function must also be trained on this same pessimistic value estimate to ensure all components of the algorithm are aligned.

In essence, the value network is being trained to be an accurate estimator of the expected future entropy-augmented return from a given state, according to the current policy and the pessimistic critics.

25. **Why can the theoretical guarantees for Soft Policy Iteration not be directly applied to the practical SAC algorithm? What are the main sources of approximation?**

**Answer**

The theoretical guarantees for Soft Policy Iteration (SPI) prove convergence in an idealized, tabular setting. These guarantees cannot be directly applied to the practical SAC algorithm due to several major sources of approximation and deviation from the theoretical model:

(a) **Function Approximation:** The most significant difference is that SAC uses neural networks to approximate the policy, Q-functions, and value function, whereas the proofs assume a tabular representation (i.e., a lookup table for every state-action value). Function approximation introduces errors; the networks may not be able to perfectly represent the true value functions. These approximation errors can break the convergence guarantees.

(b) **Incomplete Evaluation and Improvement:** The SPI framework assumes that the policy evaluation and policy improvement steps are run to convergence at each iteration.

- **Evaluation:** SAC does not iterate the Bellman backup until the Q-function converges. Instead, it takes just one (or a few) gradient steps on a mini-batch of data to update the critics.
- **Improvement:** Similarly, SAC does not find the exact new policy that minimizes the KL-divergence. It takes one (or a few) gradient steps to update the actor.

This interleaving of partial updates is more computationally feasible but deviates from the strict alternation required by the proof.

(c) **Off-Policy Sampling:** The proofs are often simplest in an on-policy context, while SAC is fundamentally off-policy, learning from a replay buffer of past experiences. While this is a strength, it introduces potential issues with stale data and distribution shift that the idealized proofs do not fully account for, although techniques like target networks are used to mitigate this.

(d) **Stochasticity of Updates:** The practical algorithm uses stochastic gradient descent on mini-batches sampled from the replay buffer. This introduces noise into the updates, which is another deviation from the deterministic updates in the theoretical SPI framework.

Despite these gaps, the theoretical framework is invaluable. It provides confidence that the loss functions and update rules used by SAC are principled and are driving the policy towards the correct objective, which helps explain why SAC is so empirically successful and stable even with these approximations.