Week 7: Policy Gradient Methods

Your Name

Your Institution

July 19, 2025

Introduction to Policy Gradient Methods

What are Policy Gradient Methods?

Policy Gradient Methods optimize the policy directly, improving agent behavior in an environment. Unlike value-based methods that estimate value functions, they directly update the policy.

- **Direct Optimization**: Policy parameters are updated via gradient ascent on expected rewards.
- **Stochastic Policies**: Can handle both deterministic and stochastic policies, enabling flexible action selection.

Significance in Reinforcement Learning

1. **Handling High-Dimensional Action Spaces**: Effective in environments with large or continuous action spaces.
2. **Better Exploration**: Directly parameterizes the policy, enhancing exploration and balancing it with exploitation.
3. **Applicable to Complex Tasks**: Suited for problems in robotics, game playing, and natural language processing.

Example: REINFORCE Algorithm

The REINFORCE algorithm is a notable policy gradient method that updates policy parameters using:

$$\theta \leftarrow \theta + \alpha \cdot G_t \nabla_\theta \log \pi_\theta(a_t|s_t) \tag{1}$$

Where:

- θ = policy parameters
- α = learning rate
- G_t = return after time t
- $\pi_\theta(a_t|s_t)$ = policy probability of taking action a_t in state s_t

Key Points to Emphasize

- Direct interaction with the policy for potentially faster learning.
- Useful in complex, multi-dimensional environments.
- Balancing exploration and exploitation is crucial for long-term rewards.
- Foundation for advanced algorithms like Actor-Critic and PPO.

Understanding Policies

What are Policies in Reinforcement Learning?

In Reinforcement Learning (RL), a **policy** defines the behavior of an agent interacting with an environment. It is essentially a strategy employed by the agent to determine its actions based on the current state it observes.

Types of Policies

Policies can be categorized into two main types: **Deterministic** and **Stochastic** policies.

- **Deterministic Policies**
  - A deterministic policy outputs a specific action for a given state:
    $$a = \pi(s)$$
  - Example: In a maze, if there is only one exit, the agent will always choose that exit.
  - **Key Points:**
    - Predictable and straightforward
    - Easier to implement in simple environments
- **Stochastic Policies**
  - A stochastic policy provides a probability distribution over actions for each state:
    $$P(a|s) = \pi(a|s)$$
  - Example: The agent might have a 70% chance to go straight (exit) and a 30% chance to turn back.
  - **Key Points:**
    - Incorporates variability in agent behavior

Summary and Applications

- **Policy**: Defined strategy used by an agent to make decisions.
- **Deterministic Policy**: Predictable output for each state (e.g., always moving right in a maze).
- **Stochastic Policy**: Probabilistic action choices for each state (e.g., a choice influenced by probabilities).

Applications

Policies are crucial in various RL algorithms, impacting exploration-exploitation balance and performance in learning tasks. Understanding policies is essential for effectively applying Policy Gradient Methods to maximize expected rewards.

Visualizing Policies

Consider deterministic policies as a fixed path through a maze, while stochastic policies can be visualized as a cloud of potential paths demonstrating exploration.

The Objective of Policy Gradient Methods - Introduction

Overview

In reinforcement learning, the ultimate goal is to identify a policy that maximizes the expected cumulative reward over time. Policy gradient methods directly parameterize policies, allowing them to optimize the expected rewards without relying on value functions.

The Objective of Policy Gradient Methods - Key Concepts

- **Policy Parameterization**:
  - Policies are expressed as functions of states and parameters (θ), denoted as $\pi(a|s;\theta)$, where:
    - s: state
    - a: action
    - θ: parameters of the policy
  - Adjusting parameters allows agents to learn complex behaviors compared to value-based methods.
- **Expected Reward Maximization**:
  - Maximize the expected return $J(\theta)$, defined as:

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} [R(\tau)] \tag{2}$$

    where $R(\tau)$ is the total return of trajectory τ.
  - Find optimal parameters θ^* that yield the highest expected return.

The Objective of Policy Gradient Methods - Advantages and Theorem

- **Core Advantages**:
  - **Direct Optimization**:
    - Unlike value-based methods, policy gradients optimize the policy directly, suitable for high-dimensional action spaces.
  - **Stochastic Policies**:
    - Naturally support stochastic policies, facilitating exploration during training to avoid local minima.
- **Policy Gradient Theorem**:

$$\nabla J(\theta) = \mathbb{E}_{s_t \sim \rho^\pi} \left[\nabla \log \pi(a_t | s_t; \theta) Q^\pi(s_t, a_t) \right] \tag{3}$$

where $Q^\pi(s, a)$ represents the action-value function under the policy π.

Key Differences from Value-Based Methods - Overview

Policy Gradient Methods and Value-Based Methods (such as Q-learning and SARSA) represent two distinct approaches to reinforcement learning (RL). Understanding their differences is crucial for choosing the appropriate method for a given task.

Key Differences from Value-Based Methods - Definitions

- **Policy Gradient Methods:**
  - Directly parameterize the policy and optimize the expected reward via gradient ascent.
  - Focus on learning a strategy that defines the probability of taking each action in a given state.
- **Value-Based Methods:**
  - Focus on estimating values of state-action pairs (Q-values) or states (V-values) to derive the best action indirectly.
  - Divide the problem into two components: estimating the value function and deriving the policy from those estimates.

Key Differences from Value-Based Methods - Learning Approach

- **Policy Gradient:**
  - Adjusts policy parameters directly using gradient updates.
  - Example update:
    $$\theta \leftarrow \theta + \alpha \nabla J(\theta) \tag{4}$$
    where $J(\theta)$ is the expected return.
- **Value-Based:**
  - Updates action-value estimates using Bellman equations.
  - Example update for Q-learning:
    $$Q(s, a) \leftarrow Q(s, a) + \alpha \left(r + \gamma \max_{a'} Q(s', a') - Q(s, a) \right) \tag{5}$$

Key Differences from Value-Based Methods - Strengths and Weaknesses

Policy Gradient Strengths

- Can handle high-dimensional action spaces (e.g., continuous actions).
- Converges to optimal policies in stochastic environments.
- More robust in noisy environments.

Policy Gradient Weaknesses

- Higher variance in updates (requires more samples).
- Less stable and may take longer to converge.

Value-Based Strengths

- More sample efficient as it leverages the value function.
- Easier to implement in discrete action spaces.

Value-Based Weaknesses

Key Differences from Value-Based Methods - When to Use Each Method

Use Policy Gradient Methods when:

- The action space is continuous.
- You need to learn stochastic policies.
- Dealing with complex systems where direct Q-value estimation is infeasible.

Use Value-Based Methods when:

- The action space is discrete and manageable.
- Aiming for faster training with lower variance in returns.

Key Differences from Value-Based Methods - Conclusion

- Policy Gradient and Value-Based methods each have unique strengths suited to different problem scenarios in reinforcement learning.
- A clear understanding of these differences can aid in selecting the right approach for specific applications.

Key Points to Remember

- Policy Gradient = Directly optimize policy
- Value-Based = Estimate values then derive policy
- Choice of method depends on the task's properties and action space.

Mathematical Foundation - Part 1

Introduction to Policy Gradient Methods

Policy Gradient Methods are a class of reinforcement learning algorithms that optimize the policy directly. The focus is on:

- Maximizing expected returns.
- Tweaking the policy parameters.

Mathematical Foundation - Part 2

Policy Gradient Theorem
The **Policy Gradient Theorem** provides a fundamental basis for estimating gradients of the expected return with respect to policy parameters.
Definition: Given a policy $\pi(a|s;\theta)$:

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

where $R(\tau)$ is the return from trajectory τ.
Gradient Calculation:

$$\nabla J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}\left[\nabla_\theta \log \pi(a|s;\theta) R(\tau)\right]$$

Mathematical Foundation - Part 3

Key Points to Emphasize

1. Direct Optimization: Policy gradients focus on direct optimization of the policy.
2. Exploration: They support exploration through probabilistic actions.
3. Variance Challenge: High variance in gradient estimates is a key training challenge.

Example: Consider a simple policy modeled as a Gaussian distribution for continuous action space:

$$\pi(a|s;\theta) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(a-\mu(s;\theta))^2}{2\sigma^2}}$$

Conclusion: The policy gradient theorem lays the groundwork for maximizing expected returns, enabling a transition to more complex architectures like Actor-Critic Methods.

Code Snippet Example

```
import numpy as np

def compute_policy_gradient(log_probs, rewards):
    # Standardize rewards
    rewards = (rewards - np.mean(rewards)) / (np.std(
        rewards) + 1e-10)
    return np.dot(log_probs.T, rewards)
```

This Python snippet illustrates a basic approach to computing the policy gradient using log probabilities and standardized rewards.

Actor-Critic Methods - Overview

Overview of Actor-Critic Architectures

Actor-Critic methods incorporate both a policy function (actor) and a value function (critic) to optimize decision-making in reinforcement learning (RL). This combination provides a more stable learning process compared to standard policy gradient approaches.

Actor-Critic Methods - Key Components

1. **Actor**:
    - Responsible for selecting actions based on the current policy.
    - Maps states to actions ($\pi(a|s)$) to maximize cumulative rewards.
    - **Example**: In a game of chess, the actor suggests moves based on the board.
2. **Critic**:
    - Evaluates actions taken by the actor by estimating the value function ($V(s)$ or $Q(s,a)$).
    - Provides feedback to refine the policy.
    - **Example**: The critic assesses chess moves proposed by the actor, estimating their effectiveness.

Actor-Critic Methods - How it Works

- The actor generates actions using its current policy, receiving feedback from the critic.
- The critic's value estimations influence the actor's policy updates.
- **Loss Functions**:

$$\text{Actor loss} = -\log(\pi(a|s)) \cdot A(s, a) \tag{6}$$

$$\text{Critic loss} = \frac{1}{2}(R_t - V(s_t))^2 \tag{7}$$

where $A(s, a)$ is the advantage function and R_t is the discounted return after taking action a in state s.

Advantages and Disadvantages of Policy Gradient Methods

Overview of Policy Gradient Methods

Policy gradient methods are a class of reinforcement learning algorithms that optimize a policy directly. Unlike value-based methods, they refine the policy through gradients and enhance exploration.

Advantages of Policy Gradient Methods

1. **Direct Policy Optimization**
    - Optimize the policy function directly, allowing fine-tuning without value function approximation.
    - *Example*: REINFORCE calculates the gradient of expected rewards to adjust policy parameters.
2. **Stochastic Policies**
    - Can represent stochastic policies essential for uncertain environments with multiple optimal actions.
    - *Example*: In games with multiple strategies yielding the same reward, stochastic policies maintain exploration.
3. **Better Handling of High-Dimensional Action Spaces**
    - Particularly beneficial in continuous action spaces, allowing nuanced action adjustments.
    - *Example*: Robots requiring fine-grained control, such as robotic arms in manufacturing.
4. **Asymptotic Convergence Guarantees**
    - Under certain conditions, guaranteed to converge to a local optimum.
    - This property can ensure stability in training.

Disadvantages of Policy Gradient Methods

1. **High Variance in Gradient Estimates**
   - Suffer from high variance leading to slow learning and instability.
   - *Illustration*: Gradient estimates may fluctuate due to randomness in sampling interactions.
2. **Sample Inefficiency**
   - Require a large number of samples to converge, leading to increased computational costs.
   - *Example*: Training through thousands of episodes may be necessary in complex environments.
3. **Local Optima Issues**
   - Optimization may converge to local optima rather than global optima, affected by initial parameters.
   - This limits performance, especially in complex landscapes.
4. **Tuning Hyperparameters**
   - Hyperparameter tuning can be complex and dramatically impact performance.
   - *Example*: The choice of learning rate can affect convergence speed and stability.

Conclusion and Key Formula

Policy gradient methods provide robust frameworks for complex reinforcement learning tasks, especially in direct optimization and handling stochasticity. However, considering issues of high variance, sample inefficiency, and tuning challenges is essential for achieving optimal results.

Key Formula: Policy Gradient Theorem

The policy gradient can be derived using:

$$\nabla J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\nabla \log \pi_\theta(a|s) \cdot R \right] \tag{8}$$

Where $J(\theta)$ is the performance measure, τ is the trajectory, and R represents the sum of rewards.

Common Algorithms

Introduction to Policy Gradient Algorithms

Policy gradient methods are a class of reinforcement learning algorithms that optimize the policy directly. These methods enable agents to learn optimal behaviors by updating their strategies based on the cumulative rewards received.

Below we introduce three widely-used policy gradient algorithms:

- REINFORCE
- Proximal Policy Optimization (PPO)
- Actor-Critic methods

1. REINFORCE Algorithm

Description

The REINFORCE algorithm is a Monte Carlo policy gradient method that updates policy parameters based on the total rewards received over an episode.

- **Key Features:**
  - Stochastic Policy: Governs the agent's action selection through probabilities.
  - Update Rule: Uses the complete reward signal to update the policy after each episode.
- **Update Formula:**

$$\theta_{t+1} = \theta_t + \alpha \cdot G_t \cdot \nabla \log(\pi_\theta(a_t|s_t)) \tag{9}$$

  where:
  - θ = Policy parameters
  - G_t = Total discounted reward from time t
  - α = Learning rate
- **Example:** In a game, if an agent receives a high score after following

2. Proximal Policy Optimization (PPO)

Description

PPO is a popular and robust algorithm that strikes a balance between exploration and exploitation while maintaining stable learning.

- **Key Features:**
  - Clipped Surrogate Objective: Limits policy updates to prevent drastic changes.
  - Efficiency: Works well with large neural networks and is sample efficient.
- **Update Rule:** Maximize the clipped objective:

$$L^{CLIP}(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right] \tag{10}$$

  where:
  - $r_t(\theta)$ = Probability ratio
  - $\hat{A}_t$ = Advantage function estimate
  - ϵ = Clip range
- **Example:** PPO can be used in training agents for environments with continuous action spaces, such as robotic control, ensuring stable and

3. Actor-Critic Methods

Description

These methods leverage two components: the actor (policy function) and the critic (value function). The actor determines the actions, while the critic evaluates them, aiming to improve learning efficiency.

- **Key Features:**
  - Reduced Variance: Value function estimation helps reduce variance in the policy updates.
  - Flexibility: Can incorporate various types of policies and value function approximations.
- **Update Rules:**
  - Actor Update: Similar to REINFORCE, using the critic's feedback.
  - Critic Update: Typically through Temporal Difference learning (e.g., TD(0)):

$$V(s_t) \leftarrow V(s_t) + \alpha \left(r_t + \gamma V(s_{t+1}) - V(s_t)\right) \tag{11}$$

- **Example:** In a navigation task, an agent uses a critic to assess the quality of its actions within an environment, resulting in a more refined policy update process

Key Points to Remember

- **Direct Policy Optimization:** Policy gradient methods focus on directly adjusting policies based on rewards.
- **Exploration vs. Exploitation:** Balancing new actions with known profitable actions to stabilize training.
- **Algorithm Choice:** The selection of an algorithm (REINFORCE, PPO, Actor-Critic) depends on specific problem requirements such as environment complexity and resource availability.

Conclusion

Understanding these common policy gradient algorithms provides a foundation for developing advanced reinforcement learning applications, setting up for the next exploration of their real-world applications in diverse fields.

Applications of Policy Gradient Methods - Introduction

Introduction to Policy Gradient Methods

Policy Gradient Methods are a class of reinforcement learning algorithms that optimize the policy directly, enhancing decision-making in complex environments.

- Allow for nuanced control in high-dimensional settings.
- Contrast with value-based methods by focusing on policy optimization.

Applications of Policy Gradient Methods - Real-World Applications

Real-World Applications

1. Robotics
    - **Robotic Arm Manipulation:** Enables robots to learn complex tasks, adapting real-time movements based on feedback. Rapid adaptation enhances versatility.
    - **Humanoid Locomotion:** Training allows humanoid robots to develop stable gaits across varied terrains by stepping through actions in training episodes.
2. Game Playing
    - **Video Games (Dota 2 AI):** AI learns strategies dynamically against opponents, adapting in real-time to improve performance.
    - **Chess and Board Games:** Develops superior strategies by evaluating numerous game states and responding optimally.

Applications of Policy Gradient Methods - Summary and Conclusion

Summary

- Policy gradient methods excel in environments requiring continuous optimization.
- Applications demonstrate their versatility in addressing complex real-world problems.

Formula

To illustrate the update mechanism in policy gradient methods:

$$\theta' = \theta + \alpha \nabla J(\theta) \tag{12}$$

Where:

- θ = parameters of the policy
- α = learning rate
- $J(\theta)$ = expected reward function based on the policy

Future Directions and Research Trends - Overview

Overview of Policy Gradient Methods

Policy gradient methods are a cornerstone of reinforcement learning, aiming to optimize the policy directly using gradient ascent. This field is rapidly evolving with several exciting research themes enhancing their capabilities and applicability.

Current Research Trends - Part 1

1. **Sample Efficiency Improvements**
    - Off-Policy Training: Utilizing experience replay and off-policy strategies.
    - Meta-Learning: Adapting from fewer examples.
2. **Exploration Strategies**
    - Variational Exploration: Adaptive approaches based on model uncertainty.
    - Intrinsic Motivation: Rewarding novelty in exploration.

Current Research Trends - Part 2

3. **Integration with Deep Learning**
   - Actor-Critic Methods: Utilizing both policy (actor) and value (critic) networks.

$$\theta \leftarrow \theta + \alpha \nabla J(\theta) \tag{13}$$

4. **Multi-Agent Systems**
   - Cooperative learning and adversarial training.
5. **Generalization and Transfer Learning**
   - Domain Randomization and Hierarchical Policy Learning.

Future Directions

1. Robustness to Model Uncertainty: Developing policies for uncertain or adversarial conditions.
2. Real-Time Implementation: Optimizing for environments requiring swift decision-making, like autonomous driving.
3. Neuroscience-Inspired Methods: Exploring biological learning mechanisms for new policy techniques.

Key Points to Emphasize

- The evolving landscape of policy gradient methods is shaped by demands for efficiency, adaptability, and robustness.
- Innovations in exploration, deep learning integration, and multi-agent systems pave the way for more sophisticated applications.
- Future research is likely to enhance real-world applications in AI-driven fields.