Week 7: Actor-Critic Methods

Your Name

Your Institution

June 30, 2025

Week 7: Actor-Critic Methods

Your Name

Your Institution

June 30, 2025

What are Actor-Critic Methods?

Actor-Critic methods combine the benefits of two approaches in reinforcement learning:

- **Actor:** Selects actions based on the current policy.
- **Critic:** Evaluates actions by estimating the value function.

Significance of Actor-Critic Methods

1. **Combining Strengths:**
    - *Policy-based methods:* Learn directly from the policy, requiring more data.
    - *Value-based methods:* Focus on estimating value functions, struggling with continuous actions.
2. **Sample Efficiency:** Less sample requirement to learn effective policies compared to pure reinforcement methods.
3. **Stability:** The use of value estimates reduces variance in policy updates.

Applications of Actor-Critic Methods

- **Game Playing:** Used in environments like video games (e.g., AlphaGo).
- **Robotics:** Trained for complex tasks such as balancing in real-time.
- **Autonomous Vehicles:** Implemented in decision-making systems for effective navigation.

Illustrative Example: A Simple Grid World

Consider a grid world where an agent navigates to a goal while avoiding obstacles:

- **Actor:** Proposes movements (up, down, left, right) based on the current state.
- **Critic:** Evaluates actions and provides rewards (positive for goal proximity, negative for obstacles).

Key Points to Emphasize

- Merges exploratory action selection and careful value evaluation.
- Effective learning in both discrete and continuous action spaces.
- Versatile framework adaptable to various reinforcement learning challenges.

Mathematical Representation

For the value function $V(s)$ estimated by the critic:

$$V(s) \leftarrow V(s) + \alpha \cdot \delta \tag{1}$$

where:

- α is the learning rate.
- $\delta = r + \gamma V(s') - V(s)$ (the TD-error).

The policy update for the actor might resemble:

$$\pi(a|s) \leftarrow \pi(a|s) + \beta \cdot \nabla \log(\pi(a|s)) \cdot \delta \tag{2}$$

where β is the step size for the policy update.

Reinforcement Learning Fundamentals - Overview

Reinforcement Learning (RL) is a paradigm in machine learning focused on decision-making through interaction. Key concepts include:

- Agent
- Environment
- State
- Action
- Reward
- Value Function

Key Concepts in Reinforcement Learning - Part 1

1. Agent

- **Definition**: The entity that makes decisions to achieve maximal cumulative reward.
- **Example**: In a game of chess, the player (or AI) is the agent making moves.

2. Environment

- **Definition**: The setting in which the agent operates, providing feedback based on actions.
- **Example**: The chessboard is the environment where the agent acts.

Key Concepts in Reinforcement Learning - Part 2

3. State (s)

- **Definition**: A specific situation or configuration of the environment at a time.
- **Example**: The arrangement of chess pieces after a move, e.g., “Pawn on E4, Knight on G1".

4. Action (a)

- **Definition**: A decision made by the agent that affects the environment's state.
- **Example**: Moving a Knight from G1 to F3 in chess.

Key Concepts in Reinforcement Learning - Part 3

5. Reward (r)

- **Definition**: A scalar value received by the agent after an action, indicating effectiveness.
- **Example**: Capturing a piece yields a positive reward, while losing one results in a negative reward.

6. Value Function (V(s))

- **Definition**: Estimates the expected cumulative reward of a state s when following a policy.
- **Example**: A high value represents a state likely to lead to winning.

Key Points and Example Scenario

Key Points:

- **Interaction**: RL is about the agent's interaction with the environment and learning from actions.
- **Feedback Loop**: Rewards from actions influence future decisions.
- **Exploration vs. Exploitation**: Agents must balance exploring new actions and exploiting known rewarding actions.

Example Scenario:

- Robot (agent) navigating a maze (environment):
  - **State**: Current position in the maze.
  - **Action**: Turning left, right, moving forward or backward.
  - **Reward**: +10 for reaching the exit, -1 for hitting a wall.
  - **Value Function**: Estimates expected rewards for moving to each position.

Conclusion

Understanding these fundamental concepts is essential for comprehending advanced topics in reinforcement learning, such as Actor-Critic methods. In the next slide, we will explore the specific roles of the Actor and Critic in the RL framework.

Actor-Critic Architecture - Overview

Overview of Actor-Critic Model

The Actor-Critic architecture is a pivotal framework in reinforcement learning, combining both value-based and policy-based approaches. This dual structure allows for more efficient learning and enhanced performance in complex environments.

Actor-Critic Architecture - Key Components

1. **Actor**
   - Learns and improves the policy (mapping from states to actions).
   - Uses policy gradient methods to estimate optimal actions.
   - Updates policy guided by the Critic's feedback.
   - **Formula:**
     $$\theta \leftarrow \theta + \alpha \nabla J(\theta) \tag{3}$$
     where θ are policy parameters, α is the learning rate, and $J(\theta)$ is the performance objective.
2. **Critic**
   - Assesses the Actor's action by evaluating expected future rewards via the value function (V).
   - Computes the Temporal Difference (TD) error.
   - **Formula:**
     $$\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t) \tag{4}$$
     where r_t is reward at time t, γ is the discount factor, $V(s_t)$ is current state value, and $V(s_{t+1})$ is next state value.

Actor-Critic Architecture - Interaction Mechanism

Interaction Mechanism

- The Actor selects actions based on its policy in the environment.
- The Critic evaluates the chosen action and computes the TD error for updating the value function.
- The Actor adjusts its policy based on feedback from the Critic, optimizing its strategy for future actions.

Example

Consider a robot navigating a maze:

- **Actor:** Learns which actions (e.g., go left, turn right) to take to approach the goal.
- **Critic:** Evaluates success of actions based on proximity to the goal, updating the value function accordingly.

Comparison with Value-Based Methods - Overview

Value-Based Methods (e.g., Q-Learning)

- **Definition:** Focus on learning a value function estimating expected return for actions in a given state.
- **Mechanism:** Use the Bellman equation for iterative value estimates.

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left[r + \gamma \max_{a'} Q(s', a') - Q(s, a) \right] \tag{5}$$

- Where:
    - $Q(s, a)$ = value of action a in state s
    - r = reward received
    - γ = discount factor
    - s' = next state
    - α = learning rate

Comparison with Value-Based Methods - Actor-Critic Methods

Actor-Critic Methods

- **Definition:** Combines value-based and policy-based methods with two components: the Actor and the Critic.
- **Mechanism:**
  - **Actor:** Suggests actions based on policy.
  - **Critic:** Evaluates actions and refines value estimates.

Example in Maze Scenario

- The Actor proposes a move (e.g., left or right).
- The Critic assesses the effectiveness of the suggested move based on rewards achieved.

Key Comparisons Between Methods

- **Learning Framework:**
  - Value-Based: Learns a value function (deterministic inference of best action).
  - Actor-Critic: Learns a policy while refining value estimates.
- **Exploration vs. Exploitation:**
  - Value-Based: Vulnerable to suboptimal policies without adequate exploration (e.g., ϵ-greedy).
  - Actor-Critic: Enhances exploration enabling diverse actions even in established states.
- **Convergence:**
  - Value-Based: Slower convergence in large state spaces.
  - Actor-Critic: Faster convergence and better performance in continuous action spaces.

Advantages of Actor-Critic Methods

Actor-Critic methods combine policy estimation (Actor) and value estimation (Critic). This results in:

- Improved efficiency and stability in learning
- Reduced update variance
- Greater expressive power through function approximation
- Flexibility across various environments
- Enhanced sample efficiency with experience replay

Introduction to Actor-Critic Methods

Actor-Critic methods are a hybrid of two reinforcement learning paradigms:

- **Actor**: Responsible for policy improvement, directly updating actions based on feedback.
- **Critic**: Estimates the value function, helping stabilize learning by providing a baseline.

Key Advantages of Actor-Critic Methods

1. **Efficiency in Learning**
   - Direct policy updates via the Actor and value estimation from the Critic.
   - Suitable for continuous action spaces (e.g., robotics).
2. **Reduced Variance**
   - The Critic reduces variance through advantage estimation.
   - Leads to faster convergence and stable learning dynamics.
3. **Expressive Power**
   - Capable of complex function approximation using deep learning for high-dimensional tasks.

Flexibility and Sample Efficiency

4. **Flexibility in Environments**
    - Effective in diverse applications: game playing, robotics, finance.
5. **Improved Sample Efficiency**
    - Utilization of experience replay to maximize learning from past interactions.

Example: Application in Robotics

Consider a robotic arm learning to pick up different objects:

- **Actor**: Generates actions based on current observations (e.g., object positions).
- **Critic**: Evaluates actions by predicting expected future rewards (e.g., success in picking objects).
- Feedback from the Critic improves the Actor's policies leads to better performance over time.

Conclusion and Key Points

Actor-Critic methods provide significant benefits:

- More stable and efficient learning process.
- Capability to handle complex and dynamic environments.
- Effective in continuous action settings due to reduced variance and enhanced expressive power.

Key Points

- The **Actor-Critic structure** combines advantages of both strategies.
- Highly adaptable to a variety of applications.
- **Reduced variance** and **expressive power** are essential for effective performance.

Common Variants of Actor-Critic Methods - Overview

Actor-Critic Methods Overview

Actor-Critic methods are foundational in reinforcement learning (RL) and combine the benefits of policy-based and value-based approaches. They consist of two components:

- **Actor**: Updates the policy.
- **Critic**: Evaluates actions taken by the Actor based on a value function.

Common Variants of Actor-Critic Methods - A2C

Advantage Actor-Critic (A2C)

- **Concept**: A2C improves stability and performance by introducing the advantage function, which measures the relative value of taking a specific action in a state, compared to the average.
- **Formula**:
  $$A(s, a) = Q(s, a) - V(s) \tag{6}$$
  where A is the advantage, Q is the action-value function, and V is the state-value function.
- **Example**: In a game where an agent chooses between "attack" or "defend", A2C evaluates the potential gain of "attack" over "defend" by comparing immediate rewards to expected rewards.

Common Variants of Actor-Critic Methods - DDPG and PPO

Deep Deterministic Policy Gradient (DDPG)

- **Concept**: DDPG is a model-free algorithm tailored for continuous action spaces.
- **Key Features**:
  - **Experience Replay**: Stores past experiences to improve learning efficiency.
  - **Target Networks**: Stabilizes training by reducing the correlation of updates.
- **Example**: In robotic control (e.g., a robotic arm), DDPG can predict and refine precise movements by adjusting continuous joint angles through exploration of the action space.

Proximal Policy Optimization (PPO)

- **Concept**: PPO strikes a balance between sample efficiency and ease of implementation.

Performance Evaluation Techniques

Introduction

Evaluating the performance of Actor-Critic methods is essential for understanding their effectiveness and making improvements. This slide focuses on three key metrics:

- Convergence Rates
- Cumulative Rewards
- Robustness

Convergence Rates

- **Definition:** Convergence rate refers to how quickly an Actor-Critic model approaches its optimal policy.
- **Importance:** Fast convergence is desirable as it reduces training time and resources.
- **Example:** A steep initial rise in the graph of average reward over episodes indicates good convergence.
- **Key Point:** Monitor rewards over time; a flattening curve typically signifies that the model is close to optimal.

Cumulative Rewards

- **Definition:** Cumulative reward is the total reward received by an agent over a certain period, often calculated across episodes.
- **Importance:** Provides insights into how well the policy performs and reflects the effectiveness of the learned strategy.
- **Example:** In a gridworld, if an agent receives $+1$ for reaching a goal, the cumulative reward increases as the agent learns the best path.
- **Key Point:** Higher cumulative rewards indicate a more successful policy. Compare rewards across episodes to evaluate performance.

Robustness and Key Metrics Summary

- **Definition:** Robustness measures how well the Actor-Critic model performs under varying conditions, such as changes in the environment.
- **Importance:** A robust policy ensures consistent performance, even when faced with unexpected scenarios.
- **Example:** An agent trained in a simulated environment performs well in a novel environment, demonstrating robustness.
- **Key Point:** Evaluate robustness by running the model in diverse settings and observing variations in reward and behavior.

Summary of Key Metrics

Metric	Definition
Convergence Rates	Speed of policy stabilization to optimum
Cumulative Rewards	Total rewards over episodes
Robustness	Capability to maintain performance amidst changes in

Formula for Cumulative Reward

$$R = \sum_{t=0}^{T} r_t \tag{8}$$

- Where R is the cumulative reward, r_t is the reward at time step t, and T is the total time steps.

Conclusion

Understanding and evaluating convergence rates, cumulative rewards, and robustness provides essential insights into the performance of Actor-Critic models. Through careful monitoring and analysis of these metrics, we can enhance our reinforcement learning algorithms effectively.

Next, we will explore practical implementation guidelines using popular Python libraries!

Practical Implementation of Actor-Critic Methods

Actor-Critic methods combine policy-based and value-based approaches in Reinforcement Learning. Here, the **Actor** updates the policy based on feedback from the **Critic**, which evaluates the actions taken. Implementation can be done using libraries like **TensorFlow** and **PyTorch**.

Guidelines for Implementation - Setup

1. **Set Up Your Environment:**
   - Ensure Python is installed.
   - Install required libraries:

```
pip install numpy gym tensorflow torch
```

2. **Define the Environment:**
   - Use environments from OpenAI's Gym:

```
import gym
env = gym.make('CartPole-v1')
```

3. **Build Actor and Critic Networks:**
   - Example in TensorFlow:

```
import tensorflow as tf

class Actor(tf.keras.Model):
    def __init__(self, action_size):
        super(Actor, self).__init__()
        self.dense1 = tf.keras.layers.Dense(24,
            activation='relu')
        self.dense2 = tf.keras.layers.Dense(
           action_size, activation='softmax')

    def call(self, state):
        x = self.dense1(state)
        return self.dense2(x)

class Critic(tf.keras.Model):
    def __init__(self):
        super(Critic, self).__init__()
```

Key Points and Resources

- **Actor-Critic Architecture:** Understand how the Actor and Critic interact.
- **Framework Choice:** Use TensorFlow or PyTorch based on personal preference.
- **Performance Metrics:** Focus on convergence rates and cumulative rewards.

Additional Resources

- OpenAI Gym Documentation
- TensorFlow Tutorials
- PyTorch Documentation

Real-World Applications of Actor-Critic Methods - Introduction

Overview

Actor-Critic methods are a type of Reinforcement Learning (RL) approach that utilize two components:

- **Actor**: Determines the best action to take.
- **Critic**: Evaluates the action against a value function.

This combination allows for more stable training and improved policy learning.

Real-World Applications of Actor-Critic Methods - Applications Across Domains

Actor-Critic methods have been applied successfully in various fields. Here are three prominent areas:

1. **Robotics**
    - **Example**: Humanoid Robot Navigation
    - **Outcome**: Enhanced efficiency in movement and adaptability to dynamic environments.
    - *Case Study*: Researchers trained a humanoid robot using Actor-Critic to navigate complex environments.
2. **Finance**
    - **Example**: Automated Trading
    - **Outcome**: Improved decision-making leading to higher returns.
    - *Case Study*: An investment firm employed Actor-Critic for trading algorithms.
3. **Gaming**
    - **Example**: Dynamic Game Agents
    - **Outcome**: More challenging AI opponents enhancing player engagement.
    - *Case Study*: Researchers created AI using Actor-Critic to adapt

Real-World Applications of Actor-Critic Methods - Key Points and Conclusion

Key Points to Emphasize

- **Hybrid Learning**: Actor-Critic allows for efficient learning and policy improvement.
- **Versatility**: Adaptable to various real-world scenarios beyond gaming.
- **Scalability**: Scalable to complex task environments, valuable for advanced applications.

Conclusion

Actor-Critic methods have real-world implications across robotics, finance, and gaming. Ongoing research aims to improve algorithm efficiency and applicability, opening new opportunities for innovation.

Real-World Applications of Actor-Critic Methods - Example Algorithm

Here is a simplified structure of an Actor-Critic algorithm implemented in Python:

```
class ActorCriticAgent:
    def __init__(self, actor_model, critic_model):
        self.actor = actor_model
        self.critic = critic_model

    def train(self, state, action, reward, next_state)
        :
        # Update Critic
        value = self.critic.predict(state)
        next_value = self.critic.predict(next_state)
        td_target = reward + next_value
        td_error = td_target - value
        self.critic.update(state, td_target)

        # Update Actor using TD error
```

Ethical Considerations - Introduction

As we explore the practical applications of Actor-Critic methods in various domains, it is crucial to reflect on the ethical implications of deploying these techniques. This discussion focuses on two primary areas of concern:

- **Bias**
- **Fairness**

Ethical Considerations - Bias

1. Bias

- **Definition:** Bias in machine learning refers to the systematic error that results in unfair outcomes for certain groups or individuals.
- **Sources of Bias:**
  - Data Bias: Training data reflecting historical inequalities.
  - Algorithmic Bias: Model architecture or feature selection may favor certain outcomes.
- **Example:** In financial applications, a trained Actor-Critic model on biased lending data may unfairly deny loans to specific demographics.

Ethical Considerations - Fairness

2. Fairness

- **Definition:** Fairness implies outcomes of ML models should be impartial and just for all individuals, irrespective of their background.
- **Types of Fairness:**
  - Demographic Parity: Decision-making process proportional across groups.
  - Equal Opportunity: Individuals who qualify for positive outcomes have equal chances of receiving them.
- **Example:** In healthcare, an Actor-Critic model must ensure that all patients receive similar evaluations, irrespective of race or socioeconomic status.

Ethical Considerations - Framework

Ethical Framework for Actor-Critic Methods

To ensure ethical deployment:

1. Data Auditing: Regularly inspect and clean datasets.
2. Model Fairness Evaluation: Implement fairness metrics during model evaluation.
3. Stakeholder Engagement: Collaborate with affected communities.
4. Transparency and Accountability: Document model decisions clearly.

Ethical Considerations - Summary and Conclusion

- Recognize Potential Bias: Understand how data and algorithms can lead to biased outcomes.
- Strive for Fairness: Employ diverse fairness metrics to assess impact.
- Ongoing Monitoring: Continuously adapt to new ethical standards post-deployment.

Conclusion: Ethical considerations must be prioritized to promote fairness and mitigate bias to ensure technology serves all of society equitably.