John Smith, Ph.D.

Department of Computer Science
University Name

Email: email@university.edu
Website: www.university.edu

July 13, 2025

Introduction to Actor-Critic Algorithms

Overview of Actor-Critic Architecture

Actor-Critic algorithms in reinforcement learning (RL) combine value-based and policy-based methods, consisting of:

- **Actor**: Selects actions based on the current policy, mapping states to action probabilities and improving the policy during training.
- **Critic**: Evaluates actions taken by the actor by estimating the value function, producing a numerical value (advantage or value estimate) that reflects the quality of actions.

Significance of Actor-Critic Algorithms

- **Combining Strengths**: Balances exploratory and exploitative strategies, enabling faster convergence and better performance than pure methods.
- **Policy Improvement**: Actor adjusts probabilities using feedback from the critic, leading to continuous improvement of the policy.
- **Stability**: Separating value estimation and action selection creates more stable learning dynamics, beneficial for complex environments.

Applications

Actor-Critic algorithms are utilized in:

- **Robotics**: Training robots for tasks like manipulation and navigation.
- **Game Playing**: Achieving high performance in games such as Go and various video games.
- **Finance**: Portfolio management and optimizing strategies in dynamic environments.

Key Concepts in Actor-Critic

- **Learning Process**: Actor and critic learn simultaneously and can update at different rates, adapting learning dynamics to specific problems.
- **Exploration vs. Exploitation**: The stochastic nature of the actor encourages exploration, while the critic supports exploitation of learned actions.

Example

In a grid world, the agent moves toward the goal while avoiding obstacles:

- **Actor**: Chooses among actions (up, down, left, right) based on the current policy.
- **Critic**: Provides a value estimate reflecting the expected reward of reaching the goal.

Mathematical Foundations

Value Estimation

The value of state s is given by:

$$V(s) = E[R_t | s_t = s] \tag{1}$$

where $V(s)$ is the value function and R_t is the reward at time t.

Policy Update

The policy parameters are updated using:

$$\theta_{new} = \theta + \alpha \nabla_\theta J(\theta) \tag{2}$$

where θ are the policy parameters, α is the learning rate, and $J(\theta)$ is the performance objective.

Reinforcement Learning Fundamentals - Overview

- A review of foundational concepts in reinforcement learning (RL).
- Key topics include:
  - Agents and environments
  - States and actions
  - Rewards and policies

Understanding Key Concepts in Reinforcement Learning

- **Agent**:
  - Definition: The learner or decision maker.
  - Role: Choose actions to maximize cumulative rewards.
  - Example: A robot navigating a maze.
- **Environment**:
  - Definition: Everything the agent interacts with.
  - Role: Responds to the agent's actions.
  - Example: The maze including walls and paths.
- **State (s)**:
  - Definition: Description of the current situation.
  - Role: Influences agent behavior.
  - Example: The robot's location in the maze.

Continuing Key Concepts in RL

- **Action (a)**:
  - Definition: A choice made by the agent.
  - Role: Represents moves the agent can make.
  - Example: Move forward, turn left/right.
- **Reward (r)**:
  - Definition: Feedback signal after an action.
  - Role: Indicates success or failure, guiding learning.
  - Example: +1 for reaching exit, -1 for hitting wall.
- **Policy (π)**:
  - Definition: A strategy that defines agent behavior.
  - Role: Maps states to actions.
  - Example: "If at (3, 5), turn right."

Reinforcement Learning Process

1. **Perceive State**: Observe the environment's current state.
2. **Select Action**: Choose an action based on policy.
3. **Receive Reward**: Environment provides a reward and state transition.
4. **Update Policy**: Adjust policy based on received reward.

Cumulative Reward (Return)

$$G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots \tag{3}$$

Total reward accumulated over time, essential for evaluating performance.

Key Points and Conclusion

- **Feedback Loop**: Agent learns through trial and error, optimizing policy based on experiences.
- **Exploration vs Exploitation**: Balancing new actions vs known rewarding actions.

Conclusion

Understanding these concepts is crucial for diving into actor-critic methods—where the roles of the agent split into distinct functions.

The Actor-Critic Framework - Overview

- The Actor-Critic framework merges value-based and policy-based methods in reinforcement learning.
- Key components:
  - **Actor**: Learns the policy mapping states to actions.
  - **Critic**: Evaluates the Actor's actions and estimates the value function.
- This interaction optimizes learning and policy performance.

The Actor-Critic Framework - Roles

1. The Actor

- **Purpose**: Learns the policy to decide actions based on the current state.
- **Feedback Mechanism**: Updates its parameters through feedback from the Critic.
- **Example**: Navigating a maze by proposing moves that lead to rewards.

2. The Critic

- **Purpose**: Evaluates the actions chosen by the Actor by computing the value function.
- **Feedback Mechanism**: Utilizes the Temporal-Difference error for updates.
- **Example**: Assessing the effectiveness of the Actor's moves during maze navigation.

The Actor-Critic Framework - Interaction and Formulas

- **Interaction**:
  - The Actor adjusts its policy based on the Critic's evaluations.
  - Forms a feedback loop improving policy and action choices over time.
- **Key Formulas**:

$$\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t) \tag{4}$$

$$\theta \leftarrow \theta + \alpha \delta_t \nabla \log \pi(a_t|s_t;\theta) \tag{5}$$

  - δ_t: Temporal-Difference error
  - θ: Policy parameters
  - α: Learning rate

Actor vs. Critic: Key Differences - Function

- **Actor:**
  - **Definition:** The actor selects actions based on the current policy.
  - **Role:** Maps states to actions (policy function).
  - **Output:** Produces action probabilities (e.g., $\pi(a|s)$).
- **Critic:**
  - **Definition:** Evaluates the actions taken by the actor, providing feedback.
  - **Role:** Estimates the value function (either $V(s)$ or $Q(s, a)$).
  - **Output:** Provides a scalar value (e.g., $V(s)$).

Actor vs. Critic: Key Differences - Learning Objectives

- **Actor:**
  - **Objective:** Optimize the policy directly by improving action selection over time.
  - **Goal:** Maximize expected return from each state by adjusting π based on feedback from the critic.
- **Critic:**
  - **Objective:** Evaluate the performance of the actor by estimating value of states and actions.
  - **Goal:** Minimize the difference between estimated values and observed returns (temporal-difference learning).

Actor vs. Critic: Key Differences - Algorithmic Implementation

- **Actor:**
  - **Method:** Uses policy gradient methods to adjust policy parameters θ.
  - **Update Rule:**
$$\theta \leftarrow \theta + \alpha \nabla \log \pi(a|s;\theta) \cdot (G_t - V(s;w)) \tag{6}$$
- **Critic:**
  - **Method:** Utilizes value-based techniques, such as Temporal Difference (TD) learning.
  - **Update Rule:**
$$w \leftarrow w + \beta(G_t - V(s;w))\nabla V(s;w) \tag{7}$$
- **Key Points:**
  - Complementary roles create a feedback loop for policy and value refinement.
  - Separation of policy improvement and value estimation enhances stability and efficiency.

Actor-Critic Architecture

Concept Overview

Actor-Critic methods combine two key components in reinforcement learning:

- **Actor:** Proposes actions based on a policy.
- **Critic:** Evaluates the action taken by assessing its value.

This dual architecture enables efficient exploration of the action space while stabilizing the learning process.

Advantages of Actor-Critic Methods - Stability and Sample Efficiency

1. **Stability**
   - Actor-Critic methods stabilize training by addressing variance in policy evaluation.
   - **Example:** In variable environments like video games, the Critic reduces reward signal fluctuations, allowing reliable learning.
2. **High Sample Efficiency**
   - Fewer interactions with the environment are required to learn effective policies.
   - **Example:** In robotic tasks, Actor-Critic methods achieve optimal movements using fewer trials than traditional methods.

Advantages of Actor-Critic Methods - Direct Optimization and Key Points

3. **Directly Optimizing Policy**
   - These methods update policy parameters directly, leading to smoother updates.
   - **Example:** In navigation tasks, direct action evaluation by the Critic facilitates optimal decision-making.

Key Points to Emphasize

- Efficient blending of policy gradient and value function methods enhances performance.
- Robust against fluctuations in less stable reward environments.
- Adaptable architectures using modern enhancements like DDPG and A3C for complex problems.

Formulas and Code Snippets

Policy Gradient Update

The policy for the Actor can be updated using the gradient of the expected return:

$$\nabla J(\theta) \approx \mathbb{E}_t \left[\nabla \log \pi_\theta(a_t|s_t) A_t \right] \tag{8}$$

Where:

- $J(\theta)$ is the objective function.
- $\pi_\theta(a_t|s_t)$ is the policy.
- A_t is the advantage function.

Value Update for Critic

The Critic refines its value estimate through temporal difference learning:

$$V(s_t) \leftarrow V(s_t) + \alpha(R_t + \gamma V(s_{t+1}) - V(s_t)) \tag{9}$$

Types of Actor-Critic Algorithms

Actor-Critic algorithms combine the benefits of policy-based and value-based reinforcement learning. They consist of two main components:

- **Actor**: Updates the policy based on feedback from the critic.
- **Critic**: Evaluates the action taken by the actor and provides feedback for improvement.

Key Variants of Actor-Critic Algorithms

1. **A3C (Asynchronous Actor-Critic Agents)**
   - *Overview*: Uses multiple agents exploring in parallel and updates a shared model.
   - *Key Points*:
     - Asynchronous Updates: Faster and more stable convergence.
     - Entropy Regularization: Encourages exploration.
   - *Example Use Case*: Video games with simultaneous agent learning.
2. **DDPG (Deep Deterministic Policy Gradient)**
   - *Overview*: Designed for continuous action spaces with off-policy learning.
   - *Key Points*:
     - Actor-Critic Architecture: Actor selects actions, critic evaluates them.
     - Replay Buffer: Stores transitions to improve learning efficiency.
   - *Example Use Case*: Robotics control tasks requiring precise actions.

More Actor-Critic Variants

3 PPO (Proximal Policy Optimization)

- *Overview*: Balances performance with implementation simplicity.
- *Key Points*:
  - Clipped Objective Function: Prevents destabilizing updates.
  - Trust Region Updates: Ensures stable learning by limiting policy deviation.
- *Example Use Case*: Robust training in continuous or discrete environments.

4 TRPO (Trust Region Policy Optimization)

- *Overview*: Optimizes policy with constraints for stable updates.
- *Key Points*:
  - Constrained Optimization: Balances exploration and exploitation.
  - Natural Gradients: Enhances stability in updates.
- *Example Use Case*: Complex continuous control tasks like drone flight.

Key Considerations

- **Sample Efficiency**: Higher sample efficiency compared to pure actor or critic methods.
- **Stability**: Combined actor and critic structure allows for stable convergence.
- **Exploration vs. Exploitation**: Critical for success; balancing new and known actions is essential.

Conclusion: Each variant of Actor-Critic algorithms has unique advantages tailored for diverse reinforcement learning problems.

Implementation Steps for Actor-Critic Algorithms

Overview

Actor-Critic algorithms combine policy-based and value-based methods. The **actor** updates the policy, guided by the **critic**, which evaluates actions and provides value feedback.

Implementation Steps - Initialization

1. **Define the Environment**
   - Specify states and actions (e.g., grid locations and movements).
2. **Initialize Actor and Critic Networks**
   - Choose neural network architectures and initialize weights.
3. **Set Hyperparameters**
   - Learning rates for actor and critic.
   - Discount factor (γ).
   - Exploration strategy (e.g., ϵ-greedy).
4. **Example**
   - For DDPG: Set up a deep neural network with separate heads for actor and critic.

Implementation Steps - Training Updates

2. **Collect Experience**
   - Generate actions and store states, actions, rewards, and next states.
3. **Compute Returns**

$$R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots \tag{10}$$

4. **Critic Update**

$$L(\theta^{critic}) = \frac{1}{N} \sum_{t=0}^{N} (R_t - V(s_t; \theta^{critic}))^2 \tag{11}$$

5. **Actor Update**

$$\nabla J(\theta^{actor}) \approx \frac{1}{N} \sum_{t=0}^{N} \nabla \log \pi(a_t | s_t; \theta^{actor})(R_t - V(s_t; \theta^{critic})) \tag{12}$$

6. **Example**
   - Use the critic's feedback to refine the actor's policy based on past actions.

Implementation Steps - Performance Evaluation

3. **Testing the Policy**
   - Evaluate learning performance using total rewards over episodes.
4. **Monitor Learning Metrics**
   - Plot average rewards, successful episodes, and episodic returns.
5. **Fine-Tuning**
   - Tune hyperparameters and architectures if performance plateaus.
6. **Key Points to Monitor**
   - Convergence speed and stability of updates.
   - Variance of returns.
   - Risks of overfitting or underfitting.

Summary

Implementing an Actor-Critic algorithm involves:

- Initializing models and parameters.
- Collecting experiences and optimizing through updates.
- Continuously evaluating agent performance for effective learning.

Additional Notes

- Use frameworks like TensorFlow or PyTorch for efficient implementation.
- Balance exploration and exploitation for effective learning.

Case Study: Actor-Critic in Practice

Overview

Actor-Critic algorithms integrate strengths of policy-based and value-based methods in reinforcement learning. We explore how an actor-critic algorithm optimized the energy consumption of heating systems in smart buildings.

Actor-Critic Architecture

- **Actor**: Learns and updates the policy function, deciding actions based on the current state.
- **Critic**: Evaluates the action taken by the actor with a value function, providing feedback to enhance policy.

Heating System Optimization Case Study

Context

Smart buildings require efficient energy management for cost reduction and environmental sustainability. An actor-critic algorithm was implemented to optimize heating in a commercial building.

Implementation Steps

1. **Initialization**
   - Define state representation: temperature, outside conditions, occupancy.
   - Initialize actor (policy) and critic (value) networks.
2. **Training**
   - State: constructed using sensory data.
   - Action Selection: actor uses neural network to choose actions (e.g., adjust heating).
   - Feedback: critic evaluates action based on fluctuations and consumption using:

Results and Key Points

Results

- **Improved Energy Efficiency**: Reduced consumption by up to 20%.
- **User Comfort**: Maintained optimal temperatures for occupant satisfaction.

Key Points

- **Adaptability**: Actor-Critic model adjusts to occupancy and weather changes.
- **Real-time Learning**: Demonstrated ability to learn and refine actions continuously.

Conclusion

Actor-Critic algorithms show promise in practical applications, enhancing energy efficiency in smart buildings for sustainable urban development.

Challenges and Limitations of Actor-Critic Algorithms - Introduction

- Actor-Critic algorithms blend value-based and policy-based reinforcement learning methods.
- Despite their successes, they face several challenges.
- This slide will explore the key issues impacting their effectiveness.

Key Challenges and Limitations - Part 1

1. **Stability and Convergence Issues**
   - Divergence between Actor and Critic hinders stable learning.
   - Can slow down learning or prevent convergence to an optimal policy.
   - *Example:* Overestimation by Critic leads to erratic policy updates.
2. **Variance in Gradient Estimates**
   - High variance in policy gradient estimates affects learning consistency.
   - *Example:* Noisy value estimations from the Critic yield unpredictable actions.

Key Challenges and Limitations - Part 2

1. **Complexity of Hyperparameter Tuning**
   - Requires careful tuning of learning rates for Actor and Critic.
   - Suboptimal hyperparameters can degrade performance.
   - *Example:* High learning rates may overshoot optimal policies; low rates can delay convergence.
2. **Sample Efficiency**
   - Many methods require substantial data for effective learning.
   - *Example:* In costly interaction scenarios (e.g., robotics), prolonged training is impractical.
3. **Function Approximation Limitations**
   - Use of function approximators can lead to generalization issues.
   - Poor generalization may cause overfitting to specific states.
   - *Example:* Insufficient exploration of the action space leads to missing optimal long-term strategies.

Future Directions in Actor-Critic Research - Overview

Overview

Actor-Critic algorithms are pivotal in reinforcement learning (RL) for their unique ability to merge value-based and policy-based strategies. As research progresses, future directions focus on enhancing algorithm performance and applicability.

Future Directions in Actor-Critic Research - Key Trends

1. **Integration with Deep Learning**
   - Utilization of deep neural networks enhances scalability and flexibility.
   - Example: Recurrent neural networks (RNNs) are integrated to manage sequential data effectively.
2. **Multi-Agent Architectures**
   - Focus on multiple actors learning simultaneously in competitive or cooperative settings.
   - Example: AI agents in gaming environments learn strategies against each other.

Future Directions in Actor-Critic Research - Continued Trends

3. **Sample Efficiency and Offline RL**
   - Aim to improve learning from limited data using offline datasets.
   - Example: Dataset Aggregation (D4PG) utilizes offline experiences to enhance learning efficiency.
4. **Exploration Strategies**
   - Developing adaptive techniques for improved exploration, crucial for enhancing performance.
   - Example: Actors incentivized to explore uncertain states discover rewarding regions effectively.

Future Directions in Actor-Critic Research - Final Trends and Conclusion

5. **Hierarchical RL**
   - Decomposes tasks into subtasks with distinct actor-critic pairs, enhancing learning efficiency.
   - Example: A robot learns to navigate and then pick up objects as separate episodes with dedicated models.
6. **Improving Stability and Convergence**
   - Ongoing methods aim to enhance stability and convergence rates of algorithms.
   - Example: Techniques like Trust Region Policy Optimization (TRPO) modify updates for stable learning.

Conclusion

The future of actor-critic research is filled with promise, driven by AI advancements and methodological innovations that will bolster robustness and efficiency in reinforcement learning systems.