Actor-Critic Methods in Reinforcement Learning

Project Overview

This project demonstrates the implementation of **Actor-Critic methods** in Reinforcement Learning (RL) using Python, TensorFlow, and OpenAI Gym. Actor-Critic methods combine the advantages of both policy-based and value-based approaches, enabling efficient learning in complex environments.

Why Use Actor-Critic Methods?

- **Combines Policy and Value Learning**: Integrates both policy optimization (actor) and value estimation (critic) for improved performance.
- **Reduced Variance**: The critic helps in reducing the variance of policy updates, leading to more stable learning.
- **Efficient Learning**: Suitable for environments with continuous action spaces and high-dimensional state spaces.

Prerequisites

Required Libraries

- `numpy`: For numerical computations.
- `gym`: For environment simulations.
- `tensorflow`: For building and training neural networks.

Installation

Install the necessary libraries using pip:

```
pip install numpy gym tensorflow
```

Files Included

- `actor_critic.py`: The Python script implementing the Actor-Critic algorithm.

Code Description

The implementation is divided into several key steps:

1. Importing Libraries

```
 import numpy as np
import gym
import tensorflow as tf
from tensorflow.keras import layers
```

2. Setting Hyperparameters

```
gamma = 0.99  # Discount factor
learning_rate = 0.01
```

3. Creating Neural Networks for Actor and Critic

```
def create_actor_network():
    model = tf.keras.Sequential([
        layers.Dense(128, activation='relu'),
        layers.Dense(env.action_space.n, activation='softmax')
    ])
    return model

def create_critic_network():
    model = tf.keras.Sequential([
        layers.Dense(128, activation='relu'),
        layers.Dense(1)
    ])
    return model
```

4. Initializing Models and Optimizer

```
actor_model = create_actor_network()
critic_model = create_critic_network()
optimizer = tf.keras.optimizers.Adam(learning_rate)
```

5. Defining the Training Step

```
def train_step(state, action, reward, next_state, done):
    with tf.GradientTape() as tape:
        state = tf.convert_to_tensor(state)
        next_state = tf.convert_to_tensor(next_state)
        logits = actor_model(state)
        action_prob = logits[0, action]
        value = critic_model(state)
        next_value = critic_model(next_state)
        td_target = reward + gamma * next_value * (1 - done)
        td_error = td_target - value

        # Actor loss (policy gradient)
        actor_loss = -tf.math.log(action_prob) * td_error
        # Critic loss (value prediction)
        critic_loss = td_error ** 2

        total_loss = actor_loss + critic_loss

    grads = tape.gradient(total_loss, actor_model.trainable_variables + critic_model.tra
    optimizer.apply_gradients(zip(grads, actor_model.trainable_variables + critic_model
```

6. Training the Actor-Critic Model

```
def actor_critic(env, n_episodes=1000):
    for episode in range(n_episodes):
        state = env.reset()
        done = False
        while not done:
            state = np.expand_dims(state, axis=0)
            action_probs = actor_model(state)
            action = np.random.choice(np.arange(env.action_space.n), p=action_probs[0].n
            next_state, reward, done, _ = env.step(action)

            train_step(state, action, reward, next_state, done)

            state = next_state

env = gym.make('CartPole-v0')
actor_critic(env)
```

Expected Outputs

- **Trained Agent**: An agent capable of balancing the pole in the CartPole environment.
- **Learning Curve**: Observation of the agent's performance improving over episodes.

Use Cases

- **Robotics**: Learning control policies for robotic arms.
- **Game Playing**: Developing strategies for complex games.
- **Autonomous Vehicles**: Decision-making in dynamic environments.

Future Enhancements

- **Hyperparameter Tuning**: Experiment with different learning rates and network architectures.
- **Advanced Algorithms**: Implement more sophisticated Actor-Critic variants like A3C or PPO.
- **Environment Exploration**: Apply the algorithm to more complex environments.

References

- Playing CartPole with the Actor-Critic method - TensorFlow
- Actor-Critic Algorithm in Reinforcement Learning - GeeksforGeeks

---