Documentation of DQLAgent Implementation

Your Name

April 10, 2024

1 Introduction

This document outlines the implementation details of the `DQLAgent` class designed for the OpenAI gym's 'CartPole-v1' environment. The agent uses a Deep Q-Learning approach, balancing the exploration and exploitation to achieve better learning outcomes.

2 Agent Initialization

The `DQLAgent` class initializes with the environment, setting up parameters and hyperparameters necessary for the neural network, memory buffer, and the epsilon-greedy strategy for action selection.

2.1 Neural Network Model

The neural network model consists of two layers, an input layer with a `tanh` activation function and an output layer with a `linear` activation, constructed using Keras:

```
def build_model(self):
    model = Sequential()
    model.add(Dense(48, input_dim = self.state_size, activation =
        'tanh'))
    model.add(Dense(self.action_size, activation = 'linear'))
    model.compile(loss = 'mse', optimizer = Adam(learning_rate =
        self.learning_rate))
    return model
```

2.2 Memory and Action Selection

The agent stores experiences in a replay buffer and uses an epsilon-greedy strategy for action selection, allowing for both exploration of the environment and exploitation of known states.

2.3 Learning and Adjustments

Through the replay mechanism, the agent iteratively learns from past experiences, adjusting the neural network weights. The adaptive epsilon decay further refines the balance between exploration and exploitation over time.

3 Experiment

A brief experiment with the ‘CartPole-v1‘ environment demonstrates the agent’s ability to learn and sustain the pole balance for increasing durations over episodes.

4 Conclusion

The `DQLAgent` represents a robust approach to solving reinforcement learning problems using deep Q-learning, showing promising results in the ‘CartPole-v1‘ task.