HW5: Advanced DRL

Prompt

Introduce the **Highway task** in the **HighwayEnv** project and the DQN solution. Analyze the execution results of `main.ipynb` and generate a report.

Related links:

- https://github.com/Farama-Foundation/HighwayEnv
- https://highway-env.farama.org/quickstart/
- https://highway-env.farama.org/environments/highway/

Result

Report: Highway Task in highway-env and DQN Training Analysis

Project Introduction

`highway-env` is a **minimalist collection of environments for autonomous driving and tactical decision-making tasks**. Developed and maintained by Edouard Leurent, the project is supported by the Farama Foundation.

The Highway Task

The Highway task is one of the environments included in `highway-env`. In this environment, an ego-vehicle drives on a multi-lane highway populated by other vehicles. The **agent's primary objective is to achieve high speed while preventing collisions with surrounding vehicles**. Additionally, **staying in the rightmost lane is rewarded** in the Highway task.

The environment can be created using `gymnasium.make("highway-v0")`. For faster training at scale, a quicker version, `highway-fast-v0`, is available, although it has reduced simulation accuracy.

Default configuration for the Highway environment includes:

- Observation type: "Kinematics"
- Action type: "DiscreteMetaAction"
- Number of lanes (`lanes_count`): 4
- Number of vehicles (`vehicles_count`): 50
- Duration: 40 seconds
- Collision reward: -1
- Speed range for speed reward: \ m/s
- Simulation frequency: 15 Hz
- Policy frequency: 1 Hz
- Type of other vehicles: `"highway_env.vehicle.behavior.IDMVehicle"`

These settings can be modified via the environment's `config` attribute or by passing a configuration dictionary during creation. `env.reset()` must be called after changing the configuration for the changes to take effect.

Solving the Highway Task with DQN

Reinforcement learning agents can be trained to tackle tasks within the `highway-env` suite using libraries such as `eleurent/rl-agents`, `openai/baselines`, or **Stable Baselines3 (SB3)**.

The **Deep Q-Network (DQN)** is a model-free, value-based reinforcement learning algorithm. It approximates the state-action value function Q using a neural network.

The source material provides an example of training an **SB3 DQN agent on the `highway-fast-v0` environment**. This example utilizes the default kinematics observation and an MLP (Multilayer Perceptron) model. The MLP network in the example is structured with two layers, each containing 256 units.

The general training process using the example code involves:

1. Importing necessary libraries: `gymnasium`, `highway_env`, `stable_baselines3.DQN`.
2. Creating the `highway-fast-v0` environment.
3. Defining the DQN model, specifying the policy type ('MlpPolicy') and the environment, and setting hyperparameters like learning rate, buffer size, learning start, batch size, and gamma.
4. Calling the `model.learn()` method to initiate training for a specified number of total timesteps (e.g., 2e4).
5. Saving the trained model using `model.save()`.
6. Loading the saved model with `model.load()` for testing, which involves repeatedly resetting the environment, predicting actions using `model.predict(obs, deterministic=True)`, and executing actions via `env.step(action)`.

It is noted that the basic DQN implementation in SB3 lacks advanced features like Double-DQN, Dueling-DQN, and Prioritized Experience Replay. Using the DQN implementation from `eleurent/rl-agents` could provide these extensions for potentially better performance. Furthermore, modifying the observation type or model architecture can also lead to improved results. The source briefly mentions using image observations and CNN models as alternatives.

DQN Training Execution Results Analysis Report

Analyzing the provided training log data, we can observe changes in key metrics during the DQN training process. The logs track `rollout/` and `train/` metrics at various training stages, indicated by episode counts and total timesteps.

Analysis Summary:

- **Episodes and Total Timesteps**: The logs begin at episodes 12 / total_timesteps 151 and continue up to episodes 1208 / total_timesteps 19995, showing the progression of the training.
- **FPS**: Frames per second during training remained largely consistent, around 23 or 24.
- **Time Elapsed**: The elapsed time increased from 6 seconds to 805 seconds, indicating the total time spent on training.
- **Learning Rate**: The learning rate appears constant at 0.0005 throughout the provided logs.
- **Loss**: The training loss showed fluctuations across updates, without a clear monotonic decrease pattern, but later values are generally smaller (e.g., 0.123 at episodes 1112 / total_timesteps 17836 and 0.013 at episodes 1208 / total_timesteps 19995). This suggests the model was learning and attempting to reduce its prediction error.
- **n_updates**: The number of model updates increased in tandem with total timesteps, from 6 to 19794.

- **Exploration Rate**: The **exploration rate decreased progressively from an initial value of 0.99**. **In the later stages of training (roughly after episodes 936 / total_timesteps 13887)**, **it stabilized at 0.05**. This pattern aligns with the typical DQN training approach, transitioning from initial exploration (high exploration rate) to exploitation of learned strategies (low exploration rate).
- **Mean Episode Length (ep_len_mean)**: The **average episode length increased over the course of training**. It rose from approximately 12.6 in the early stages to about 22.5 towards the end. This indicates that the agent became capable of surviving for longer durations within the environment, thus delaying the end of an episode (which typically occurs due to a collision or reaching the maximum duration).
- **Mean Episode Reward (ep_rew_mean)**: The **average episode reward also increased with training progress**. It grew from about 9.38 initially to around 18.7 in the later episodes. This suggests the agent obtained higher cumulative rewards within each episode, likely achieved by a combination of maintaining high speed, staying in the right lane, and effectively avoiding collisions.

Conclusion:

Based on the provided training logs, the DQN agent trained on the `highway-fast-v0` environment demonstrated **clear performance improvements**. Both the average episode length and the average episode reward showed steady increases as the number of training timesteps and episodes grew. This signifies that the agent successfully learned how to navigate the environment more effectively, securing greater rewards and preventing premature episode termination (e.g., through collisions). The observed decrease in the exploration rate also reflects the training process shifting from exploring unknown strategies to utilizing the learned, more optimal ones.