CarRacing Reinforcement Learning Project

Descrição

Este projeto demonstra o treinamento de um agente de aprendizado por reforço no ambiente `CarRacing` do `gymnasium`. Usando a biblioteca `stable_baselines3`, o agente aprende a navegar em uma pista de corrida, visando maximizar a recompensa ao completar voltas corretamente. O projeto é disponibilizado em dois formatos: um notebook Jupyter (`Car_Racing.ipynb`) para experimentação interativa e um script Python (`car_racing.py`) para execução direta.

Implementação do PPO no Projeto

O Proximal Policy Optimization (PPO) é um dos algoritmos de aprendizado por reforço mais populares e eficazes, conhecido por seu equilíbrio entre facilidade de implementação, eficiência computacional e desempenho. O algoritmo busca otimizar políticas de decisão de forma que maximize a recompensa total acumulada pelo agente, ao mesmo tempo que mantém as atualizações de política relativamente conservadoras para evitar grandes flutuações no desempenho.

Como o PPO Funciona no Código

No notebook `Car_Racing.ipynb`, foi utilizada a biblioteca stable_baselines3 para implementar o PPO. O processo é dividido em várias etapas-chave, detalhadas a seguir:

1. **Inicialização do Ambiente:** O primeiro passo envolve a configuração do ambiente `CarRacing` utilizando a biblioteca `gymnasium`. Esse ambiente simula um carro em uma pista de corrida, onde o agente precisa aprender a navegar.

2. **Definição do Modelo PPO:** Foi utilizada a classe `PPO` disponibilizada pela `stable_baselines3`. Aqui, foi configurado o algoritmo com parâmetros específicos, como a arquitetura da rede neural que representa a política do agente e os parâmetros de otimização. O trecho de código relevante cria um objeto `PPO` ligado ao nosso ambiente, preparando-o para o treinamento:

   ```
   from stable_baselines3 import PPO

   model = PPO("MlpPolicy", env, verbose=1)
   ```

   Neste caso, `MlpPolicy` indica o uso de uma rede neural multicamadas (MLP - Multi-Layer Perceptron) para modelar a política do agente. `env` é o ambiente `CarRacing`, e `verbose=1` habilita a saída de logs detalhados durante o treinamento.

3. **Treinamento do Modelo:** Com o modelo definido, o próximo passo é o treinamento. Isso é realizado através da chamada do método `.learn()`, que recebe como argumento o número total de passos de tempo para treinar o agente. Durante o treinamento, o agente interage com o ambiente, coleta experiências, e periodicamente atualiza sua política com base no algoritmo PPO.

   ```
   ```

```
model.learn(total_timesteps=100000)
```

4. **Avaliação e Ajuste:** Após o treinamento, o desempenho do agente pode ser avaliado executando o modelo treinado no ambiente. Isso oferece a oportunidade de observar o comportamento do agente e fazer ajustes necessários, seja modificando os parâmetros do PPO ou o processo de treinamento.

A implementação do PPO no projeto é um exemplo prático de como configurar e treinar um agente de aprendizado por reforço para resolver uma tarefa complexa de controle, demonstrando a eficácia do algoritmo em aprender políticas de decisão eficientes.

Instalação

Para executar este projeto, é necessário ter Python 3.7+ instalado. Primeiro, clone o repositório ou baixe os arquivos `Car_Racing.ipynb` para executar em um ambiente interativo de notebooks e/ou `car_racing.py` se quiser executar o script Python diretamente. Então, instale as dependências necessárias executando:

```
pip install gymnasium[box2d] matplotlib stable_baselines3
```

> ⚠ Existe uma célula já implementada no notebook (`Car_Racing.ipynb`), não sendo necessária a execução do comando externamente.

Uso

Notebook Jupyter

Para explorar o projeto no formato de notebook, assegure-se de ter o Jupyter Lab ou o Jupyter Notebook instalado. Você pode iniciar o ambiente Jupyter e abrir o arquivo `Car_Racing.ipynb` para execução.

Script Python

Para executar o script Python diretamente, utilize o seguinte comando no terminal:

```
python car_racing.py
```