Московский государственный технический университет

им. Н.Э. Баумана

Факультет «Информатика и системы управления»

Кафедра «Системы обработки информации и управления»



**Лабораторная работа № 6**

**По курсу «методы машинного обучения в АСОИУ»**

**«Обучение на основе глубоких Q-сетей»**

**Выполнила:**

студентка ИУ5-23М

Светашева Ю.В.

**Проверил:**

Балашов А.М.

Подпись:

29.04.2024

Москва, 2024

**Описание задания**

* На основе рассмотренных на лекции примеров реализуйте алгоритм DQN.
* В качестве среды можно использовать классические среды (в этом случае используется полносвязная архитектура нейронной сети).
* В качестве среды можно использовать игры Atari (в этом случае используется сверточная архитектура нейронной сети).
* В случае реализации среды на основе сверточной архитектуры нейронной сети +1 балл за экзамен.

**Описание алгоритма**

DQN (Deep Q-Network) — это метод обучения с подкреплением, который объединяет идеи Q-обучения и нейронных сетей для улучшения производительности агента в сложных средах с большим количеством состояний. DQN заменяет стандартную Q-матрицу на нейронную сеть, которая приближает Q-функцию и позволяет агенту справляться с более сложными средами.

**Основные компоненты DQN:**

Нейронная сеть: Нейронная сеть используется для приближения Q-функции, где входом являются состояния среды, а выходом — оценка Q-значений для всех возможных действий.

Целевая сеть: Для повышения стабильности обучения используется целевая нейронная сеть, которая обновляется периодически из основной сети.

Опытный буфер (Replay Buffer): Буфер, в который агент записывает свой опыт (состояние, действие, награду, новое состояние) и из которого случайно извлекает данные для обучения.

Мини-партии: Обучение нейронной сети происходит на случайных мини-партиях из опыта буфера, что помогает уменьшить корреляцию между последовательными данными и повышает стабильность обучения.

Лосс-функция: DQN использует функцию потерь Huber для более стабильного обучения. Она сравнивает предсказанные Q-значения с целевыми значениями, вычисляемыми на основе награды и оценок целевой сети.

Описание алгоритма:

Алгоритм DQN (Deep Q-Network) представляет собой метод обучения с подкреплением, в котором используется глубокая нейронная сеть для обучения агента взаимодействовать с окружающей средой и максимизировать ожидаемую награду. Алгоритм основан на методе Q-обучения, который является одним из популярных методов обучения с подкреплением. В DQN используется нейронная сеть для аппроксимации функции Q-значений, что позволяет агенту принимать оптимальные решения в сложных средах.

Вот подробное описание алгоритма DQN:

1. Инициализация среды и параметров:

- Агент инициализирует среду (в данном случае, `CartPole-v1` из библиотеки OpenAI Gym).

- Устанавливаются параметры алгоритма, такие как `gamma` (коэффициент дисконтирования), `epsilon` (вероятность случайного выбора действия), `epsilon\_min` (минимальное значение `epsilon`), `epsilon\_decay` (скорость снижения `epsilon`), `learning\_rate` (скорость обучения модели), `batch\_size` (размер батча для обучения), `memory\_size` (размер буфера опыта), `train\_start` (минимальное количество опыта для начала обучения) и `update\_target\_steps` (частота обновления целевой сети).

2. Создание моделей:

- Создаются две нейронные сети: основная (`main\_model`) и целевая (`target\_model`).

- Эти сети используются для предсказания Q-значений действий в различных состояниях.

- Целевая сеть используется для стабилизации процесса обучения, она периодически обновляется, копируя веса из основной сети.

3. Буфер опыта:

- Буфер опыта (`ReplayMemory`) используется для хранения опыта агента в виде кортежей (`state`, `action`, `reward`, `next\_state`, `done`).

- Буфер ограничен по размеру (`memory\_size`), и старые данные заменяются новыми, если достигается его предельный размер.

4. Главный цикл обучения:

- Проходит через заданное количество эпизодов.

- В каждом эпизоде агент начинает в начальном состоянии и взаимодействует со средой, выбирая действия на основе `epsilon-greedy` политики.

- Если случайное значение (`np.random.rand()`) меньше `epsilon`, агент выбирает случайное действие из доступных в среде.

- В противном случае, агент выбирает действие с наибольшим предсказанным Q-значением из основной модели (`main\_model`).

- Агент выполняет выбранное действие, получает награду и наблюдает новое состояние.

- Опыт (`state`, `action`, `reward`, `next\_state`, `done`) сохраняется в буфере опыта.

- Если количество опыта в буфере больше или равно `train\_start`, начинается обучение:

- Случайно выбирается батч из опыта в буфере.

- Основная сеть предсказывает Q-значения для текущих состояний и целевых состояний.

- Обновляются Q-значения, учитывая полученные награды и ожидаемые Q-значения целевой сети.

- Основная сеть обучается на основе скорректированных Q-значений.

- Если `step\_count` кратен `update\_target\_steps`, целевая сеть обновляется копированием весов из основной сети.

- `epsilon` уменьшается по мере прогресса обучения до минимального значения (`epsilon\_min`).

5. Вывод результатов:

- В конце каждого эпизода выводится его номер и общая награда, полученная агентом за эпизод.

- Если эпизод заканчивается (достигается условие завершения эпизода), обучение продолжается со следующего эпизода.

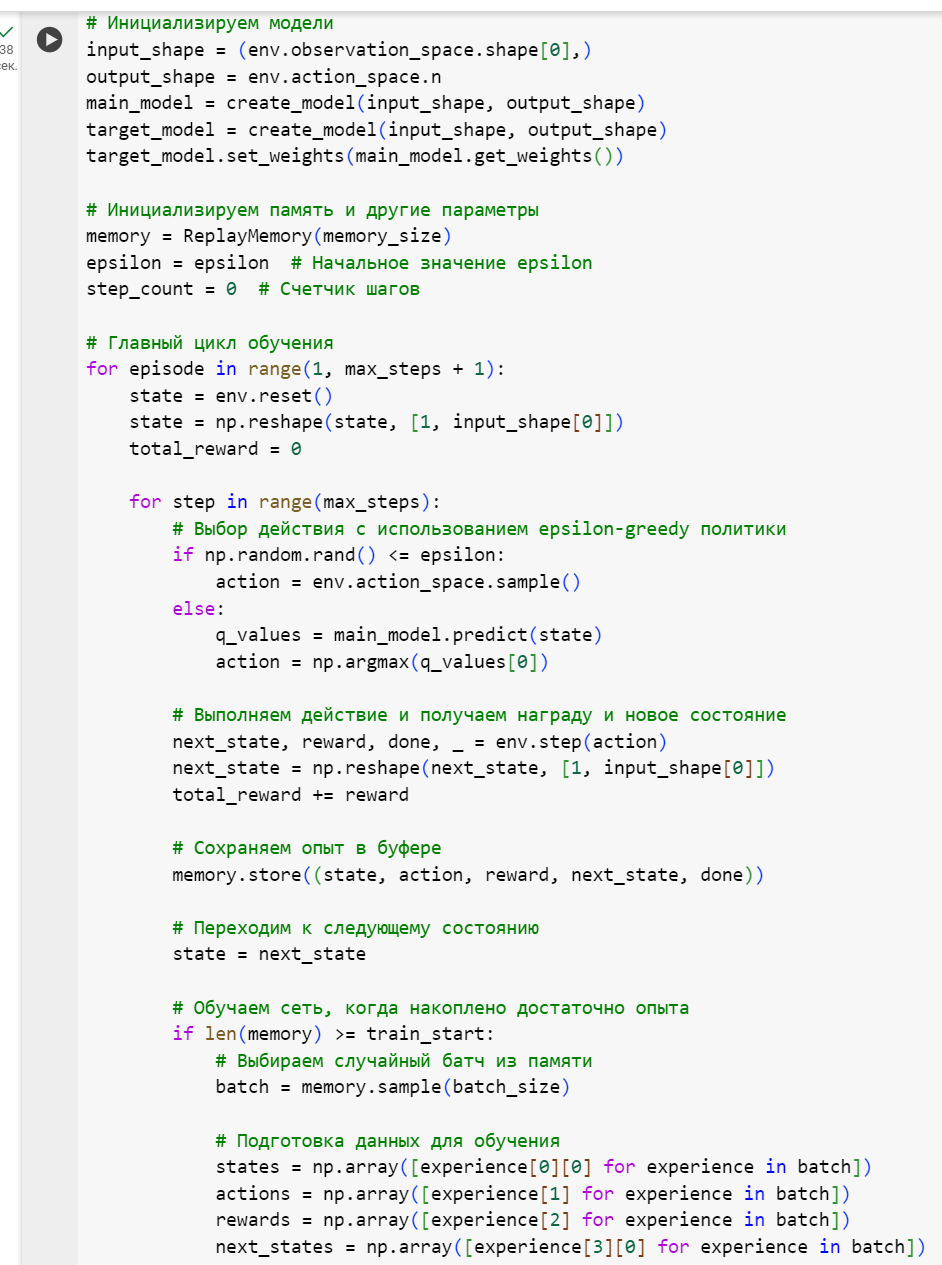
6. Завершение:

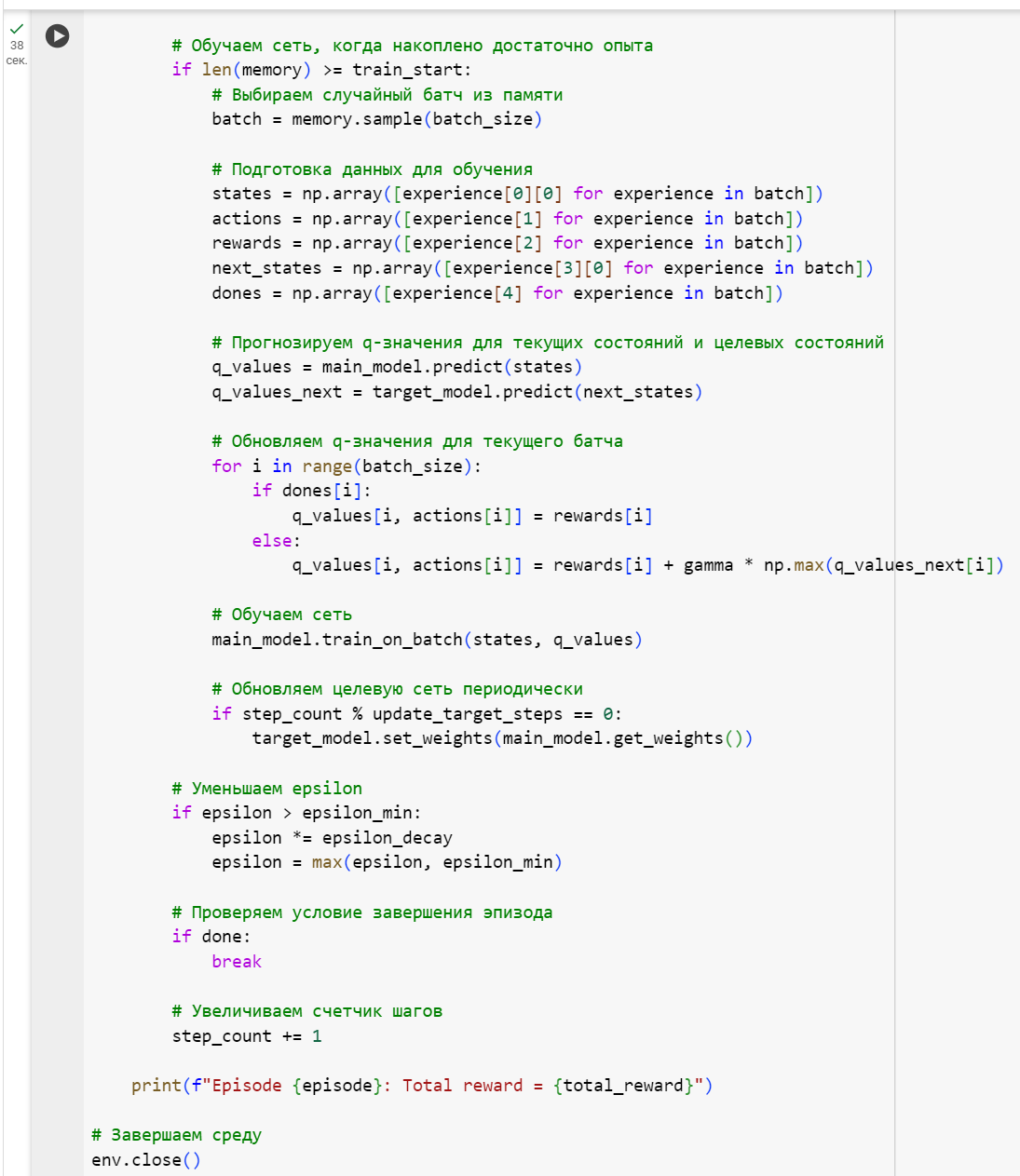
- После завершения всех эпизодов (достижения `max\_steps`) обучение завершается, и среда (`env`) закрывается.

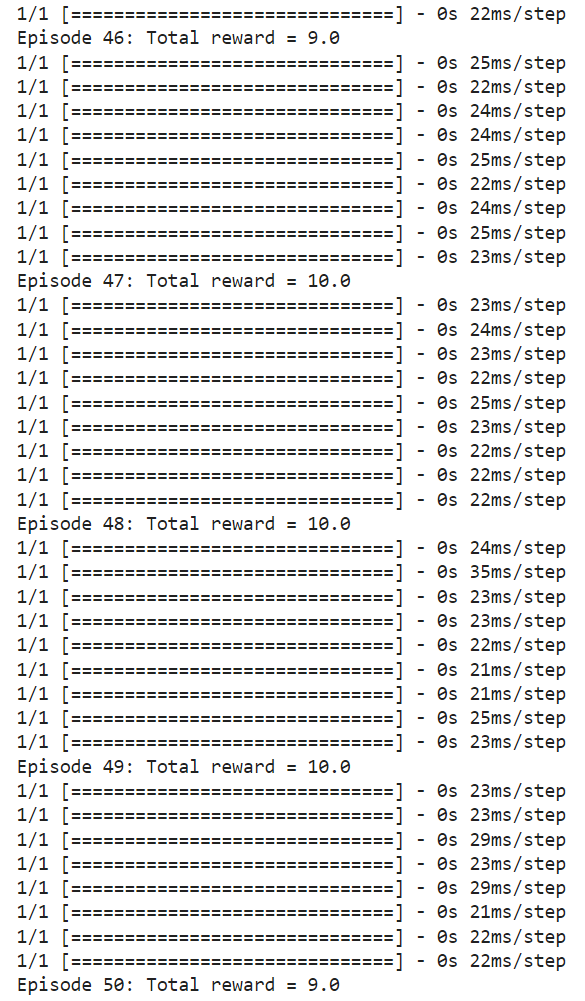
Код демонстрирует типичный процесс обучения DQN в простой среде `CartPole-v1`. По мере обучения агент становится лучше в принятии решений, направленных на максимизацию наград в среде.

**Текст программы и экранные формы с примерами выполнения программы**

****

****

****

****

**Вывод**

На основе полученных результатов можем сделать вывод о том, что алгоритм итерации политики в данном коде реализует метод для обучения агента в среде с подкреплением. Это алгоритм, который использует сочетание оценок текущей политики и ее улучшений для нахождения оптимальной политики, которая максимизирует суммарную награду за время работы агента в среде.