ОБУЧЕНИЕ АГЕНТОВ В ВИРТУАЛЬНОЙ СРЕДЕ KUKADIVERSOBJECTENV

Залогин Н.Е.
Национальный исследовательский Томский политехнический университет, ИШИТР
e-mail: nez4@tpu.ru

Аннотация

В исследовании сравниваются алгоритмы DQN и PPO, а также 3 модификации PPOv1, PPOv2, PPOv3 в среде KukaDiverseObjectEnv с использованием физического движка PyBullet. Результаты оцениваются по скорости обучения, стабильности и успешности решения задачи. Результаты исследования способствуют оптимизации алгоритмов обучения с подкреплением в сложных средах робототехники.

Ключевые слова: *обучение с подкреплением,* робот манипулятор*, pybullet, dqn, ppo.*

Введение

В последние десятилетия обучение с подкреплением (Reinforcement Learning, RL) привлекло значительное внимание исследователей и применимо к широкому спектру областей, включая робототехнику, игровую индустрию и автономную навигацию. Это обусловлено способностью алгоритмов RL эффективно принимать решения и преуспевать в сложных и динамичных средах на основе накопленного опыта. Так в последние годы исследователи из Google Research [1] опубликовали несколько работ с использованием роботов манипуляторов [2, 3], где достигли отличных результатов для крайне сложных и разнообразных задач. Однако, несмотря на прогресс в этой области, все еще существуют вызовы, связанные с некоторыми ограничениями и высокой вычислительной сложностью, в сложных средах.

Целью данной работы является реализация и сравнение алгоритмов Deep Q-Networks (DQN) [4] и Proximal Policy Optimization (PPO) [5], а также трех модификаций PPO - PPOv1, PPOv2, PPOv3. Алгоритмы реализованы и протестированы с использованием среды KukaDiverseObjectEnv [6] на физическом движке PyBullet [7]. Эти алгоритмы широко известны и демонстрируют некоторую эффективность в различных областях, но все еще требуют дальнейшего исследования и сравнения для повышения их производительности и применимости в сложных средах робототехники.

Также будет проведен анализ результатов работы каждого алгоритма на основе метрик, таких как скорость обучения, количество взаимодействий со средой и процент успешного выполнения задачи. Это позволит сравнить эффективность и применимость каждого алгоритма в среде KukaDiverseObjectEnv. Полученные результаты будут полезны для оптимизации алгоритмов обучения с подкреплением в сложных средах робототехники, а также для принятия решений о выборе наиболее подходящего алгоритма в конкретной задаче.

Описание алгоритма

KukaDiverseObjectEnv – среда симуляции на платформе PyBullet, разработанная для обучения агентов манипулированию объектами. Агенту необходимо принимать решение между 3 действиями – по одному действию для перемещения по каждой из осей x и y, а также одно действие для бездействия. При каждом шаге в среде манипулятор автоматически опускается. Таким образом агенту необходимо захватить и поднять любой объект из корзины. В данной задаче награда является бинарной, и выдается, только если один из объектов находится выше заданной высоты к концу эпизода. Входными данными, предоставляемыми агенту, являются трехканальные изображения состояния среды размером (48, 48, 3), пример такого изображения приведен на рис. 1. Благодаря редкой двоичной награде, графическим входным данным и сложности задачи в целом, среда становится достаточно интересной для исследования.

Для обучения агентов использованы алгоритмы DQN, PPO и три модификации примененного алгоритма PPO.

Поскольку входными данными является изображение с камеры, все реализованные алгоритмы используют сверточную (convolutional) нейронную сеть для его обработки и получения необходимых, для принятия решений, признаков.

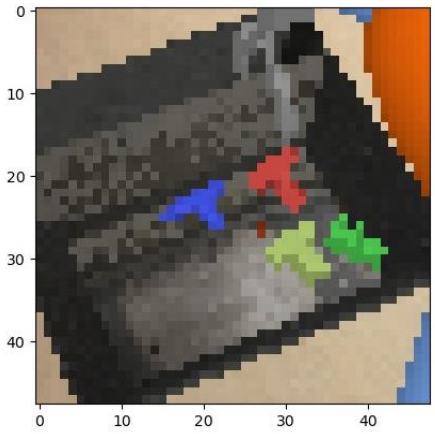

Рис. 1. Пример изображения с камеры робота

Алгоритм DQN является off-policy алгоритмом и использует нейронную сеть для приближения функции Q, которая оценивает ожидаемый суммарное дисконтированное вознаграждение агента для каждой пары состояние-действие в среде. Для тренировки алгоритма DQN будет использоваться воспроизведение опыта (Experience Replay). Оно сохраняет последовательности переходов $e_t = (S_t,A_t,R_t,S_{t+1})$ (где e – эпизод обучения, S – множество состояний среды, A – множество действий, R – множество наград), наблюдаемых агентом в память воспроизведения $D_t = \{e_1,\ldots,e_t\}$, что позволяет повторно использовать эти данные позже. Путем случайной выборки из памяти достигается декорреляция переходов, из которых формируется пакет данных (Batch). Это значительно стабилизирует и улучшает процедуру обучения алгоритма DQN. Необходимо отметить, что входные данные в данном алгоритме представлены в оттенках серого, что значительно упрощает обучение агента.

Алгоритм PPO является алгоритмом, основанным на политике (on-policy), и использует actor-critic архитектуру. PPO применяет две ключевые идеи для обеспечения стабильности и эффективности обучения. Первая идея — это обновление политики на основе отношения вероятностей между старой и новой политиками, чтобы ограничить ее изменение. Отношение политик вычисляется по формуле (1) [8]. Это означает, что ее обновление происходит малыми шагами и избегает значительных изменений, которые могут приводить к нестабильности обучения. Вторая идея – это использование значений преимуществ (advantages) для выравнивания обновлений политики, в противовес абсолютным вероятностям. Это способствует более сбалансированному и стабильному обучению.

$$r(\theta) = \frac{\pi_\theta(a \mid s)}{\pi_{\theta_{old}}(a \mid s)}, \tag{1}$$

где π – политика, θ - параметры политики, a – действие в среде, s – состояние среды, π(a|s) определяет вероятность выбора действия a в состоянии s.

Без ограничения на расстояния между новой и старой политиками, максимизация целевой функции привела бы к нестабильности с чрезвычайно большими обновлениями параметров и большими коэффициентами политики. PPO накладывает ограничение, заставляя оставаться в пределах небольшого интервала. Вычисляется целевая функция с этими изменениями по формуле (2) [8].

$$J^{CLIP}(\theta) = \mathrm{E}[\min(r(\theta)\hat{A}_{\theta_{old}}(s,a), clip(r(\theta),1-\varepsilon,1+\varepsilon)\hat{A}_{\theta_{old}}(s,a)], \tag{2}$$

где A(s,a) – функция преимуществ, ε – гиперпараметр.

В дополнение к ограниченному вознаграждению, целевая функция дополняется членом ошибки при оценке значения и энтропией, чтобы стимулировать достаточное исследование среды. В итоге целевая функция будет вычисляться по формуле (3) [8].

$$J^{CLIP}(\theta) = \mathrm{E}[J^{CLIP}(\theta) - c_1(V_0(s) - V_{target})^2 + c_2 H(s,\pi_\theta(.))] \tag{3}$$

где c_1 и c_2 – константные гиперпараметры, V(s) - ожидаемый возврат состояния s.

Топология сети в алгоритме PPO представлена следующим образом:

1. Входной слой: изображение с 3 каналами;
2. Сверточные слои:
   2.1. 16 фильтров, размер ядра (8x8), шаг (4x4);
   2.2. 32 фильтра, размер ядра (4x4), шаг (2x2);
   2.3. 32 фильтра, размер ядра (3x3), шаг (1x1).
3. Полносвязные слои:
   3.1. Общий слой: (128, 64) нейронов;
   3.2. Critic: (64, 1) нейронов;
   3.3. Actor: (64, 3) нейронов.

Алгоритм PPOv1 представляет собой первую модификацию PPO, специально адаптированную для взаимодействия со средой KukaDiverseObjectEnv. В данной модификации, помимо структурных изменений в обучающем цикле, произведены изменения в пространствах наблюдений и действий. Входные данные представлены изображениями размером (84, 84, 3), а количество доступных действий увеличено до пяти: по два действия для перемещения по каждой из осей x и y, а также одно действие для бездействия. Изменения также затронули архитектуру нейронной сети. Дополнительные скрытые слои были добавлены для сетей actor-critic. Кроме того, в сверточных слоях изменены размеры ядра и шаг, а также применена пакетная нормализация (BatchNorm2d). В качестве метода инициализации весов нейронной сети использован метод равномерного распределения Ксавье (xavier-uniform). Данные изменения должны стабилизировать и ускорить обучение агента.

Топология сети в алгоритме PPOv1 после изменения представлена следующим образом:

1. Входной слой: изображение с 3 каналами;
2. Сверточные слои:
   2.1. 16 фильтров, размер ядра (5x5), шаг (2x2);
   2.2. Пакетная нормализация: 16 нейрона;
   2.3. 32 фильтра, размер ядра (5x5), шаг (2x2);
   2.4. Пакетная нормализация: 32 нейрона;
   2.5. 32 фильтра, размер ядра (5x5), шаг (2x2);
   2.6. Пакетная нормализация: 32 нейрона.
3. Полносвязные слои:
   3.1. Общий слой: (128, 64) нейронов;
   3.2. Общий слой: (64, 64) нейронов;
   3.3. Critic: (64, 1) нейронов;
   3.4. Actor: (64, 5) нейронов.

Алгоритм PPOv2, являющийся второй модификацией PPO, вносит дополнительные изменения к PPOv1, включая синхронное параллельное выполнение действий в среде. Этот подход существенно ускоряет процесс сбора траекторий действий в среде и, следовательно, улучшает производительность обучения агента в целом. Реализация параллельных сред выполнена с использованием библиотеки «multiprocessing», при этом число параллельных сред составляет 20, что соответствует количеству потоков процессора на рабочей станции.

Алгоритм PPOv3 представляет собой третью модификацию PPO, продолжающую развитие PPOv2. В этой версии алгоритма были оптимизированы некоторые гиперпараметры благодаря использованию модульной системы обучающего цикла. В частности, главными изменениями стали уменьшение размера пакета данных и увеличение скорости обучения (Learning rate), что позволило сократить длительной эпизодов обучения и ускорить процесс обновления весов модели, что также ускорило обучение. Важным изменением является увеличение глубины слоев с большим количеством нейронов для сверточные сети, включая общие слои (shared layers) и слои actor-critic.

Топология сети в алгоритме PPOv3 после изменения представлена следующим образом:

1. Входной слой: изображение с 3 каналами;
2. Сверточные слои:
   2.1. 32 фильтров, размер ядра (5x5), шаг (2x2);
   2.2. Пакетная нормализация: 32 нейрона;
   2.3. 64 фильтра, размер ядра (5x5), шаг (2x2);
   2.4. Пакетная нормализация: 64 нейрона;
   2.5. 64 фильтра, размер ядра (5x5), шаг (2x2);

2.6. Пакетная нормализация: 64 нейрона.

3. Полносвязные слои:
   3.1. Общий слой: (576, 512) нейронов;
   3.2. Общий слой: (512, 256) нейронов;
   3.3. Общий слой: (256, 128) нейронов;
   3.4. Общий слой: (128, 64) нейронов;
   3.5. Скрытый слой Critic: (64, 512) нейронов;
   3.6. Critic: (512, 1) нейронов;
   3.7. Скрытый слой Actor: (64, 512) нейронов;
   3.8. Actor: (512, 5) нейронов.

Для тестирования и оценки алгоритмов обучения агентов был установлен критерий ранней остановки обучения – достижение средней награды в 50 за 100 эпизодов, что эквивалентно 50 % точности.

Для оценки точности обученных агентов использовалась та же среда в режиме тестирования, это позволило проверить их способность достигать соответствующих результатов при отличающихся условиях.

Результаты

По ходу обучения агентов были проведены измерения средней награды за 100 эпизодов, результаты которых представлены на рис. 2, 3.

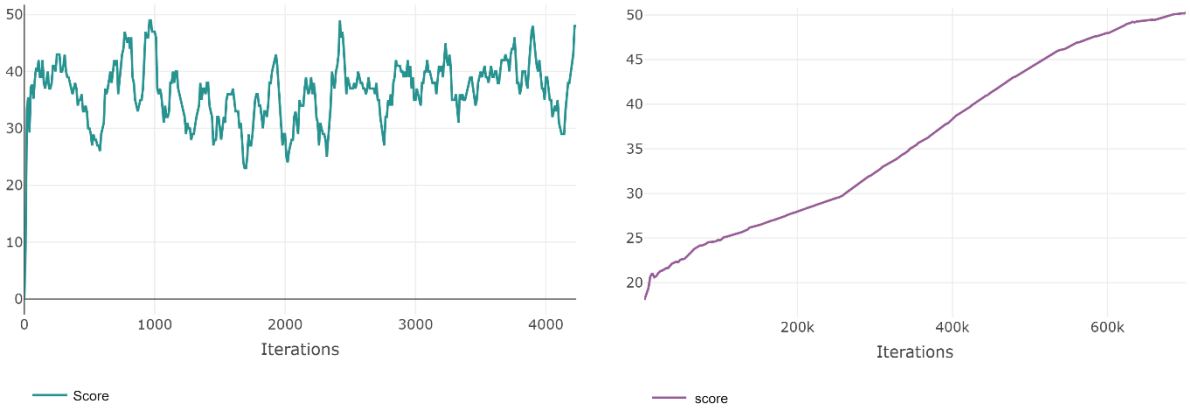

Рис. 2. Средняя награда агента DQN (а), Средняя награда агента PPO (б)

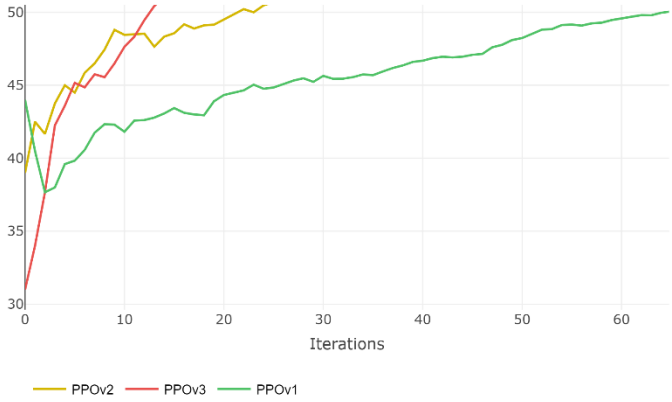

Рис. 3. Средняя награда агентов PPOv1, PPOv2, PPOv3

В дополнение к графикам средней награды, были зафиксированы показатели конечной средней награды, количество шагов обучения и время обучения, приведенные в таблице 1.

Таблица 1

Сравнение показателей обученных агентов

Агент	Средняя награда	Шаги обучения	Время обучения
DQN	51,000	4239	2:06:54,878
PPO	50,202	696320	5:12:04,500
PPOv1	50,690	26 (26624)	1:52:07,842
PPOv2	50,061	65 (66560)	0:35:12,836
PPOv3	50,067	13 (13312)	0:09:34,629

Из результатов, представленных на рис. 2, 3 и таблице 1, можно сделать вывод о том, что алгоритмы DQN и PPO достигают поставленной цели, но для этого требуется слишком продолжительное время. Модификация PPOv1 смогла сократить время обучения агента PPO до эквивалентного с агентом DQN значения. Параллелизация среды в PPOv2 значительно ускорила процесс обучения агента. Однако, это привело к увеличению количества взаимодействий со средой и соответственно итераций обучения. Модификация PPOv3 успешно решает эту проблему, позволяя сильнее сократить время обучения агента.

Для тестирования полученных агентов использовалось по 1000 эпизодов среды в тестовом режиме. Результаты тестирования приведены в таблице 2.

Таблица 2

Тестирование обученных агентов

Агент	DQN	PPO	PPOv1	PPOv2	PPOv3
True episode	47,8 %	49,7 %	50,2 %	49,1 %	52,4 %
False episode	52,2 %	50,3 %	49,8 %	50,9 %	47,6 %

Заключение

Стандартные DQN и PPO алгоритмы показали хорошие результаты и продемонстрировали способность решать поставленную задачу, но они имеют значительные проблемы с вычислительной сложностью. За счет изменения нейросети и дополнительных оптимизаций в PPOv1, время обучения для рассматриваемой задачи было сильно сокращено. Благодаря параллелизации сред в PPOv2, удалось еще сильнее сократить время обучения, несмотря на увеличение количества итераций обучения. Модификация PPOv3 устраняет этот недостаток, при этом также сокращая время обучения. Все реализованные агенты в том числе DQN, PPO и модификации PPOv1, PPOv2, PPOv3, успешно решают поставленную задачу, достигая точности в районе 50 % за 1000 тестовых эпизодов.

В будущем предполагается дальнейшее улучшение алгоритма PPO и общего процесса обучения для достижения лучших результатов. Рассмотрение различных дополнительных стратегий, может расширить возможности и повысить производительность агентов. А замена среды обучения на уже существующие варианты или создание собственной среды может открыть новые направления исследований в области обучения с подкреплением.

Список использованных источников

1. Google Research – Текст: электронный // Google. – 2024. – URL: https://research.google/ (дата обращения: 29.03.2024).
2. Zeng A. et al. Tossingbot: Learning to throw arbitrary objects with residual physics // IEEE Transactions on Robotics. – 2020. – Т. 36. – №. 4. – С. 1307-1319.
3. Zeng A. et al. Transporter networks: Rearranging the visual world for robotic manipulation //Conference on Robot Learning. – PMLR, 2021. – С. 726-747.
4. Mnih V. et al. Human-level control through deep reinforcement learning // Nature. – 2015. – Т. 518. – №7540. – С. 529-533.
5. Schulman J. et al. Proximal policy optimization algorithms //arXiv preprint arXiv:1707.06347. – 2017.

6. bulletphysics/bullet3 bullet/kuka_diverse_object_gym_env.py – Текст: электронный // bulletphysics – 2020. – URL: https://github.com/bulletphysics/bullet3/blob/master/examples/pybullet/gym/pybullet_envs/bullet/kuka_diverse_object_gym_env.py (дата обращения: 29.03.2024).
7. PyBullet – Текст: электронный // PyBullet – 2024. – URL: https://pybullet.org/wordpress/ (дата обращения: 29.03.2024).
8. Lilian Weng Policy Gradient Algorithms – Текст: электронный // Lil'Log – 2018. – URL: https://lilianweng.github.io/posts/2018-04-08-policy-gradient/#ppo (дата обращения: 29.03.2024).