ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук

Образовательная программа «Прикладная математика и информатика»

УДК 004.853

СОГЛАСОВАНО

Научный руководитель

Доцент факультета компьютерных наук базовой кафедры «Интеллектуальные технологии системного анализа и управления» ФИЦ «Информатика и управление» РАН

_____ А. И. Панов

«___» _____ 2019 г.

УТВЕРЖДАЮ

Академический руководитель образовательной программы «Прикладная математика и информатика», к. ф.-м. н.

_____ А.С. Конушин

«___» _____ 2019 г.

Отчет

к исследовательскому проекту

на тему: Обучение с подкреплением для манипулятора

(заключительный)

Выполнил

студент группы БПМИ 176

ОП ПМИ

Шабалина А. В.

_____

Подпись,

Дата

Москва 2019

Реферат

Отчет 9 с., 1 кн., 7 рис., 11 источн.

Основной целью этого исследовательского проекта было изучение Policy Gradient методов и реализация PPO(Proximal Policy Optimization) - алгоритма. Для этого потребовалось понять основы обучения с подкреплением: как работают алгоритмы из этого раздела машинного обучения, что такое среда и агент и их взаимосвязь. Для лучшего осознания этой темы сначала были реализованы более простые алгоритмы(SARSA и Q-learning). Также, во время выполнения работы, были реализованы такие алгоритмы, как DQN(Deep Q-Network) и DDPG(Deep Deterministic Policy Gradients), чтобы понять как работать с нейронными сетями и различными состояниями(states) среды.

Ключевые слова: обучение с подкреплением, алгоритм, стратегия, реализация алгоритмов, нейронные сети, policy gradient методы, TD-методы, DQN, DDPG, PPO-алгоритм.

Введение

Во время выполнения исследовательской работы были рассмотрены различные алгоритмы по теме Обучение с подкреплением (Reinforcement learning). Обучение с подкреплением - раздел машинного обучения, который основан на взаимодействии агента со средой. Агент имеет некоторое количество состояний, в которых он может находиться, и множество действий, которые он может совершить, находясь в каком-то состоянии. Также у агента есть функция вознаграждения. Он получает вознаграждение за каждое совершенное действие (размер вознаграждения зависит от действия и состояния, в котором находится агент). Главная цель агента - максимизация функции (суммы вознаграждений, который агент получит за весь эпизод). Совершая различные действия, агент таким образом взаимодействует со средой и обучается. Обучение состоит в нахождении оптимальной стратегии, следуя которой агент получает наибольшее вознаграждение.

Для реализации основного в проекте алгоритма, требовалось изучить более легкие алгоритмы Reinforcement learning. Поэтому, сначала, я реализовала SARSA и Q-learning алгоритмы. Они относятся к TD (temporal – difference) – prediction методам.

Следующим шагом был DQN-алгоритм и DDPG - алгоритм. Эти алгоритмы использовались мною, как вспомогательные, для реализации PPO-алгоритма. С их помощью стало понятно, как использовать нейронные сети, Replay Memory Buffer и т.п.

Главной целью проекта является изучение Policy Gradient методов, в которых стратегия улучшается благодаря корректировке вознаграждений через градиентный спуск, и реализация одного из этих методов - PPO (proximal policy optimization).

Для реализации алгоритмов использовалась библиотека Python OpenAI Gym[1]. Также использовалась библиотека Python TensorFlow[2] для построения нейронных сетей, т.к. обычные Q-таблицы не подходят для реализации более сложных алгоритмов из-за недостаточной эффективности.

Обзор источников

В ходе исследовательской работы были использованы разные источники информации. Основным источником является книга Reinforcement Learning(авторы: R. S. Sutton, A. G. Barto)[3]. В третьей и четвертой главах изложена основная информация, которую необходимо изучить для дальнейшей работы по теме Reinforcement Learning(в чем состоит задача обучения с подкреплением, что такое Марковский свойство и стратегия и т.п.). В шестой главе рассказано про обучение на основе временных различий и TD - методы. Информация про такие алгоритмы, как SARSA и Q-learning, особенно понадобилась мне в дальнейшем. Оба этих алгоритма основаны на динамическом программировании и используют Q-таблицу(таблица, в которой хранятся наибольшие вознаграждения для определенного состояния и действия). Но Q-learning является off-policy алгоритмом(совершает действие не ориентируясь на стратегию, которую улучшает), а SARSA - on-policy алгоритмом(совершает следующее действие соответственно текущей стратегии).

Также я использовала различные электронные источники: лекции из курса Deep RL Bootcamp[4], примеры программного кода и комментарии из курса Yandex Data

School[5], статьи, в которых описана архитектура алгоритмов(DQN[6], DDPG[7], PPO[8]). Вспомогательным источником была документация библиотеки Python OpenAI Gym[1](описание сред Taxi-v2[9], FrozenLake-v0[10], FetchPickAndPlace-v0, BreakoutDeterministic-v0, Pendulum-v0[11])

Основная часть отчета

Для начала мною изучены 3, 4 и 6 главы R. S. Sutton, A. G. Barto Reinforcement Learning: An Introduction[3]. Это необходимо для дальнейшей реализации алгоритмов. мною были изучены основные понятия обучения с подкреплением: взаимосвязь агента со средой, функции вознаграждения, стратегии, Марковские процессы принятия решения, on-policy и off-policy методы и т.д. Были рассмотрены различные TD (temporal – difference) – prediction методы. Эти алгоритмы сочетают в себе самонастраивание (обновление оценки за счет уже полученной оценки, не дожидаясь конца эпизода) и обучение только за счет полученного опыта. Некоторые из них (SARSA, Q-learning) я реализовала на Python. Для этого я использовала библиотека Python OpenAI Gym[1]. Алгоритмы были реализованы на тестовых окружениях Taxi-v2[9] и FrozenLake-v0[10].

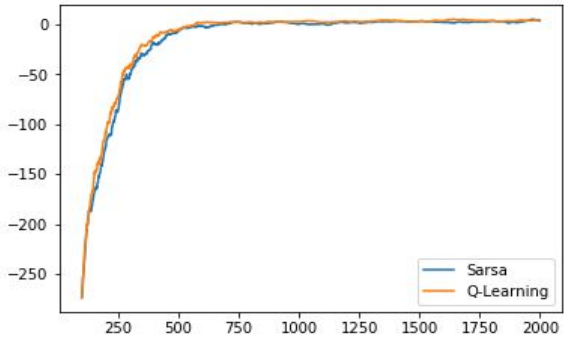

Рисунок 1 - График сходимости функции ценности для алгоритмов SARSA и Q-learning

Также, после просмотра лекций[4] и материалов из курса Yandex Data School[5], я реализовала DQN(Deep-Q-Network) - алгоритм.

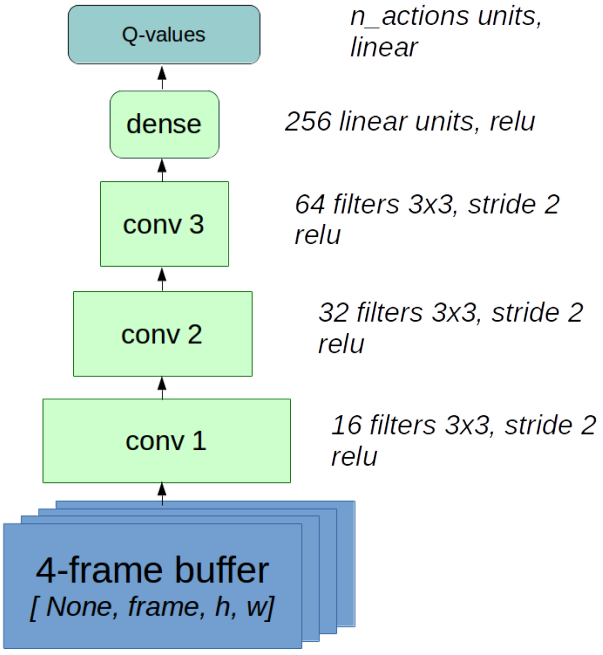

Рисунок 2 - Архитектура DQN-алгоритма

Для этого была использована среда BreakoutDeterministic-v0.

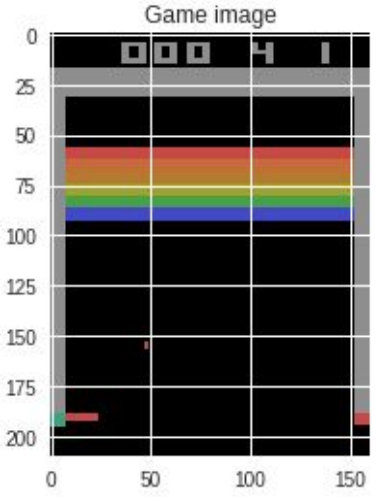

Рисунок 3 - Тестовое окружение BreakoutDeterministic-v0

Для этого алгоритма также требуется использование нейронных сетей(при помощи библиотеки TensorFlow[2]) из-за недостаточной эффективности обычных Q-таблиц. Также, в этом алгоритме был использован Replay Memory Buffer, который собирает

информацию о различных состояниях среды и подает на вход алгоритму в случайном порядке для лучшего обучения.

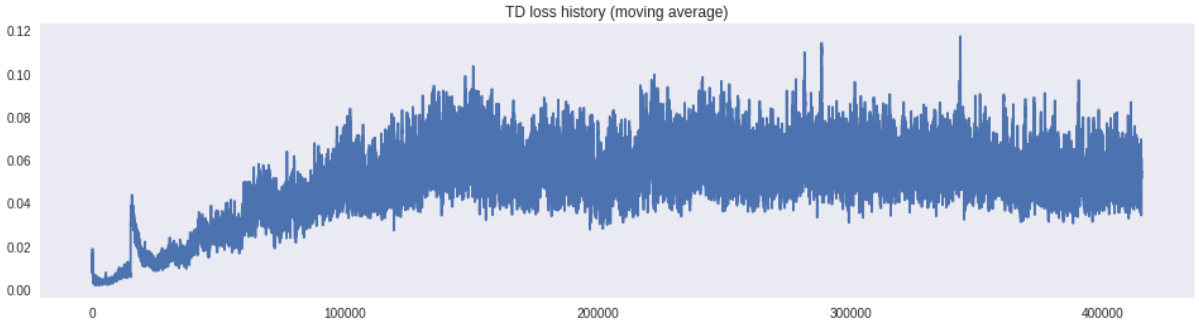

Рисунок 4 - График средней квадратичной ошибки для DQN

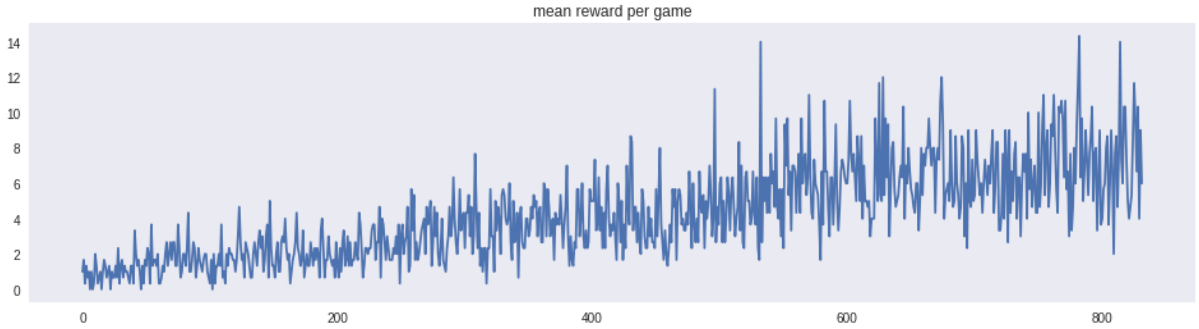

Рисунок 5 - График среднего вознаграждения для DQN

Следующим шагом была реализация DDPG - алгоритма (для лучшего понимания множества непрерывных состояний). Для этого алгоритма была выбрана среда с непрерывными состояниями (Pendulum-v0[11]). Этот алгоритм, в отличие от упоминавшегося ранее DQN - алгоритма, основан на policy gradient методе. Он оптимизирует стратегию (policy) методом градиентного подъема. В DDPG используется модель Actor-Critic, где Critic-модель сообщает Actor-модели, как она должна измениться, основываясь на значениях Value-функции.

Конечной целью этой исследовательской работы была реализация PPO-алгоритма (proximal policy optimization). Работа этого алгоритма, так же как и DDPG, основана на policy gradient методе. PPO - это, по своей сути, усовершенствованная версия TRPO-алгоритма. В нем мы также пытаемся менять нашу стратегию ограниченными шагами. Для этого используется расстояние Кульбака — Лейблера (KL-divergence). Таким образом мы улучшаем обучение, так как большое

изменение стратегии иногда может значительно ухудшить результаты. В этом алгоритме также используется Actor-Critic модель.

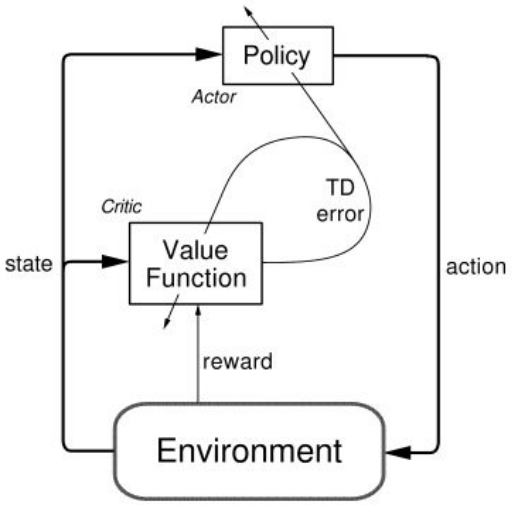

Рисунок 6 - Архитектура Actor-Critic модели

Реализация алгоритма была поделена на два этапа: реализация на простой среде (с малым количеством состояний и действий, например Pendulum-v0[11]) и реализация на более сложной среде (FetchPickAndPlace-v0). На первом этапе мне удалось добиться среднего вознаграждения чуть меньше 200 (в среде Pendulum-v0 лучшим результатом

Рисунок 7 - Тестовое окружение Pendulum-v0

считается вознаграждение равное 0). Для второго этапа требуется библиотека MuJoCo, в которой находятся более сложные тестовые окружения (как FetchPickAndPlace-v0). Пока что мне не удалось установить эту библиотеку по техническим причинам, но я

планирую в ближайшее время это исправить и запустить алгоритм и на FetchPickAndPlace-v0.

Заключение

В течение этого исследовательского проекта я постаралась изучить Reinforcement Learning, узнала основные термины и понятия из этого раздела и разобралась в архитектуре таких алгоритмов, как SARSA, Q-learning, DQN, DDPG и PPO. В дальнейшем я раздумываю над подачей заявки с доработанным вариантом проекта в Летнюю школу Российской ассоциации искусственного интеллекта, чтобы еще глубже изучить эту тему.

Список использованных источников

1. Gym [Электронный ресурс] Toolkit for developing and comparing reinforcement learning algorithms / Режим доступа: https://gym.openai.com/docs/, свободный. (дата обращения: 26.11.2018)
2. TensorFlow [Электронный ресурс] Open source machine learning framework / Режим доступа: https://www.tensorflow.org/, свободный. (дата обращения: 12.12.2018)
3. R. S. Sutton, A. G. Barto Reinforcement Learning: An Introduction / R. S. Sutton, A. G. Barto. – 2-е издание. - The MIT Press Cambridge, Massachusetts London, England. – 548 с. – ISBN: 978-0-262-19398-6
4. Deep RL Bootcamp [Электронный ресурс] / Режим доступа: https://sites.google.com/view/deep-rl-bootcamp/lectures, свободный. (дата обращения: 15.03.2019)
5. Practical_RL [Электронный ресурс] Lectures about neural nets and deep learning / Режим доступа:https://github.com/yandexdataschool/Practical_RL/tree/master/week4_%5Brecap%5D_deep_learning, свободный. (дата обращения: 10.12.2018)
6. Playing Atari with Deep Reinforcement Learning / Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, Martin

Riedmiller // [Электронный ресурс] - Режим доступа: https://arxiv.org/pdf/1312.5602.pdf, свободный. (дата обращения: 02.02.2019)

7. Continuous control with deep reinforcement learning / Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, Daan Wierstra // [Электронный ресурс] - Режим доступа:https://arxiv.org/pdf/1509.02971.pdf, свободный. (дата обращения: 01.03.2019)
8. Proximal Policy Optimization Algorithms / John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, Oleg Klimov // [Электронный ресурс] - Режим доступа: https://arxiv.org/pdf/1707.06347.pdf, свободный. (дата обращения: 12.03.2019)
9. Taxi-v2 [Электронный ресурс] GitHub / Режим доступа: https://gym.openai.com/envs/Taxi-v2/, свободный. (дата обращения: 3.12.2018)
10. FrozenLake-v0 [Электронный ресурс] GitHub / Режим доступа: https://github.com/openai/gym/blob/master/gym/envs/toy_text/frozen_lake.py, свободный. (дата обращения: 3.12.2018)
11. Pendulum-v0 [Электронный ресурс] GitHub / Режим доступа: https://github.com/openai/gym/wiki/Pendulum-v0, свободный. (дата обращения: 22.03.2019)