**Безопасность цифровых технологий, № 1(101), 2021, ISSN 2782-2230, страницы 7-168**

Статья 1

АВТОМАТИЧЕСКОЕ УПРАВЛЕНИЕ И ИДЕНТИФИКАЦИЯ (Раздел сборника)

7-25 (страницы статьи)

Евсеенко Алла Александровна, Романников Дмитрий Олегович

Новосибирский государственный технический университет (место работы авторов)

630073, Российская Федерация, г. Новосибирск, пр. Карла Маркса, 20 (адрес места работы)

**Евсеенко (индексы автора проставляются только те, которые у него есть, остальные удаляются)**

ORCID: [0000-0003-0254-2877](http://orcid.org/0000-0003-0254-2877)

Web of Science Researcher ID:

Scopus Author ID:

Google Scholar:

Researcher ID:

researchgate.net

ArXiv Author ID:

Author ID (РИНЦ):

SPIN-код (РИНЦ):

Istina Researcher ID (IRID):

[vnattali@mail.ru](mailto:vnattali@mail.ru)

**Романников**

ORCID:

Web of Science Researcher ID:

Scopus Author ID:

Google Scholar:

Researcher ID:

researchgate.net

ArXiv Author ID:

Author ID (РИНЦ): 794925

SPIN-код (РИНЦ): 9203-2803

Istina Researcher ID (IRID):

[dmitry.romannikov@gmail.com](mailto:dmitry.romannikov@gmail.com)

**Применение алгоритмов Deep Q-learning и Double Deep Q-learning к задаче управления перевернутым маятником (название статьи, ниже - аннотация)**

На сегодняшний день в мире бурно развивается такой раздел науки, как «искусственный интеллект». Системы, построенные на основе методов искусственного интеллекта, обладают свойством выполнять функции, которые традиционно считаются прерогативой человека. Искусственный интеллект обладает широким спектром областей исследований. Одной из таких областей является машинное обучение. В данной статье рассматриваются алгоритмы одного из подходов машинного обучения – обучение с подкреплением (англ. reinforcement learning или RL), по которому осуществляются много исследований и разработок в течении последних семи лет. Разработки и исследования по данному подходу в основном осуществляются для решения задач в играх Atari 2600 или в других подобных. В данной статье обучение с подкреплением будет применяться к одному из динамических объектов – перевернутому маятнику. В качестве модели указанного объекта рассматривается модель перевернутого маятника на тележке, взятая из библиотеки Gym, в которой находятся много моделей, которые используются для тестирования и анализа алгоритмов обучения с подкреплением. В статье приводится реализация и исследование двух алгоритмов из данного подхода Deep Q-learning и Double Deep Q-learning. В качестве результата представлены графики обучения, тестирования и времени обучения для каждого алгоритма, на основе которых делается вывод, что желательно использовать алгоритм Double Deep Q-learning, потому что время обучения составляет приблизительно 2 минуты и осуществляет наилучшее управление моделью перевернутого маятника на тележке.

УДК

519.24

DOI: (проставляется издательством)

10.17212/2307-6879-2020-1-2-7-25

**Ключевые слова:**

нейронные сети, искусственный интеллект, модель перевернутого маятника, Python, Gym, PyTorch, Deep Q-learning (DQN), Double Deep Q-learning (DDQN), обучение с подкреплением (reinforcement learning или RL)

СПИСОК ЛИТЕРАТУРЫ

1. Playing Atari with deep reinforcement learning / V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller. – URL: https://arxiv.org/abs/1312.5602 (accessed: 03.07.2020).

2. Gym: официальный сайт проекта Gym. – URL: http://gym.openai.com/html (дата обращения: 03.07.2020).

3. Перевернутый маятник. – URL: http://www.100byte.ru/python/cartPole/cartPole.html (дата обращения: 03.07.2020).

4. CartPole-v0 // OpenAI Wiki. – URL: https://github.com/openai/gym/wiki/CartPole-v0 (accessed: 03.07.2020).

5. *Barto A.G., Sutton R.S., Anderson C.W.* Neuronlike adaptive elements that can solve difficult learning control problems // IEEE Transactions on Systems, Man, and Cybernetics. – 1983. – Vol. SMC-13, N 5. – P. 834–846. – URL: http://www.derongliu.org/adp/adp-cdrom/Barto1983.pdf (accessed: 03.07.2020).

6. *Hasselt H. van, Guez A., Silver D*. Deep reinforcement learning with Double Q-learning. – URL: https://arxiv.org/abs/1509.06461 (accessed: 03.07.2020).

7. Python: website. – URL: https://www.python.org/ (accessed: 03.07.2020).

8. PyTorch: website. – URL: https://pytorch.org/ (accessed: 03.07.2020).

9. *Нильсон Н.* Искусственный интеллект. – М.: Мир, 1973. – 273 с.

10. *Хайкин С.* Нейронные сети: полный курс: пер. с англ. – 2-е изд. – М.: Вильямс, 2006. – 1104 с.

11. *Kingma D.P., Ba J.L*. Adam: a method for stochastic optimization // International Conference on Learning Representations (ICLR 2015). – Ithaca, NY, 2015. – URL: https://arxiv.org/pdf/1412.6980.pdf (accessed: 03.07.2020).

12. *Sutton R.S., Barto A.G.* Introduction to reinforcement learning. – Cambridge: MIT Press, 1998.

13. Видеокарта MSI GeForce GTX 1050 Ti // MSI: web-сайт. – URL: <https://ru.msi.com/Graphics-card/support/GeForce-GTX-1050-Ti-GAMING-4G> (дата обращения: 03.07.2020).

14. CUDA Toolkit Documentation v11.0.171 // Developer Zone NVIDIA. – URL: https://docs.nvidia.com/cuda/ (accessed: 03.07.2020).

15. *Evseenko A.A.* Analysis of the applicability of artificial intelligence methods to solving problems of stabilization of dynamic systems / research adviser D.O. Romannikov, language adviser R.A. Chesnokova // Progress through innovations: proceedings 2019 VIII International academic and research conference of graduate and postgraduate students, March 28, 2019, Novosibirsk, Russia. – Novosibirsk, 2019. – P. 55–57. – ISBN 978-5-7782-3848-0.

16. *Романников Д.О.* Исследование работы нейронных сетей на примере задачи управления перевернутым маятником // Сборник научных трудов НГТУ. – 2018. – № 1 (91). – С. 95–103.

Дата поступления:

04.05.2020

AUTOMATIC CONTROL AND IDENTIFICATION (англоязычная версия)

Evseenko Alla A., Romannikov Dmitrii O.

Novosibirsk State Technical University

20, Karl Marx Prospekt, Novosibirsk, 630073, Russian Federation

**Evseenko**

ORCID: [0000-0003-0254-2877](http://orcid.org/0000-0003-0254-2877)

Web of Science Researcher ID:

Scopus Author ID:

Google Scholar:

Researcher ID:

researchgate.net

ArXiv Author ID:

Author ID (RSCI):

SPIN-code (RSCI):

Istina Researcher ID (IRID):

[vnattali@mail.ru](mailto:vnattali@mail.ru)

**Romannikov**

ORCID:

Web of Science Researcher ID:

Scopus Author ID:

Google Scholar:

Researcher ID:

researchgate.net

ArXiv Author ID:

Istina Researcher ID (IRID):

SPIN-code (RSCI): 9203-2803

Author ID (RSCI): 794925

dmitry.romannikov@gmail.com

**Application of Deep Q-learning and double Deep Q-learning algorithms to the task of control an inverted pendulum**

Today, such a branch of science as «artificial intelligence» is booming in the world. Systems built on the basis of artificial intelligence methods have the ability to perform functions that are traditionally considered the prerogative of man. Artificial intelligence has a wide range of research areas. One such area is machine learning. This article discusses the algorithms of one of the approaches of machine learning – reinforcement learning (RL), according to which a lot of research and development has been carried out over the past seven years. Development and research on this approach is mainly carried out to solve problems in Atari 2600 games or in other similar ones. In this article, reinforcement training will be applied to one of the dynamic objects – an inverted pendulum. As a model of this object, we consider a model of an inverted pendulum on a cart taken from the Gym library, which contains many models that are used to test and analyze reinforcement learning algorithms. The article describes the implementation and study of two algorithms from this approach, Deep Q-learning and Double Deep Q-learning. As a result, training, testing and training time graphs for each algorithm are presented, on the basis of which it is concluded that it is desirable to use the Double Deep Q-learning algorithm, because the training time is approximately 2 minutes and provides the best control for the model of an inverted pendulum on a cart.

**Keywords:**

neural networks, artificial intelligence, inverted pendulum model, Python, Gym, PyTorch, Deep Q-learning (DQN), Double Deep Q-learning (DDQN), reinforcement learning (RL)

**Текст статьи** (из статьи удаляются все рисунки, таблицы, формулы, размерности, буквенные обозначения; вместо них оставляют пропуски)

**Введение.**

Подход обучения с подкреплением начал бурно развиваться после выпуска статьи в 2013 год от компании DeepMind о глубоком обучении с подкреплением для игр Atari 2600 [1]. Для использования обучения с подкреплением не нужно формировать тестовую и обучающую выборку. Обучение с подкреплением (англ. Reinforcement learning или RL) – один из подходов машинного обучения в ходе которого агент обучается, взаимодействуя с некоторой средой. В качестве агента выступает нейронная сеть, которую обучают с помощью алгоритма обучения, а среда – модель объекта (объект) и ее поведение. В этой статье указанный подход машинного обучения используется для управления моделью перевернутого маятника на тележке. В качестве алгоритмов обучения с подкрепление реализуются и исследуются для данной модели алгоритмы Deep Q-learning и Double Deep Q-learning.

**1. Описание задачи.**

Перевернутый маятник (рис. 1) состоит из передвижной тележки (на рисунке изображена черным цветом с цифрой 1), на которой расположен сам маятник (вертикальный шест, на рисунке изображен светло-коричневым цветом с цифрой 2). Маятник прикреплен на шарнир (на рисунке изображен фиолетовым цветом с цифрой 3) и может осуществлять вращение на 360 градусов. Данный объект удерживается в перевернутом состоянии за счет изменения скорости тележки. Возможны два действия, оказываемые на тележку, при воздействии которых она движется горизонтально: вправо или влево. Основная задача заключается в разработке системы, которая удерживает шест перевернутого маятника в вертикальном положении. При использовании обучении с подкреплением для решения этой задачи необходимо для каждого состояния s объекта управления правильно выбирать действие a, оказываемое на тележку с маятником. Используется модель перевернутого маятника на тележке из библиотеки Gym – CartPole-v0 (рис. 1) [2].У данной модели определены следующие параметры [3,4]:

Состояние, которые описывается следующими величинами:

позиция тележки, которая принимает значения в диапазоне ;

скорость тележки;

угол отклонения шеста от вертикального положения, который принимает значение в диапазоне [-41.8°, 41.8°];

скорость изменения наклона угла шеста;

Действие, которое принимает следующие значения:

0 – приложить горизонтальную силу к тележке, направленную влево;

1 – приложить горизонтальную силу к тележке, направленную вправо;

Награда, которая на каждом шаге равна 1, включая и последний шаг;

Начальное состояние задается при помощи датчика равномерно распределенных значений в диапазоне ;

Случаи завершения эпизода:

угол отклонения шеста от вертикального положения вышел из диапазона

позиция тележки вышла из допустимого диапазона значений ;

длина эпизода превышает значение 200.

Задача для данной модели считается решенной в обучении с подкреплением, если средняя награда за 100 последовательных эпизодов обучения не меньше значения 195.

Рис. 1. Иллюстративное представление задачи перевернутого маятника

Модель перевернутого маятника из Gym реализована на основе статьи [5]. Она описывается следующими дифференциальными уравнениями без учета сил трения и сопротивления:

,

где: - угол отклонения шеста от вертикального положения в момент времени ; – скорость изменения наклона угла шеста в момент времени ; – позиция тележки в момент времени ; – скорость тележки в момент времени ; ; масса тележки; масса шеста; половина длины шеста; сила, приложенная к центру масс тележки в момент времени .

Управление перевернутым маятником в теории автоматического управления считается сложной задачей, так как сама система является неустойчивой, а переходный процесс передаточной функции является колебательным. Описав среду, в качестве которой выступает модель перевернутого маятника, необходимо описать агентов и алгоритмы обучения с подкреплением, которые нам позволят решить поставленную здесь задачу управления.

**2. Алгоритмы deep q-learning и double deep q-learning.**

Алгоритм Deep Q-learning, который реализуется и исследуется в данной статье, был реализован компанией DeepMind в 2013 году и представлен в статье [1]. Данный алгоритм входит в группу алгоритмов Q-learning, которые оперируют функцией качества, обозначающей наибольшую возможную награду, которую может получить агент, выполняя действие в состоянии. Deep Q-learning использует нейронную сеть для аппроксимации функции. Нейронная сеть получает состояние, после чего она возвращает значения для каждого действия в этом состоянии.

Одна из особенностей использования нейронных сетей в обучении с подкреплением – это отсутствие гарантий сходимости. Для улучшения сходимости используется подход известный как повторение опыта ( ). Данный подход подразумевает использование некоторого хранилища с определенным размером, который называется памятью воспроизведения ( ), где хранятся последние (размер памяти воспроизведения) опытов агента(алгоритма), представленные в виде ( ()), где:– состояние объекта в момент времени,– действие в момент времени,– награда, полученная за выполнение действия в момент времени, () – состояние объекта в момент времени. При обучении используется случайная выборка определенного размера из памяти воспроизведения и применяется обновление Q-learning. После воспроизведения опыта агент выбирает и выполняет действие в соответствии с -жадной политикой. В итоге на основе описанного алгоритма в статье [1] и применительно к модели перевернутого маятника из библиотеки Gym, получаем следующий алгоритм:

1. Инициализовать память воспроизведения размерностью ([])

2. Инициализовать случайными весам функцию (т.е. нейронную сеть)

3. Повторять для каждой игры

4. Инициализовать

5. Повторять для каждого шага

6. Выбрать по (-жадную)

7. Выполнить, найти

8. Занести в память воспроизведения []

9. Выбрать случайным образом из памяти воспроизведения коллекцию []

10.

11. Выполнить градиентный спуск на

12.

Преимущества алгоритма, представленного в статье [1]:

1) каждый шаг опыта потенциально используется во многих обновлениях весов нейронной сети, что позволяет повысить эффективность данных;

2) обучение непосредственно из последовательных выборок неэффективно из-за сильной корреляции между выборками; использование случайных выборок нарушает эти корреляции и, следовательно, уменьшает дисперсию в обновлениях.

Кроме алгоритма Deep Q-learning в данной статье реализуется и исследуется еще один алгоритм из группы Q-learning – Double Deep Q-learning, кото-рый также был реализован компанией DeepMind в 2015 году и представлен в статье [6]. Идея Double Q-learning заключается в том, что такие операции, как выбор действия и оценка действия, разделены на две нейронные сети такие, как основная (online network) и целевая (target network) нейронные сети. При этом архитектура нейронных сетей остается одинаковой, целевая нейронная сеть является копией основой, веса которой обновляются с определенной пе-риодичностью. В данной алгоритме предлагается оценить -жадную политику в соответствии с основной нейронной сетью, но используя целевую сеть, чтобы оценить ее значения. В итоге на основе описанного алгоритма в статье [6] и применительно к модели перевернутого маятника из библиотеки Gym, получаем следующий алгоритм:

1. Инициализовать память воспроизведения размерностью ([ ])

2. Инициализовать случайными весам функцию (т.е. основную нейронную сеть)

3. Инициализовать случайными весам функцию (т.е. целевую нейронную сеть)

4. Инициализировать шаг обновления функции

5. Повторять для каждой игры

6. Инициализовать

7. Повторять для каждого шага

8. Выбрать по ( -жадную)

9. Выполнить , найти

10. Занести в память воспроизведения [ ]

11. Выбрать случайным образом из памяти воспроизведения коллекцию [ ]

12.

13. Выполнить градиентный спуск на

14.

15. Обновить функцию в соответствии с установленным шагом

Алгоритм Double Deep Q-learning является некоторой модификацией алгоритма DeepQ-learning. Цель создания алгоритма Double Deep Q-learning, заключается в том, чтобы получить большую часть преимуществ от Q-learning, сохраняя при этом остальную часть алгоритма Deep Q-learning без изменений для правильного сравнения, и с минимальными вычислительными затратами.

Описав принцип работы алгоритмов Deep Q-learning и Double Deep Q-learning, перейдем к их реализации и исследованию.

**3. Реализация алгоритмов deep q-learning и double deep q-learning.**

Алгоритмы Deeep Q-learning и Double Deep Q-learning были реализованы на языке программирования Python c использованием фреймворка машинного обучения PyTorch, и модель перевернутого маятника на тележке (CartPole-v0) была взята из библиотеки Gym.

Для реализации потребовались следующие библиотеки Python [7]: – библиотека для работы с числами и математическими операциями;– библиотека для генерации случайных чисел;– библиотека для работы с многомерными массивами данных;– библиотека для визуализации данных и построения графиков; – библиотека для работы со временем.

Для реализации также потребовались следующие модули фреймворка PyTorch [8]: – модуль самого фреймворка PyTorch;– модуль, который отвечает за архитектуру нейронных сетей; – модуль, который отвечает за методы оптимизаций нейронных сетей; – модуль, который позволяет использовать возможности видеокарты, которая поддерживает архитектуру cuda.

Для работы с моделью перевернутого маятника (CartPole-v0) потребовались следующие инструменты из Gym [2]: – завершает текущий эпизод и начинает новый; возвращает начальное состояние; – отображает текущее состояние среды; – совершает указанное действие; с помощью него получаем новое состояние, награду, завершился ли эпизод; – количество действий, определенных в модели; – количество состояний у модели.

Разработанная нейронная сеть для двух алгоритмов представляет собой многослойную нейронную сеть, состоящую из трех слоев: 1-й слой состоит из 4 входных (по количеству состояний модели) и 128 выходных нейронов, 2-й слой состоит из 128 входных и выходных нейронов, а 3-й слой состоит из 128 входных и 2 выходных (по количеству действий, определенных в модели)нейронов (см. рис. 2) [9-11].

Рис. 2. Схема разработанной многослойной нейронной сети для 2-х алгоритмов Deep Q-learning и Double Deep Q-learning

Обучение вышеприведенной сети выполняется методом стохастической оптимизации (Adam) [12].

Реализация алгоритмов Deep Q-learning и Double Deep Q-learning на языке программирования Python c использованием фреймворка машинного обучения PyTorch осуществлялась с использование объектно-ориентированного подхода, через классы: class MNN – класс, в котором описана структура разработанной нейронной сети (см. рис. 2); class ReplayBuffer – класс, в котором реализован метод воспроизведения опыта (experience replay); class Tester – класс, который отвечает за процесс тестирования обученной нейронной сети на модели перевернутого маятника на тележке; class Configuration – класс, в котором хранятся параметры алгоритмов, обучения и нейронной сети; class DQNAgent и DDQNAgent – классы, в которых реализованы алгоритмы работы Deep Q-learning и Double Deep Q-learning, описанные в пункте 2 данной статьи; class Trainer – класс, в котором реализован процесс обучения нейронной сети в соответствии с алгоритмом обучения (Deep Q-learning или Double Deep Q-learning). Взаимосвязь классов и подробное их представление показано на рис. 3 – 4.

Рис. 3. Диаграмма классов реализации алгоритма Deep Q-learning и машинного обучения Reinforcement Learning

Рис. 4. Диаграмма классов реализации алгоритма Double Deep Q-learning и машинного обучения Reinforcement Learning

Реализация работы алгоритма Deep Q-learning, представленной в пункте 2, на языке программирования Python c использованием фреймворка PyTorch:

Реализация работы алгоритма Double Deep Q-learning, представленной в пункте 2, на языке программирования Python c использованием фреймворка PyTorch:

4. Исследование алгоритмов deep q-learning и double deep q-learning.

Исследование алгоритмов Deep Q-learning и Double Deep Q-learning, реализованных в пункте 4 данной статьи, проходило в 2 процесса: обучение и тестирование.

Процесс обучения представлял собой обучение нейронной сети по одному из 2-х реализованных алгоритмов с одинаковыми исходными параметрами. Параметры при использовании алгоритма Deep Q-learning:;, ,– для расчета -жадной политики;– количество кадров(шагов) обучения;– скорость обучения; – объем памяти воспроизведения; – размер случайной партии из памяти воспроизведения;. Параметры при использовании алгоритма Double Deep Q-learning такие же как у Deep Q-learning только добавляется такой параметр как– обновление целевой сети (target network). Для достижения решения поставленной задачи управления перевернутым маятником на тележке обучение завершается: если за последние 100 эпизодов обучения среднее значение наград превышает или равно 198 (win\_reward), и награда за последний эпизод обучения больше или равна 198. Данное значение завершения обучения было выбрано равным 198 на основе исследований, представленных в статье [1].

Процесс тестирования представлял собой применение обученной нейронной сети для управления перевернутым маятником. Тестирование проводилось до 200 временных шагов, которые указаны в параметрах завершения эпизода модели перевернутого маятника на тележке CartPole-v0 (см. пункт 2).

Исследование проводилось с использованием возможности видеокарты MSI GeForce GTX 1050 Ti [13], которая поддерживает архитектуру cuda [14]. Исследования алгоритмов Deep Q-learning и Double Deep Q-learning проводились по времени обучения, по количеству эпизодов обучения и как осуществляет управление моделью перевернутого маятника обученной нейронной сетью (см. рис. 5-9) [15].

Рис. 5. Время обучение нейронной сети алгоритмам Deep Q-learning и Double Deep Q-learning

Рис. 6. Процесс обучения при использовании алгоритма Double Deep Q-learning

Рис. 7. Процесс обучения при использовании алгоритма Deep Q-learning

Как видно из рис. 5 алгоритм Double Deep Q-learning намного быстрее обучает нейронную сеть, чем алгоритм Deep Q-learning. Кроме того, количество эпизодов при обучении алгоритмом Double Deep Q-learning составило всего 187, с 88 эпизода нейронная сеть в состоянии управлять моделью перевернутого маятника (т.к. награда составляет 200 – максимальное время устойчивости данной модели) (см. рис. 6). Однако, чтобы также начать управлять моделью при использовании алгоритма Deep Q-learning потребовалось 551 эпизод обучение, только с 451 эпизода нейронная сеть была в состоянии управлять моделью перевернутого маятника (см. рис. 7).

Протестируем обученные нейронные сети на управлении моделью перевернутого маятника на тележке (см. рис. 8-9). При обученной нейронной сети маятник должен удерживаться в вертикальном положении на всем временном промежутке (т.е. в течении 200 временных шагов) такие параметры как скорость тележки и скорость изменения наклона шеста должны сходиться к отметке 0 и, если есть отклонения в позиции тележки, они не должны слишком сильно расходиться от оси 0 [16]. С этим справилась нейронная сеть, обученная по алгоритму Double Deep Q-learning.

Рисунок 8. Процесс тестирования после использования алгоритма Deep Q-learning

Рисунок 9. Процесс тестирования после использования алгоритма Deep Q-learning

Можно сделать вывод, что для получения наилучшего результата по управлению моделью перевернутого с минимальными затратами по времени и количеству эпизодов обучения лучше использовать алгоритм Double Deep Q-learning.

**Заключение.**

В данной работе были рассмотрены 2 алгоритма обучения с подкреплением – Deep Q-learning и Double Deep Q-learning. Указанные алгоритмы были применены для решения задачи управления динамической системы перевернутого маятника, взятой из библиотеки моделей обучения с подкреплением Gym. В результате исследования был сделан вывод, что для наилучшего управления моделью перевернутого маятника при минимальных затратах на время обучения и количество эпизодов обучения лучше использовать алго-ритм Double Deep Q-learning.

Внимание авторов!!! Формат заполнения образца строго контролируется, поэтому необходимо его воспроизвести безошибочно!!! Ошибки в оформлении влекут отклонение статьи и задержку ее выхода при получении положительного заключения от экспертов о возможности ее опубликования!!!