УДК 004.85

ИСПОЛЬЗОВАНИЕ DQN ДЛЯ ОБУЧЕНИЯ АГЕНТОВ ИГР (ATARI 2600)

С.П. Зязюлькин
Магистрант кафедры информатики

С.Н. Нестеренков
Доцент кафедры программного обеспечения информационных технологий, кандидат технических наук, доцент

Белорусский государственный университет информатики и радиоэлектроники,
Республика Беларусь
E-mail: nsn@bsuir.by

С.П. Зязюлькин
Окончил Белорусский государственный университет в 2017 году по специальности «Прикладная информатика», магистрант первого года обучения по специальности «Информатика и технологии программирования» БГУИР.

С.Н. Нестеренков
Окончил БГУИР в 2007 году по специальности «Программное обеспечение информационных технологий», окончил магистратуру БГУИР в 2008 по специальности «Системный анализ, управление и обработка информации», окончил аспирантуру БГУИР в 2013 по специальности «Системный анализ, управление и обработка информации», окончил магистратуру БГУИР в 2013 по специальности «Экономика и управление народным хозяйством», в 2017 защитил диссертацию на соискание ученой степени кандидата технических наук по специальности «Системный анализ, управление и обработка информации».

Аннотация. Современное состояние области машинного обучения с подкреплением позволяет обучать агентов, превосходящих человека в некоторых задачах. Примером таких задач являются игры для Atari 2600 – один из популярных бенчмарков для алгоритмов и моделей машинного обучения с подкреплением. В данной статье рассматриваются способы улучшить эффективность классической модели DQN для обучения агентов для игр для Atari 2600.

Ключевые слова: машинное обучение, обучение с подкреплением, Atari 2600, DQN, N-step DQN, double DQN, dueling DQN, noisy networks, prioritized experience replay, temporal consistency loss, expert demonstrations, RND.

Введение. Под агентом понимается сущность, которая взаимодействует с окружающей средой, выполняя на основе наблюдений определённые действия и получая за них награду. Примером агента является робот, который перемещается в пространстве из точки A в точку B, избегая препятствий.

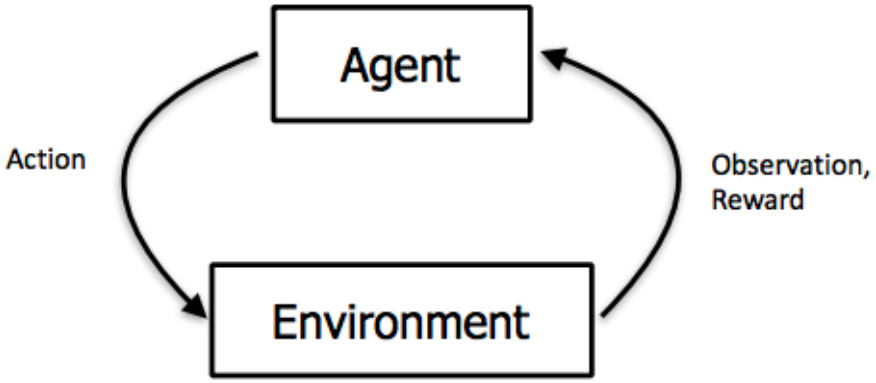

Рисунок *1*. – Взаимодействие агента со средой

В области машинного обучения существует большое число инструментов для обучения агентов. Примером являются различные генетические алгоритмы [1]. В данной статье будут рассмотрены инструменты машинного обучения с подкреплением (Reinforcement Learning, далее RL), адресующего проблему автоматического обучения принятию оптимальных решений. Оптимальность выполнения действия a в состоянии s определяется уравнением оптимальности Беллмана:

$$Q(s,a) = E_{s' \sim P(s,a)}[R(s,a) + \gamma Q(s',a')\,].$$

Q-функция определяет ценность действия a в состоянии s, R – награду за выполнение действия a в состоянии s, γ выступает в роли дисконтирующего множителя. Чем большее значение он имеет, тем выше ценность отложенных наград. Оптимальным действием в состоянии s является действие, имеющее наибольшую ценность.

Нахождение Q-функции для вычисления оптимального действия называется Q-learning. Для обновления Q-функции используется уравнение

$$Q(s_t, a_t) = r_t + \gamma Q(s_{t+1}, a)\,.$$

Для приближения Q-функции в машинном обучении используются глубокие нейронные сети. Такие модели получили название Deep Q-network или просто DQN.

В качестве бенчмарка RL-алгоритмов часто используют игры для Atari 2600. Так, например, в оригинальной статье [2] авторы DQN демонстрировали эффективность модели на семи играх для Atari.

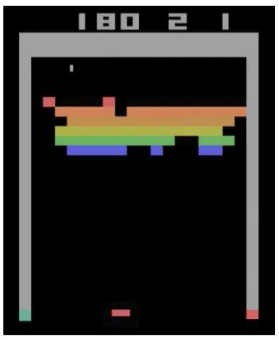

Рисунок *2*. – Игровой процесс игры Breakout для Atari 2600

Улучшение базовой версии DQN. Несмотря на свою эффективность, оригинальный вариант DQN имеет ряд ограничений и недостатков. Далее предложен ряд способов улучшить качество модели как теоретического, так и технического характера.

Первое из предлагаемых улучшений, называемое N-step DQN, было впервые представлено ещё в 1988-м году в статье [3]. Заметим, что уравнение обновления Q-функции является рекурсивным. Если предположить, что агент выбирает оптимальное или близкое к оптимальному действие на шаге $t+1$, то уравнение можно переписать в виде

$$Q(s_t, a_t) = r_t + \gamma r_{t+1} + \gamma^2 Q(s_{t+2}, a)\,.$$

Можно продолжить разворачивать уравнение на произвольное число шагов. Это позволяет обновлять DQN с большим интервалом, что заметно увеличивает скорость обучения. Однако стоит отметить, что нельзя разворачивать уравнение на слишком большое число шагов, т.к. наше предположение о том, что агент во время обучения выбирает оптимальные или близкие к оптимальным действия, в общем случае неверно. В конечном итоге разворачивание уравнения приводит к тому, что процесс обучения начинает расходиться. На практике уравнение обновления Q-функции рекомендуется разворачивать на 2-6 шагов.

Авторы статьи [4] продемонстрировали, что базовый вариант DQN имеет тенденцию переоценивать значение Q-функции, что может негативно сказываться на скорости и качестве обучения и иногда приводить к неоптимальному поведению агентов. Чтобы решить эту проблему, было предложено ввести дополнительную нейронную сеть, называемую целевой. Такой вариант DQN получил название Double DQN. Основная нейронная сеть отвечает за выбор оптимального действия. Целевая нейронная сеть используется для вычисления ценности действия при обновлении весов основной нейронной сети. Раз в несколько итераций веса целевой нейронной сети синхронизируются с весами основной. Уравнение обновления Q-функции принимает вид

$$Q(s_t, a_t) = r_t + \gamma Q'\big(s_{t+1}, argmax_a Q(s_{t+1}, a)\big),$$

где для вычисления Q' используется целевая нейронная сеть.

В классическом варианте DQN для улучшения процесса исследования среды во время обучения используется дополнительный гиперпараметр, определяющий процент случайных действий, выполняемых агентом. Этот параметр постепенно уменьшается в процессе обучения. Такой подход хорошо работает в случае простой среды и коротких эпизодов, но даже в простых случаях он требует подстройки, чтобы процесс обучения был эффективным. Авторы статьи [5] предложили альтернативный вариант улучшения процесса исследования среды – добавить гауссовский шум к весам полносвязных слоёв нейронной сети. Параметры этого шума участвуют в процессе обучения наравне с другими параметрами сети. Такие нейронные сети получили название Noisy Networks.

Модель DQN предполагает использование буфера накопленного опыта (replay buffer) в процессе обучения. В базовой версии DQN сэмплирование данных из буфера накопленного опыта ведётся случайным образом. В статье [6] предлагается улучшить эффективность выборок из буфера за счёт приоритизации сэмплов в зависимости от значения функции потерь. Сэмплирование данных для обучения с учётом этих приоритетов позволяет значительно улучшить сходимость процесса обучения и качество итоговых агентов. Данное улучшение носит название Prioritized Experience Replay.

Значение функции $Q(s, a)$ может быть разбито на две части: ценность состояния $V(s)$ и преимущество $A(s, a)$, которое даёт действие a. В рамках статьи [7] предложено

использовать две различные головы из полносвязных слоёв для вычисления $V(s)$ и $A(s,a)$ вместо $Q(s,a)$. Чтобы гарантировать, что нейронная сеть корректно приближает $V(s)$ и $A(s,a)$, а не, например, зануляет значение $V(s)$, что делает вычисление $A(s,a)$ эквивалентным вычислению $Q(s,a)$, вводится дополнительное ограничение: среднее значение $A(s,a)$ для каждого состояния должно быть равно 0. Итоговый вариант представления $Q(s,a)$ имеет вид:

$$Q(s,a) = V(s) + A(s,a) - \frac{1}{N}\sum_i \quad A(s,a_i).$$

Различие архитектур классической DQN и так называемой Dueling DQN представлено на следующем рисунке.

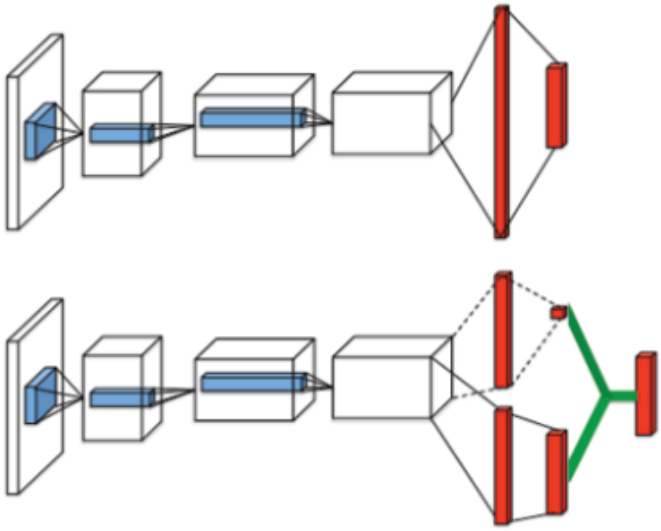

Рисунок *3*. – Классический (сверху) и Dueling (снизу) варианты DQN

Величина дисконтирующего множителя γ определяет ценность отложенных наград. Чем ниже γ, тем меньшую ценность они имеют. Для того чтобы обучить агента принимать действия, приводящие к награде лишь в отдалённой перспективе, необходимо использовать близкие к единице значения γ. Однако использование больших значений γ негативно сказывается на скорости и сходимости обучения. Эта проблема была адресована в работе [8]. Авторы статьи предлагают ввести в функцию потерь дополнительную компоненту, имеющую название temporal consistency (TC) loss. Эта компонента имеет следующий вид:

$$L_{TC}(\theta^N, \theta^{N-1}, s, a) = H\big(Q_{\theta^N}(s,a) - Q_{\theta^{N-1}}(s,a)\big),$$

где $H(x)$ – Huber loss.

На старте обучения предпринимаемые агентом действия близки к случайным. Поэтому в случае среды с разреженными наградами на старте обучения агенту может быть сложно обнаружить набор действий, приводящих к какой-либо награде. Примером такой среды является игра Montezuma’s Revenge для Atari 2600. Для решения этой проблемы было предложено использовать экспертные демонстрации, от которых можно «оттолкнуться» в процессе обучения [9]. Существует большое число вариантов использования экспертных демонстраций: постоянно использовать фиксированную долю экспертных демонстраций, динамически изменять величину доли экспертных демонстраций в процессе обучения, использовать экспертные демонстрации лишь на ранних этапах обучения и т.д.

Одной из основных проблем машинного обучения с подкреплением является исследование среды. Эта проблема была адресована в работе [10]. Авторы статьи предлагают добавить награду за «любопытство», что позволяет повысить эффективность исследования среды и улучшить качество обучаемых агентов для сред с разреженными наградами. Для вычисления награды за «любопытство» вводятся две нейронные сети, имеющие одну и ту же архитектуру. Первая нейронная сеть называется целевой и инициализируется случайными весами, которые остаются неизменными на всём процессе обучения. Вторая сеть называется предсказывающей, она обучается предсказывать выход первой нейронной сети. Обе сети принимают на вход текущее состояние среды (наблюдение). Величина награды за «любопытство» определяется по следующей формуле:

$$r_t^i = \|\hat{f}(s_{t+1}) - f(s_{t+1})\|_2^2,$$

где $\hat{f}$ и f – предсказывающая и целевая сети соответственно. Эта схема, которая получила название RND, представлена на следующем рисунке.

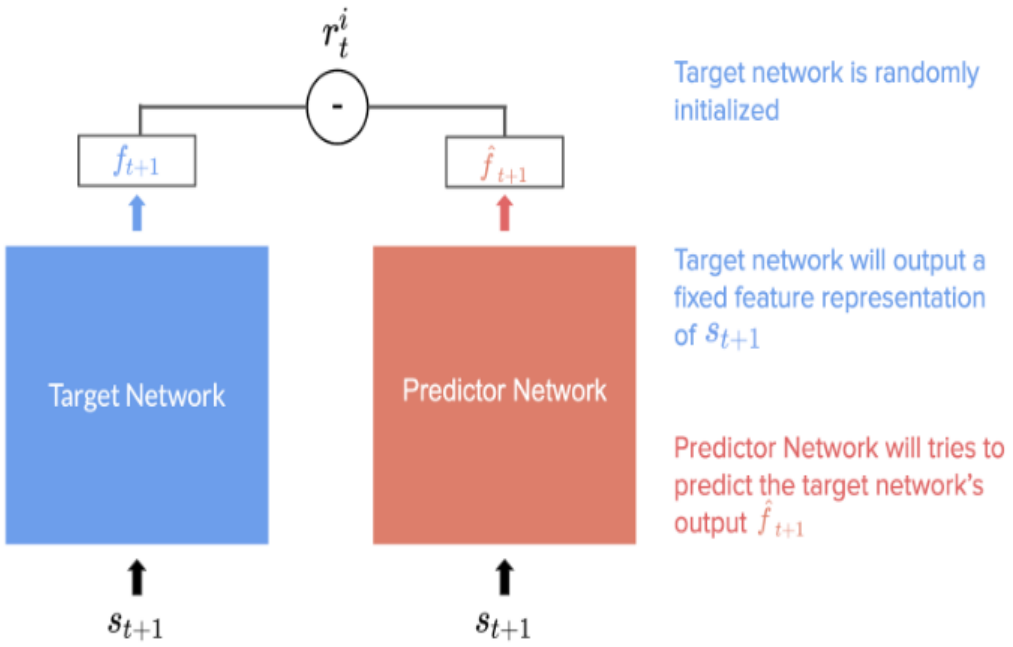

Рисунок *4*. – Random Network Distillation (RND)

Повысить скорость обучения можно не только при помощи теоретических, но и за счёт технических улучшений модели. Процесс обучения состоит из двух частей: обучение агента (обновление весов модели), генерация данных для обучения (взаимодействие агента со средой). Даже для обучения агентов для игры в простые игры для Atari требуется большой объём данных (миллионы игровых кадров). Для решения этой проблемы предлагается использовать несколько копий среды, с которыми будет взаимодействовать агент. Результат взаимодействия агента со всеми копиями среды аккумулируется в буфере накопленного опыта. Более того, процессы обучения и генерации данных являются независимыми, поэтому могут быть разделены на отдельные потоки или процессы. Это позволяет ускорить процесс обучения, особенно в случаях, когда процесс взаимодействия со средой является громоздким и медленным.

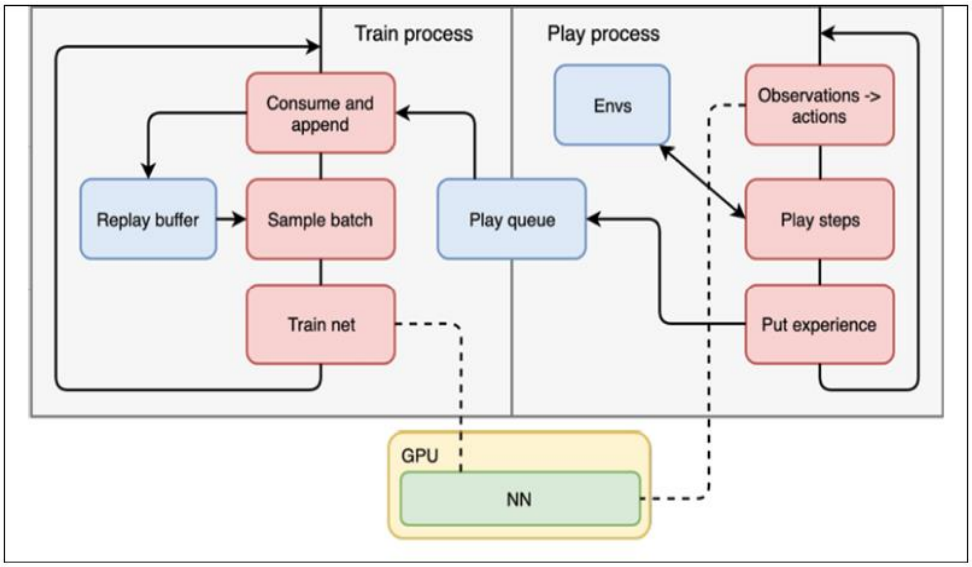

Рисунок 5. – Разделение процесса обучения и генерации данных

Существует большое число технических улучшений, позволяющих значительно ускорить процесс обучения при работе с конкретными средами. В частности, рассмотрим игры для Atari 2600. Игровой кадр представляет собой цветное изображение (3 канала: R, G, B) с разрешением 210x160. Сжатие игрового кадра до чёрно-белого изображения с разрешением 80x80 позволяет уменьшить размер кадра почти в 16 раз, при этом такое преобразование не оказывает заметного влияния на качество обучения. Другим способом повысить скорость процесса обучения является выбор действия не на каждом кадре, а раз в несколько кадров. На промежуточных кадрах предполагается повторять последнее действие. Обработка игрового кадра нейронной сетью является тяжёлой операцией, поэтому этот подход позволяет заметно ускорить процесс обучения.

Заключение. Приведённые улучшения базовой версии DQN позволяют значительно повысить эффективность и качество обучения агентов для игр для Atari 2600 и достичь state-of-the-art результатов на многих из них. Изучение влияния отдельных улучшений на обучение агентов, а также поиск иных улучшений DQN остаются открытыми вопросами

Список литературы

[1.] Нестеренков, С.Н. Модифицированный генетический алгоритм для обучения нейронной сети / С.Н. Нестеренков, К.П. Белов // Информационные технологии и системы 2017 (ИТС 2017): материалы междунар. науч. конф., Минск, 25 окт. 2017 г. / Белорус. гос. ун-т информатики и радиоэлектроники; редкол.: Л.Ю. Шилин [и др.]. – Минск, 2017. – С. 204-205.

[2.] Mnih, V. Playing Atari with Deep Reinforcement Learning / V. Mnih, K. Kavukcuoglu, D. Silver, [и др.] // arXiv:1312.5602.

[3.] Sutton, R.S. Learning to predict by the methods of temporal differences. Mach Learn 3, 9–44 (1988).

[4.] Hasselt, H. Deep Reinforcement Learning with Double Q-learning / H. Hasselt, A. Guez, D. Silver // arXiv:1509.06461.

[5.] Fortunato, M. Noisy Networks for Exploration / M. Fortunato, M.G. Azar, B. Piot, [и др.] // arXiv:1706.10295.

[6.] Schaul, T. Prioritized Experience Replay / T. Schaul, J. Quan, I. Antonoglou, D. Silver // arXiv:1511.05952.

[7.] Wang, Z. Dueling Network Architectures for Deep Reinforcement Learning / Z. Wang, T. Schaul, M. Hessel, [и др.] // arXiv:1511.06581.

[8.] Pohlen, T. Observe and Look Further: Achieving Consistent Performance on Atari / T. Pohlen, B. Piot, T. Hester, [и др.] // arXiv:1805.11593.

[9.] Hester, T. Deep Q-learning from demonstrations / T. Hester, M. Vecerik, O. Pietquin, [и др.] // In Proc. of AAAI. – 2018.

[10.] Burda, Y. Exploration by Random Network Distillation / Y. Burda, H. Edwards, A. Storkey, O. Klimov // arXiv:1810.12894.

USING DQN TO TRAIN AGENTS FOR ATARI 2600 GAMES

S.P. Zyazyulkin
Master student of department of Informatics

S.N. Nesterenkov
PhD Associate professor of department of The software of information technologies

Belarusian State University of Informatics and Radioelectronics,
Republic of Belarus
E-mail: nsn@bsuir.by

Abstract. The current state of deep reinforcement learning makes it possible to train agents that are superior to humans in some tasks. Atari 2600 games are an example of such tasks. Atari 2600 is one of the popular benchmarks for reinforcement learning models and algorithms. This article considers ways to improve performance of the classic DQN model on training agents for Atari 2600 games.

Keywords: machine learning, reinforcement learning, Atari 2600, DQN, N-step DQN, double DQN, dueling DQN, noisy networks, prioritized experience replay, temporal consistency loss, expert demonstrations, RND.