УДК 004.855 DOI: 10.18698/2541-8009-2020-11-652

МЕТОДЫ МУЛЬТИАГЕНТНОГО ОБУЧЕНИЯ С ПОДКРЕПЛЕНИЕМ С ИСПОЛЬЗОВАНИЕМ АЛГОРИТМОВ ТЕОРИИ ИГР

В.Э. Большаков bolshakov.official@gmail.com
SPIN-код: 9554-9749

МГТУ им. Н.Э. Баумана, Москва, Российская Федерация

Аннотация

Рассмотрены методы мультиагентного обучения с подкреплением для стохастических игр с общей суммой. В качестве алгоритма обучения с подкреплением предлагается использовать Q-обучение и его различные модификации, в том числе глубокое Q-обучение. Теоретико-игровой составляющей являются алгоритмы, опирающиеся на такие понятия, как совместные действия агентов, равновесие Нэша и матричные игры. Описана успешная попытка совмещения методов обучения с подкреплением и теории игр для среды мультиагентных стратегических взаимодействий в StarCraft II. Предложен и реализован алгоритм глубокого обучения с подкреплением с поиском равновесия Нэша, или Deep Nash Q-Network (Nash-DQN).

Ключевые слова

Глубокое обучение, теория игр, мультиагентное обучение с подкреплением, равновесие Нэша, нейронные сети, стохастические игры, StarCraft II, поиск равновесия, матричные игры

Поступила в редакцию 26.10.2020

Введение. В области машинного обучения с подкреплением большое внимание уделяется такому методу, как Q-обучение (Q-learning). Преимущества этого метода заключаются в том, что для его использования не нужно формировать модель среды, он прост в понимании и имеет прочную фундаментальную основу в теории марковского процесса принятия решений (Markov decision process, MDP). Все вышесказанное позволяет использовать его при решении множества задач.

В классическом случае Q-обучение помогает единственному агенту сформировать некоторое представление о среде и успешно действовать, основываясь на предыдущем опыте. С одной стороны, если применять его при мультиагентном подходе, результаты далеко не всегда будут удовлетворительными [1]. С другой стороны, такой раздел математики, как теория игр, описывает стратегические взаимодействия и может помочь в выборе лучшей стратегии для нескольких агентов. Таким образом, идеи, возникающие на стыке этих двух областей, могут быть полезны при решении ряда проблем в мультиагентном обучении с подкреплением.

В статье мультиагентная система рассматривается как стохастическая игра с общей суммой, в которой награда каждого агента зависит от совместных действий всех агентов в текущем состоянии, а сами переходы в следующие состояния подчиняются марковскому свойству, т. е. вероятность попадания в опреде-

ленное состояние зависит от нынешнего состояния, а не от последовательности предшествующих состояний.

Основной концепцией нахождения решений для игр с общей суммой является нахождение равновесия Нэша [2]. Каждый агент имеет представление о поведении других игроков, при этом все игроки действуют рационально, т. е. таким образом, что действие агента — это его лучший ответ на стратегии других игроков, и если агент отклонится от такого поведения, то ухудшит свое положение. Тогда равновесие Нэша можно определить как набор стратегий игроков, в котором ни один из участников не может увеличить свой выигрыш, какую бы другую стратегию он не выбрал, если остальные игроки менять стратегии не будут.

Q-обучение и методы теории игр. Q-обучение — это алгоритм обучения с подкреплением, который позволяет агенту найти такую стратегию поведения, которая максимизирует его общую ожидаемую награду в рамках MDP.

Пусть существует набор состояний S. В определенном состоянии $s \in S$ агент может совершить определенный набор действий A. Совершив действие $a \in A$, агент может получить награду $r \in R$ и перейти в другое состояние. С помощью алгоритма Q-обучения пытаются найти такую функцию, которая подсчитывает ценность действий в состояниях, и максимизировать общую награду, используя действия с наибольшей ценностью. Эта функция получила название Q-функция:

$$Q : S \times A \to R.$$

Классическое правило для Q-обучения основано на уравнении Беллмана

$$Q_{new}(s_t, a) = Q(s_t, a) + \left[r_t + \max_a Q(s_{t+1}, a) - Q(s_t, a) \right], \quad (1)$$

где t — шаг взаимодействия со средой, $\alpha \in [0; 1)$ — коэффициент, регулирующий скорость обучения, $\gamma \in [0; 1)$ — дисконтирующий фактор, показывающий степень учета агентом ценности будущих действий.

Для того чтобы агент мог исследовать разные состояния среды, в обучении с подкреплением используется ε-жадная стратегия. Это означает, что на каждом шаге агент с некоторой вероятностью совершает не лучшее из доступных действий, а случайное. Вероятность случайных действий постепенно снижается, и к концу обучения агент уже в большинстве случаев совершает действия намеренно, выбирая лучшие по показателю их ценности.

Часто Q-функцию представляют в виде таблицы, у которой строки соответствуют состояниям, а столбцы — действиям агента, которые он может предпринять в определенном состоянии. На пересечении строки и столбца находится ценность действия, доступного в этом состоянии. Такую таблицу называют Q-таблицей.

Для сравнения с алгоритмами, в которых используются теоретико-игровые концепции, были рассмотрены и реализованы два алгоритма, построенные

только на Q-обучении для двух агентов: Common Q-Table (CQT), в котором два агента заполняют одну Q-таблицу, и Independent Q-Learning (IQL), в котором у каждого агента своя собственная Q-таблица.

Начнем вводить понятия теории игр с понятия «совместные действия» (joint actions). Совместное действие — это комбинация действий всех игроков. Используя этот подход, агент получает награду за действие в зависимости от того, какое действие выбрали другие агенты. На этом принципе основан алгоритм Joint Action Learning (JAL). Он и его модификация Joint Action Learning + Fictitious play (JAL + FP) были рассмотрены и реализованы. Суть техники Fictitious play заключается в том, чтобы запоминать, в каких состояниях агенты находились чаще, и в зависимости от этого выбирать действия для агентов [3].

В последнем из алгоритмов, для которых использовались Q-таблицы, агенты стремятся к поиску равновесия Нэша. Один из методов такого типа был рассмотрен Веллманом и Ху [4]. Они доказали теоретическую сходимость алгоритма при определенных ограничениях, а сам алгоритм был основан на следующем правиле:

$$Q^{i}_{new}(s_t, a_1, \ldots, a_n) = (1-\alpha)Q^{i}(s_t, a_1, \ldots, a_n) + \alpha\left[r^{i}_{t} + \gamma\,\mathrm{Nash}\,Q^{i}(s_{t+1})\right], \qquad (2)$$

где индекс i соответствует обучающемуся агенту из числа всех агентов n; функция $\mathrm{Nash}\,Q^{i}(s_{t+1})$ для разыгрываемого равновесия вычисляет ценность следующего состояния для i-го агента. Ожидается, что все агенты будут выбирать стратегии в соответствии с равновесием для текущей стадии игры. Для этого агенты должны иметь доступ к наградам других игроков.

Рассмотрим каждый шаг обучающего процесса как матричную игру. Для случая с двумя агентами получим биматричную игру с общей суммой. Она определяется таким образом, что каждый агент обозначается буквами, например A_x и A_y. Игрок A_x может сыграть m стратегий, у игрока A_y — n стратегий. Теперь определим матрицы наград X и Y размером $m \times n$, такие, что их элементы x_{ij} и y_{ij} соответственно являются выигрышами агента A_x и A_y в случае, когда A_x играет свою i-ю стратегию, а игрок A_y — j-ю стратегию. Игра считается полностью определенной, когда заданы матрицы выигрышей X и Y. Можно заметить, что как у агентов в типичном для MDP представлении, так и у игроков биматричной игры в терминах теории игр есть некоторые награды. Составив матрицы выигрышей для каждого обучающегося агента из его Q-значений, можно найти равновесие Нэша для такой биматричной игры. После нахождения равновесия можно подставить соответствующие значения в уравнение (2) и перейти к следующей итерации, где процесс нахождения равновесия Нэша будет повторен, причем уже для новых Q-значений. На таком принципе и основан алгоритм NashQ-learning (NashQ).

Среди способов нахождения равновесия Нэша в биматричных играх выделяется метод Лемке — Хоусона [5]. Он является одним из наиболее эффективных на практике.

Глубокое Q-обучение. Ранее было показано, что Q-функция может быть представлена в виде таблицы, которая определяет поведение агента в каждом состоянии. У такого подхода есть ограничения, и при увеличении состояний и доступных действий агентов размер Q-таблицы и занимаемая ей память быстро растет.

Еще одним способом представления Q-функции может быть нейросеть. На вход нейросети подаются некоторые значения, которые характеризуют определенное состояние. Выходной слой нейросети будет возвращать Q-значения для каждого из действий, доступных в этом состоянии.

У такого метода глубокого обучения, как Deep Q-Network (DQN), есть свои особенности. Если нейросеть обучать на каждом шаге, причем только значениями, характеризующими этот единственный шаг (одно состояние среды), то она подстраивает веса под это одно состояние, что приводит к простому запоминанию ситуации, и как итог — к переобучению. Чтобы этого избежать, можно сохранять часть прошлых шагов игрового процесса и использовать эту историю событий в качестве обучающей выборки. Практика показывает, что такой метод делает работу DQN более эффективной [6].

Еще одной техникой, улучшающей результаты DQN, является использование двух нейросетей для одной модели агента. Все дело в том, что изначально веса нейросети инициализированы случайными значениями, проще говоря, связи между нейронами бессмысленны. Добавим к этому тот факт, что агенты используют ε-жадную стратегию, которая тоже вносит в процесс обучения долю случайности. Это в большинстве случаев приводит к тому, что нейросеть неспособна обучиться эффективно. Значительно ослабить этот эффект можно с помощью обучающейся нейросети и целевой нейросети. Первая будет обучаться на выборке из истории событий на каждом шаге алгоритма, но решение о выборе действия всегда будет приниматься на основе выходных значений второй. Целевая нейросеть, в свою очередь, будет через определенное количество шагов обучения копировать веса обучающейся нейросети, тем самым фактически обучаясь, но не на каждой итерации алгоритма.

Типы и архитектуры нейросетей при таком подходе могут быть разными. Рассмотрим DQN с использованием сверточной нейронной сети (Convolutional Neural Network, CNN), а ниже будет представлен алгоритм Nash-DQN, в котором используются нейросети прямого распространения (Fully Connected Feed-Forward Neural Networks, FNN).

Сверточные нейросети пользуются большой популярностью в тех разделах машинного обучения, где требуется обрабатывать изображения [7]. По сути, изображение — это и есть входные данные для CNN. Среда взаимодействия агентов тоже может быть представлена в виде изображения. Рассматривая среду StarCraft II, можно отобразить агентов в виде точек определенного цвета, а расположение этих точек будет соответствовать относительному расположению всех агентов в самой среде взаимодействий. Так формируется карта признаков (feature map) [8].

В реализованном алгоритме DQN на базе CNN выходными значениями нейросети являются совместные действия агентов, то есть нейросеть имеет 64 нейрона на выходном слое. Таким образом, это можно считать модификацией алгоритма JAL. Нейросеть имеет два скрытых сверточных слоя и один полносвязный. Стоит отметить, что в этом алгоритме обучение происходило согласно правилу (1).

Алгоритм Nash-DQN. Последним реализованным алгоритмом является Nash-DQN. Следуя ему, агенты стремятся найти равновесие Нэша. Каждого из двух агентов представляет нейросеть типа Feedforward Neural Network (FNN) [9].

Входными данными являются некоторые признаки среды, расположение агентов, их особенные характеристики. Все это объединяется в вектор признаков постоянного размера и передается на вход каждой нейросети двух агентов. Далее сам подход к обучению является похожим: существует некоторая история событий, из которой формируются выборки для обучения нейросетей, а модели двух агентов также содержат в себе две нейросети, взаимодействующие по тому же принципу, что и в алгоритме DQN. Существенным отличием является то, что обучение происходит по правилу (2). Каждое состояние среды вновь рассматривается как биматричная игра с общей суммой, а агенты действуют в соответствии с текущим равновесием Нэша, которое находится по матрице наград.

В связи с тем, что агенты использую отдельные модели с нейросетями, архитектуру этих нейросетей можно упростить. Теперь каждый агент имеет один скрытый полносвязный слой, а нейронов на выходном слое восемь.

Для взаимодействия со средой StarCraft II использовалась библиотека SMAC [10], с помощью которой и формировался вектор признаков.

Эксперименты с табличными методами Q-обучения. Для всех экспериментов использовалась карта прямоугольной формы размером 16×3 единичных клеток среды. Она была создана в конструкторе карт для StarCraft II. Сценарий эксперимента таков: два союзных агента появляются в одной части карты, два их врага — в противоположной. Союзные агенты — это боевые единицы (юниты) с дальним типом атаки, вражеские юниты — с ближним типом. Эпизод заканчивается при полном уничтожении одной из команд либо при достижении ограничения на число шагов. Единственным условием победы для обучаемых агентов является уничтожение команды врага.

Каждому агенту доступны восемь действий: четыре для перемещения (вверх, вниз, вправо, влево), два для атаки соответствующего врага, и еще два являются пассивным ожиданием.

В экспериментах с использованием Q-таблиц вся карта была разбита на сектора площадью 1×1. Нахождение в определенном секторе принимается за состояние. Также различаются ситуации, когда агент может атаковать, а когда такое действие ему недоступно. Таким образом, количество возможных состояний — 96, а значит, столько и строк в Q-таблице.

Агенты получают награду за атаку врага и штраф за перемещение. Для того чтобы они стремились завершить эпизод победой как можно быстрее, награды распределяются по правилу, соответствующему рис. 1.

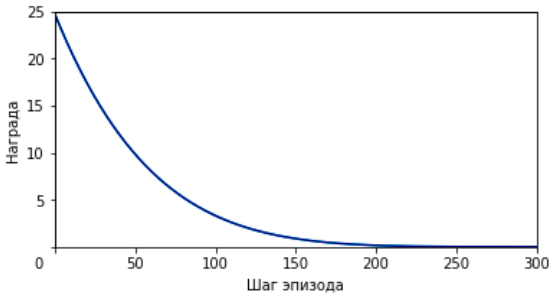

Рис. 1. Зависимость размера награды от времени

Для алгоритма CQT размер Q-таблицы составлял 96×8. В алгоритме IQL таблиц такого же размера было две — по одной для каждого агента. Оптимальное время обучения, при котором агенты гарантированно начинали достигать наибольшей награды, составило 2000 эпизодов. На рис. 2 видно, что после этой отметки размер награды перестает расти.

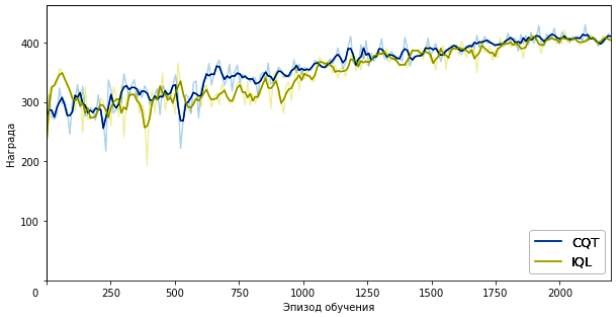

Рис. 2. Награды агентов при обучении алгоритмами CQT и IQL

Для алгоритма JAL и его модификации JAL + FP размер Q-таблицы составил 96×64. Все совместные действия агентов — это все комбинации действий агентов по отдельности. Агенты использовали общую Q-таблицу. Было выдвинуто предположение, что агентам потребуется больше времени на обучение, так как размеры Q-таблицы значительно увеличились. Также было решено в этом эксперименте изменять ε не линейно, а согласно графику на рис. 3.

Время обучения возросло до 10 000 эпизодов. Предполагалось, что агенты будут изучать и закреплять оптимальную стратегию, избегая локально оптимальных стратегий из-за волнообразного изменения ε. Результаты обучения представлены на рис. 4.

Алгоритм JAL показал лучшие результаты. Также видно, что этим методам достаточно прежних 2000 эпизодов для обучения, несмотря на увеличение раз-

меров Q-таблиц. Для дальнейшего сравнения выбран метод JAL как более успешный.

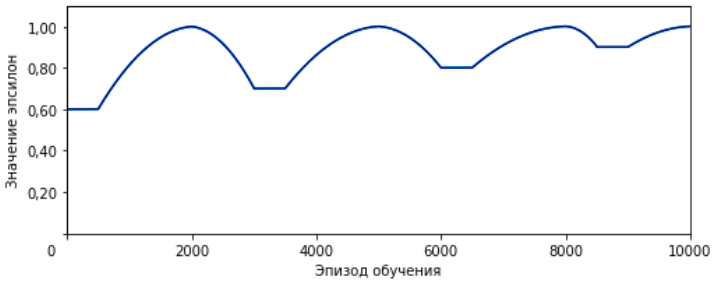

Рис. 3. График зависимости ε от номера эпизода обучения

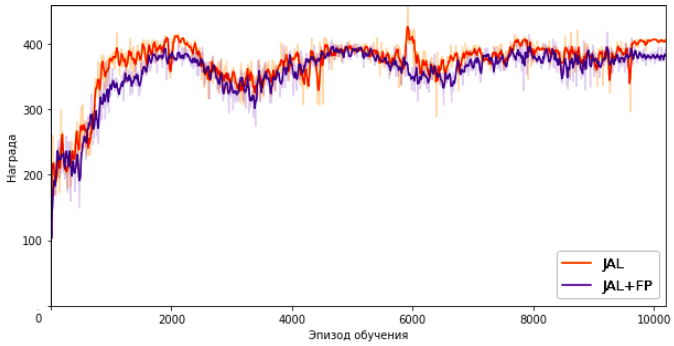

Рис. 4. Награды агентов при обучении алгоритмами JAL и JAL + FP

В алгоритме NashQ каждый агент имеет свою собственную Q-таблицу. Время обучения вновь 10 000 эпизодов, а изменяется по волнообразному закону, как и в предыдущем эпизоде. Для нахождения равновесия Нэша используется библиотека nashpy и метод Лемке — Хоусона. Результаты обучения представлены на рис. 5.

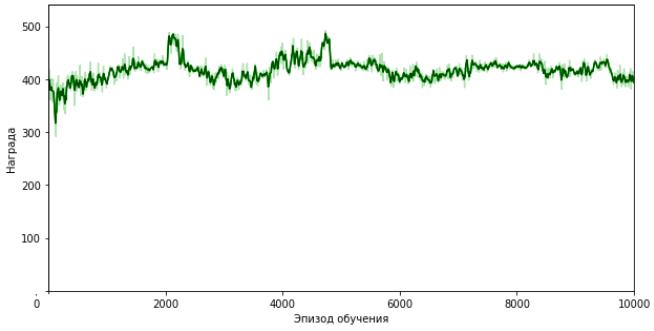

Рис. 5. Награды агентов при обучении алгоритмом NashQ

Сравнение лучших методов обучения с подкреплением с использованием Q-таблиц представлено на рис. 6.

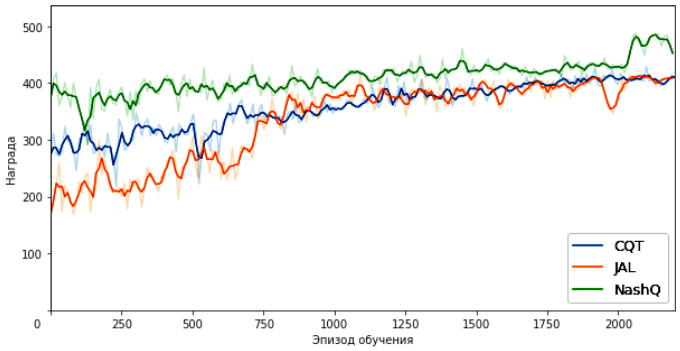

Рис. 6. Сравнение методов CQT, JAL и NashQ

Эксперименты с DQN и Nash-DQN. Q-обучение с использованием нейросетей занимает значительно больше времени, чем при использовании табличных методов. Но стоит отметить, что алгоритм Nash-DQN работает намного быстрее, нежели DQN с использованием CNN.

Для обучения DQN было проведено 10 000 эпизодов игры. Результаты игры представлены на рис. 7.

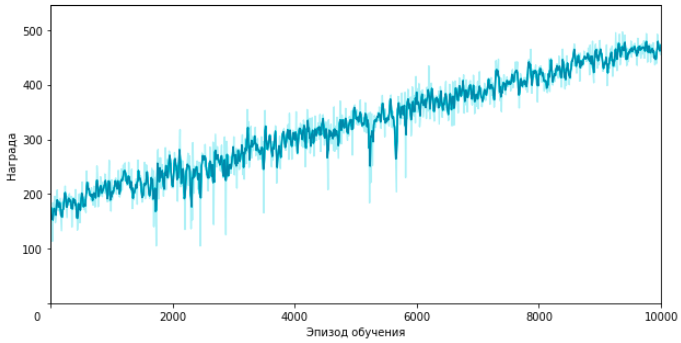

Рис. 7. Награды агентов при обучении алгоритмом DQN

Из-за меньшего количества слоев и нейронов, для обучения Nash-DQN достаточно не более 1 000 эпизодов. Сравнение алгоритмов Nash-DQN и DQN, обученного на двух тысячах эпизодах игры, представлено на рис. 8.

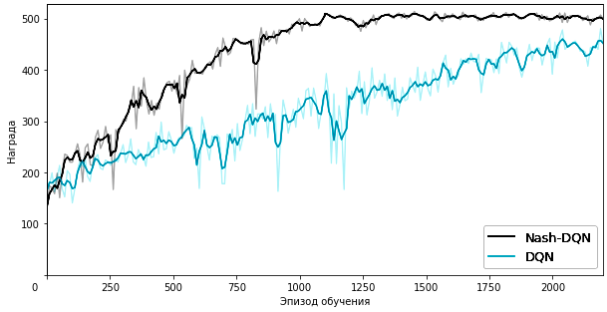

Рис. 8. Результаты обучения агентов методами Nash-DQN и DQN

Из сравнения двух алгоритмов, основанных на глубоком обучении, видно, что методу Nash-DQN нужно намного меньше времени для обучения, а его максимальный результат после 1000 эпизодов обучения превосходит результат DQN даже после 10 000 эпизодов обучения.

Сравним лучшие из всех рассмотренных методов: CQT, JAL, NashQ, Nash-DQN. Результаты сравнения приведены на рис. 9.

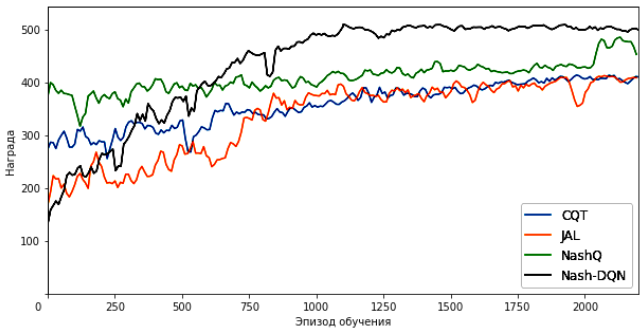

Рис. 9. Общее сравнение самых успешных алгоритмов

Заключение. В статье представлен алгоритм Nash-DQN, который сочетает в себе методы глубокого машинного обучения с подкреплением и методы нахождения равновесия Нэша в стохастической игре с общей суммой.

Было выяснено, что объединение классического подхода к Q-обучению с методами теории игр помогает улучшить результаты обучения в мультиагентной среде. При сравнении самых успешных из рассмотренных алгоритмов видно, что лучшими оказались методы, использующие концепцию теории игр, а именно находящие равновесие Нэша на каждой стадии эпизода игры.

На примере StarCraft II показано, что предложенный алгоритм Nash-DQN способен давать хорошие результаты в задачах повышенной сложности в среде с несколькими обучающимися агентами, при этом по скорости обучения данный метод превосходит другие известные подходы.

Оказались неэффективными алгоритмы JAL и JAL + FP. Время на обучение при использовании этих методов превышает время, затрачиваемое агентами на обучение с использованием CQT или IQL, т. е. методов с чисто табличным подходом. Также показано, что DQN с использованием CNN значительно уступает по времени обучения алгоритму Nash-DQN.

Литература

[1] Hausknecht M., Stone P. Deep recurrent Q-learning for partially observable MDPs. *AAAI Fall Symp. Sequential Decision Making for Intelligent Agents*, 2015. URL: https://arxiv.org/pdf/1507.06527.pdf (дата обращения: 15.06.2020).

[2] Nash J. Non-cooperative games. *Ann. Math.*, 1951, vol. 54, no. 2, pp. 286–295. DOI: https://doi.org/10.2307/1969529

[3] Abernethy J., Lai K.A., Wibisono A. Fictitious play: convergence, smoothness, and optimism. *arxiv.org: веб-сайт*. URL: https://arxiv.org/abs/1911.08418v1 (дата обращения: 15.06.2020).
[4] Wellman M.P., Hu J. Nash Q-learning for general-sum stochastic games. *J. Mach. Learn. Res.*, 2003, vol. 4, no. 4, pp. 1039–1069.
[5] Lemke C.E., Howson J.T.Jr. Equilibrium points of bimatrix games. *J. Soc. Ind. Appl. Math.*, 1964, vol. 12, no. 2, pp. 413–423. DOI: https://doi.org/10.1137/0112033
[6] Foerster J., Nardelli N., Farquhar G., et al. Stabilising experience replay for deep multi-agent reinforcement learning. *Proc. 34th Int. Conf. Machine Learning*, 2017, pp. 1146–1155.
[7] Krizhevsky A., Sutskever I., Hinton G.E. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.
[8] Алфимцев А.Н. Декларативно-процессная технология разработки интеллектуальных мультимодальных интерфейсов. Автореф. дисс. ... док. тех. наук. М., ИПУ РАН, 2016.
[9] Dai D., Tan W., Zhan H. Understanding the feedforward artificial neural network model from the perspective of network flow. *arxiv.org: веб-сайт*. URL: https://arxiv.org/abs/1704.08068 (дата обращения: 15.06.2020).
[10] Samvelyan M., Rashid T., de Witt C.S., et al. The starcraft multi-agent challenge. accepted at the workshop on deep reinforcement learning. *arxiv.org: веб-сайт*. URL: https://arxiv.org/abs/1902.04043 (дата обращения: 15.06.2020).

Большаков Владислав Эдуардович — магистрант кафедры «Информационные системы и телекоммуникации», МГТУ им. Н.Э. Баумана, Москва, Российская Федерация.

Научный руководитель — Алфимцев Александр Николаевич, доктор технических наук, профессор кафедры «Информационные системы и телекоммуникации», МГТУ им. Н.Э. Баумана, Москва, Российская Федерация.

Ссылку на эту статью просим оформлять следующим образом:
Большаков В.Э. Методы контроля и высокочастотное диагностирование полиамидных изделий. *Политехнический молодежный журнал*, 2020, № 11(52). http://dx.doi.org/10.18698/2541-8009-2020-11-652

MULTI-AGENT LEARNING METHODS WITH REINFORCEMENT USING GAME THEORY ALGORITHMS

V.E. Bolshakov

bolshakov.official@gmail.com
SPIN-code: 9554-9749

Bauman Moscow State Technical University, Moscow, Russian Federation

Abstract

The paper considers the methods of multi-agent learning with reinforcement for stochastic games with total sum. It is proposed to use Q-learning and its various modifications, including deep Q-learning, as a reinforcement learning algorithm. The game-theoretic component consists of algorithms based on concepts such as joint actions of agents, Nash equilibrium, and matrix games. Authors describe a successful attempt to combine reinforcement learning and game theory for a multi-agent strategic interaction environment in StarCraft II. An algorithm for deep reinforcement learning with Nash equilibrium search, or Deep Nash Q-Network (Nash-DQN), is proposed and implemented.

Keywords

Deep learning, game theory, multi-agent reinforcement learning, Nash equilibrium, neural networks, stochastic games, StarCraft II, equilibrium search, matrix games

Received 26.10.2020

References

[1] Hausknecht M., Stone P. Deep recurrent Q-learning for partially observable MDPs. *AAAI Fall Symp. Sequential Decision Making for Intelligent Agents*, 2015. URL: https://arxiv.org/pdf/1507.06527.pdf (accessed: 15.06.2020).

[2] Nash J. Non-cooperative games. *Ann. Math.*, 1951, vol. 54, no. 2, pp. 286–295. DOI: https://doi.org/10.2307/1969529

[3] Abernethy J., Lai K.A., Wibisono A. Fictitious play: convergence, smoothness, and optimism. *arxiv.org: website*. URL: https://arxiv.org/abs/1911.08418v1 (accessed: 15.06.2020).

[4] Wellman M.P., Hu J. Nash Q-learning for general-sum stochastic games. *J. Mach. Learn. Res.*, 2003, vol. 4, no. 4, pp. 1039–1069.

[5] Lemke C.E., Howson J.T.Jr. Equilibrium points of bimatrix games. *J. Soc. Ind. Appl. Math.*, 1964, vol. 12, no. 2, pp. 413–423. DOI: https://doi.org/10.1137/0112033

[6] Foerster J., Nardelli N., Farquhar G., et al. Stabilising experience replay for deep multi-agent reinforcement learning. *Proc. 34th Int. Conf. Machine Learning*, 2017, pp. 1146–1155.

[7] Krizhevsky A., Sutskever I., Hinton G.E. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[8] Alfimtsev A.N. Deklarativno-protsessnaya tekhnologiya razrabotki intellektual'nykh mul'timodal'nykh interfeysov. Avtoref. diss. dok. tekh. nauk [Declarative-processive development technology for intelligent multimode interfaces. Abs. doc. tech. sci. diss.]. Moscow, ICS RAS Publ., 2016 (in Russ.).

[9] Dai D., Tan W., Zhan H. Understanding the feedforward artificial neural network model from the perspective of network flow. *arxiv.org: website*. URL: https://arxiv.org/abs/1704.08068 (accessed: 15.06.2020).

[10] Samvelyan M., Rashid T., de Witt C.S., et al. The starcraft multi-agent challenge. accepted at the workshop on deep reinforcement learning. *arxiv.org: website*. URL: https://arxiv.org/abs/1902.04043 (accessed: 15.06.2020).

Bolshakov V.E. — Master's Degree Student, Department of Information Systems and Telecommunications, Bauman Moscow State Technical University, Moscow, Russian Federation.

Scientific advisor — Alfimtsev A.N., Professor, Department of Information Systems and Telecommunications, Bauman Moscow State Technical University, Moscow, Russian Federation.

Please cite this article in English as:
Bolshakov V.E. Multi-agent learning methods with reinforcement using game theory algorithms. *Politekhnicheskiy molodezhnyy zhurnal* [Politechnical student journal], 2020, no. 11(52). http://dx.doi.org/ 10.18698/2541-8009-2020-11-652.html (in Russ.).