УДК 004.85 DOI: 10.18698/2541-8009-2023-11-952

ПРИМЕНЕНИЕ МЕТОДОВ МАШИННОГО ОБУЧЕНИЯ ДЛЯ УПРАВЛЕНИЯ ПОВЕДЕНИЕМ АГЕНТА

М.А. Федотов mixxxaile@gmail.com
А.Ю. Чапаев ayuchapaev@gmail.com

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

Аннотация

Работа посвящена применению методов машинного обучения для управления агентом. Рассмотрен метод обучения с подкреплением. Выполнено сравнение алгоритмов обучения с подкреплением: Q-learning, SARSA, EV-SARSA, DDQN, при этом DDQN является наиболее подходящим для управления поведением агента в недетерминированной среде. Реализован алгоритм DDQN на языке программирования C++. Разработанная реализация метода машинного обучения применена для управления агентом в игровом приложении «Змейка». Представлены вычислительные эксперименты по исследованию эффективности разработанного метода машинного обучения для управления поведением агента. Эксперименты демонстрируют преимущества использования DDQN в условиях изменяющихся условий среды, подтверждая эффективность алгоритма для решения задач управления поведением агента.

Ключевые слова

Искусственная нейронная сеть, методы машинного обучения, обучение с подкреплением, Double Deep Q-learning, алгоритм оптимизации, управление агентом, гиперпараметры, скорость обучения

Поступила в редакцию 15.11.2023

Введение. В условиях развития технологий машинное обучение является перспективным инструментом для создания инновационных решений во многих областях, включая робототехнику. Применение методов машинного обучения в контексте управления поведением роботизированных агентов (испытуемых систем) позволяет создавать системы, обладающие гибкостью, интеллектуальными способностями и высокой эффективностью. В ситуациях, где агент действует в переменчивой и не всегда предсказуемой среде (например, при городской доставке грузов), способность к самообучению и адаптации критически важна.

Одним из ключевых аспектов успешного применения машинного обучения в управлении поведением агента является выбор наиболее подходящей модели и метода. Верный выбор улучшает работоспособность и адаптивность роботизированных систем, позволяя им обучаться «на лету», адаптируясь к различным условиям и задачам.

Данное исследование посвящено выбору наиболее подходящего метода машинного обучения для управления поведением роботизированного агента.

В работе представлены обзор существующих методов и алгоритмов машинного обучения, детальное описание реализации выбранного метода, описание проведенных вычислительных экспериментах, а также анализ полученных результатов.

Обзор методов машинного обучения для управления поведением агента. В области робототехники существуют различные подходы к обучению агентов.

Популяционные алгоритмы, к которым относятся генетические алгоритмы и стратегии эволюции, представляют собой специализированные методы оптимизации и поиска в глобальном пространстве возможных решений. Эти методы основаны на принципах, вдохновленных биологической эволюцией, таких как селекция, скрещивание и мутация.

Основная идея заключается в том, чтобы «выращивать» популяцию потенциальных решений и систематически улучшать ее путем отбора наиболее подходящих кандидатов и их комбинации, пока не будет достигнуто оптимальное или приемлемое решение. Популяционные алгоритмы особенно эффективны в ситуациях, когда стандартные методы поиска и оптимизации сталкиваются с проблемами. Это может произойти, когда пространство решений обладает высокой размерностью [1], что делает его изучение крайне затратным, или когда задача обладает сложной структурой, которая не поддается анализу традиционными методами.

Обучение с учителем — подход в области машинного обучения, который ориентирован на обработку четко структурированных данных [2, с. 4]. В основе метода лежит использование предварительно подготовленного набора данных, который состоит из пар «входные данные — выходные данные». В каждой из этих пар входные данные представляют собой конкретный пример или ситуацию, в то время как выходные данные служат «ответом» или результатом для данного примера.

Принцип работы агента в контексте обучения с учителем заключается в изучении зависимостей и паттернов, присутствующих в предоставленном наборе данных. По сути, агент стремится обнаружить функциональные соответствия между входными и выходными данными, чтобы впоследствии, получив новые, ранее неизвестные данные на входе, корректно предсказать, какие действия следует выполнять на выходе. Главная цель — достичь такой степени обобщения, при которой модель будет способна делать точные предсказания на новых, ранее неизвестных данных, опираясь на информацию, извлеченную из обучающего набора.

Обучение с подкреплением — метод машинного обучения, который не полагается на предварительно аннотированные данные, а вместо этого ориентируется на взаимодействие агента с его окружающей средой. В процессе этого взаимодействия агент предпринимает действия и получает обратную связь от среды в форме наград или штрафов, которые отражают правильность или ошибочность его действий [3]. Основная цель такого подхода — максимизировать суммарное вознаграждение за определенный период времени или последователь-

ность действий. Общая схема алгоритма машинного обучения с подкреплением представлена на рис. 1.

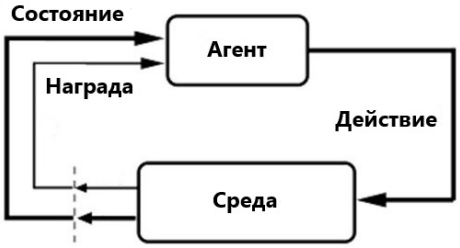

Рис. 1. Схема алгоритма машинного обучения с подкреплением

Машинное обучение с подкреплением базируется на модели марковского процесса принятия решений (МППР). В такой модели агент оценивает вероятности вознаграждений для каждого возможного действия в каждом состоянии и на основе этих оценок принимает решения о том, какое действие предпринять. После выполнения действия агент наблюдает результат и обновляет свои знания на основе полученной обратной связи [4].

В контексте робототехники обучение с подкреплением особенно актуально. Роботы часто действуют в сложных, динамичных и непредсказуемых средах. В таких условиях методы обучения с подкреплением позволяют агентам самостоятельно адаптироваться к изменяющимся обстоятельствам, постоянно корректируя свое поведение на основе накопленного опыта и полученной обратной связи от среды.

Обзор алгоритмов машинного обучения с подкреплением для управления поведением агента. Алгоритм машинного обучения с подкреплением Q-Learning разработан с целью обеспечить эффективное обучение агента в ситуациях, где точное знание о динамике окружающей среды отсутствует. Функция оценки действий (Q-функция) оценивает ожидаемую долгосрочную награду за действие, предпринятое в определенном состоянии [5]. Эта оценка не только показывает полезность конкретного действия в данном состоянии, но и служит основой для выбора действий агента в будущем. Агент, использующий Q-Learning, постоянно обновляет свои Q-значения по формуле на основе полученных наград и результатов своих действий. Q-функцию вычисляют по следующей формуле:

$$Q(s,a) = (1-\alpha)Q(s,a) + \alpha\left[r(s,a,s') + \gamma \max_{a'} Q(s',a')\right],$$

где $Q(s,a)$ — Q-функция действия a в состоянии s; α — скорость обучения, определяющая, насколько сильно обновляется Q-значение на каждой итерации; $r(s,a,s')$ — награда при переходе в состояние s' из s при действии a; γ — дис-

контный фактор, определяющий насколько сильно агент ориентируется на будущие награды; $\max_{a'} Q(s', a')$ — максимальная Q-функция действия a' в состоянии s'.

При правильной настройке скорости обучения α и дисконтного фактора γ, а также достаточном количестве итераций значения функции оценки действий сходятся к оптимальным [6], что позволяет агенту действовать максимально эффективно в любом возможном состоянии.

Метод SARSA является еще одним методом обучения с подкреплением. Основное различие между SARSA и Q-Learning заключается в том, как они учитывают будущие действия: SARSA учитывает текущую политику при выборе действий [7], а Q-Learning ориентирован на максимизацию ожидаемой награды. Q-функцию для метода обучения SARSA вычисляют по следующей формуле:

$$Q(s,a) = (1-\alpha)Q(s,a) + \alpha\left[r(s,a,s') + \gamma Q(s', a')\right].$$

Метод EV-SARSA (Expected Value SARSA) — это усовершенствованный вариант алгоритма SARSA. Основная идея EV-SARSA заключается в использовании ожидаемого значения следующего действия при обновлении Q-значений, вместо фактического действия, как это делается в традиционном методе SARSA. Это позволяет сгладить оценки и делает процесс обучения более стабильным, особенно в условиях, где следующее действие может иметь высокую степень неопределенности. Для вычисления значения Q-функции в EV-SARSA используется следующая формула:

$$Q(s,a) = (1-\alpha)Q(s,a) + \alpha\left[r(s,a,s') + \gamma E_{a'} Q(s', a')\right],$$

где $E_{a'} Q(s', a')$ — математическое ожидание Q-функции в состоянии s'.

Алгоритм Double Deep Q Network (DDQN) представляет собой развитие стандартного Q-Learning метода. Основное решение, предложенное в DDQN, заключается в использовании двух отдельных нейронных сетей для выполнения разных задач в процессе обучения: первой — для определения оптимального действия, второй — для оценки Q-значения выбранного действия. Такое разделение позволяет устранить систематические ошибки переоценки Q-значений [8], которые часто возникают в традиционных методах Q-Learning. Устранение проблемы переоценки Q-значений ведет к более точному определению стратегий поведения, что важно для роботов, работающих в реальном мире. Поэтому использование алгоритма DDQN обеспечивает повышенную стабильность и надежность при обучении роботов, что, в свою очередь, повышает безопасность их эксплуатации в сложных и непредсказуемых средах.

Программная реализация алгоритма DDQN. В исследовании применен метод машинного обучения с подкреплением DDQN в игровом приложении «Змейка». В роли агента выступает сущность «змея», а в качестве среды — игровое поле.

Для представления Q-функции, оценивающей качество действий агента в различных ситуациях, выбрана полносвязная нейронная сеть. Сеть имеет четырехслойную архитектуру, обеспечивая баланс между вычислительной эффективностью и способностью выявлять сложные зависимости. Она состоит из 28 входных нейронов, каждый из которых отображает параметр или свойство игрового окружения. Сеть завершается выходным слоем из четырех нейронов, соответствующих четырем возможным направлениям движения агента. Графическое представление используемой нейронной сети показано на рис. 2.

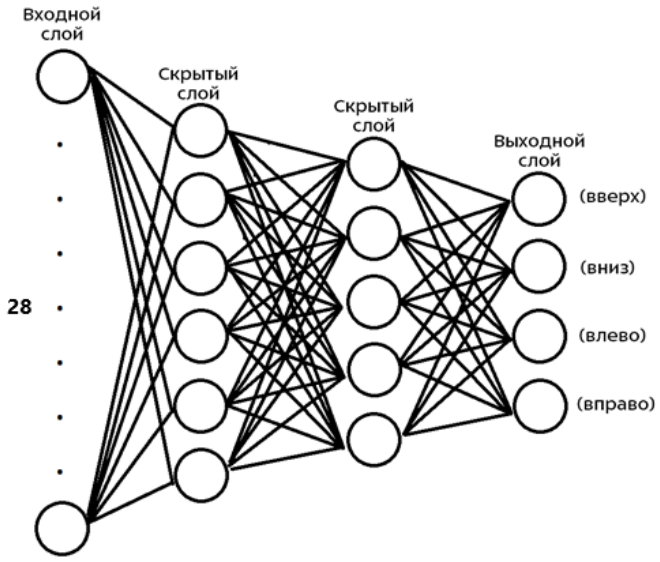

Рис. 2. Графическое представление используемой нейронной сети

На всех слоях, кроме выходного, используется функция активации ReLU, благодаря которой сеть может выделять и комбинировать полезные признаки из входных данных:

$$\mathrm{ReLU}(x) = \max(0, x).$$

Функция ошибки измеряет среднее квадратичное отклонение между предсказанными и фактическими значениями. Чем меньше значение функции, тем лучше модель соответствует данным. В рассматриваемой модели используется среднеквадратичная функция ошибки

$$\mathrm{MSE}=\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2.$$

В методе DDQN используются две нейронные сети: основная сеть для генерации действий и целевая сеть для вычисления ожидаемого Q-значения. Это обеспечивает стабильное и эффективное обучение.

Q-функцию в процессе обучения определяют как

$$Q(s,a) = (1-\alpha)Q(s,a) + \alpha\left[r(s,a,s') + \gamma \max_{a'} Q(s',a')\right].$$

Оптимизация параметров (обучение) нейронной сети осуществляется с помощью метода обратного распространения ошибки [2, с. 107]. Этот метод основывается на принципе градиентного спуска, который позволяет найти оптимальные значения весов нейронов для минимизации ошибки модели.

Общий алгоритм работы нейронных сетей в DDQN имеет следующий вид.

1. Инициализация:

– определяются априорные значения для весов всех нейронов в обеих нейронных сетях;

– задается скорость обучения (learning rate) и другие гиперпараметры.

2. Прямой проход:

– подаются входные данные (состояние среды) на входной слой основной и целевой сетей;

– в каждой сети выполняются вычисления для каждого нейрона на каждом слое, передавая значения от предыдущего слоя к следующему с помощью функции активации (ReLU);

– вычисляется значение Q-функции для каждого действия.

3. Вычисление ошибки:

– сравнивается предсказанное значение функции Q от основной сети с суммой награды и максимальным значением функции Q следующего состояния от целевой сети;

– вычисляется градиент ошибки на выходном слое.

4. Обратное распространение ошибки:

– начиная с последнего слоя, вычисляются градиенты ошибки для каждого нейрона на предыдущем слое основной нейронной сети;

– градиенты ошибки передаются от текущего слоя к предыдущему слою с учетом правила дифференцирования сложных функций;

– вычисляются градиенты ошибки для всех нейронов на каждом слое, двигаясь от выходного слоя к входному.

5. Обновление весов основной сети:

– обновляются веса каждого нейрона на каждом слое, используя вычисленные градиенты ошибки и скорость обучения;

– корректируются веса в направлении, противоположном градиенту ошибки.

6. Обновление весов целевой сети:

– после каждого определенного количества итераций веса целевой сети обновляются путем копирования весов из основной сети.

7. Повторение процесса:

– шаги 2–5 повторяются на протяжении заданного количества эпох обучения.

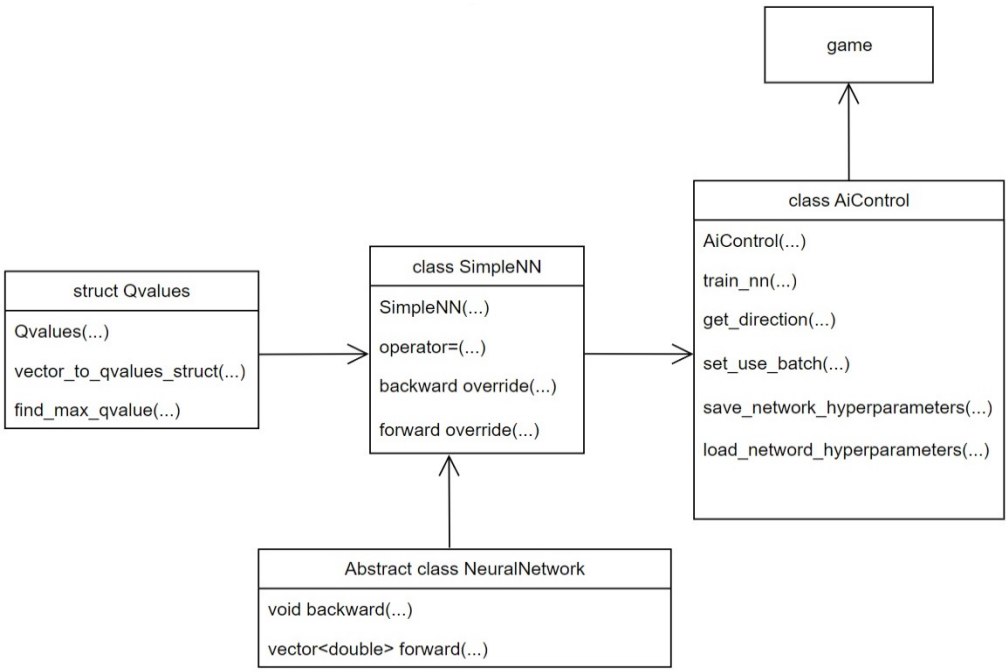

Рис. 3. Диаграмма классов программной реализации

Для реализации алгоритма выбран язык программирования C++, который характеризуется высокой производительностью, что критично при работе с нейронными сетями. Разработан абстрактный класс NeuralNetwork, задающий основные методы, такие как прямое и обратное распространение ошибки. Для представления конкретной архитектуры полносвязной сети создан производный класс SimpleNN. В дополнение к этому для интеграции нейронной сети в игровое приложение и ее обучения разработан класс aiControl. Также реализована структура Qvalues, хранящая Q-функцию для действий up, down, left и right. Диаграмма классов программной реализации представлена на рис. 3.

Вычислительные эксперименты. В рамках проведения вычислительных экспериментов создана нейронная сеть на основе метода обучения с подкреплением и алгоритма Double Deep Q-Network. Основной проблемой стало переобучение агента, когда агент слишком точно запоминает обучающие данные и теряет способность к обобщению на новые ситуации. Для решения этой проблемы было проведено четыре эксперимента.

В первом проанализирована и изменена структура состояния. Структура состояния — набор параметров, которые описывают текущее положение агента в контексте игрового приложения. От состояния зависит, как агент воспринимает свое окружение, что напрямую влияет на то, как он обучается и принимает решения. Если состояние слишком простое, оно может не предоставить агенту достаточно информации для эффективного обучения. Если состояние слишком сложное или избыточное, это может затруднить обучение, поскольку агенту будет

сложнее найти полезные обобщения, и он может столкнуться с проблемой «проклятия размерности». По итогам эксперимента удалось упростить структуру состояния, при этом улучшив вычислительную эффективность нейронной сети.

Во втором эксперименте исследованы способы начисления награды. Оказалось, что, если награды слишком редки или недостаточно информативны, агенту может быть сложно узнать, какие действия полезны. Если награды слишком часты или сложны, они могут привести к переобучению или создать шум, который затруднит обучение. В результате эксперимента способы начисления награды пересмотрены с учетом множества параметров, что способствовало улучшению адаптации агента к условиям в недетерминированной среде.

В третьем испытании оптимизирована скорость обучения α. Установлено, что слишком большая скорость обучения может привести к тому, что в процессе обучения мы будем «перепрыгивать» через минимум функции потерь. На каждом шаге обучения веса обновляются настолько сильно, что модель пропускает точку минимума и «отскакивает» на другую сторону. Однако слишком маленькая скорость обучения может привести к чрезмерно медленному обучению. Модель будет делать небольшие шаги на каждом этапе обучения, и это может привести к тому, что она не достигнет минимума функции потерь за разумное время. Кроме того, слишком маленькая скорость обучения может вызвать застревание в локальных минимумах функции потерь, когда модель находит не самое оптимальное, но удобное решение и не может из него выбраться. По итогам эксперимента найдено оптимальное значение скорости обучения, с которым агент смог достигнуть баланса между скоростью и точностью обучения.

Наконец, в четвертом эксперименте варьировали значение коэффициента дисконтирования γ, важного для определения значимости будущих наград. Слишком большой коэффициент дисконтирования означает, что модель слишком сильно фокусируется на долгосрочном будущем, пренебрегая непосредственными вознаграждениями. Это может привести к тому, что модель будет игнорировать текущую ситуацию и принимать решения, которые могут показаться неоптимальными в ближайшем будущем, даже если они обещают большое вознаграждение в далеком будущем. Слишком маленький коэффициент дисконтирования означает, что модель слишком сосредоточена на непосредственном вознаграждении и не обращает должного внимания на долгосрочные последствия своих действий. Это может способствовать поведению, когда модель выбирает действия, которые приводят к максимальному немедленному вознаграждению, но могут быть неоптимальными в долгосрочной перспективе. В результате найдено оптимальное значение коэффициента дисконтирования, которое уравновешивает важность немедленных и долгосрочных вознаграждений. Это позволило агенту принимать решения, которые не только приводят к немедленной выгоде, но и учитывают долгосрочные последствия, обеспечивая стабильное и эффективное обучение.

В результате экспериментов реализована новая структура состояния, улучшены способы начисления награды и определены оптимальные гиперпараметры для обучения. С использованием новой структуры состояния и улучшенных способов начисления наград агент получил возможность лучше понимать свое окружение и более точно отслеживать свои цели. Оптимизированные гиперпараметры обучения способствуют глубокому и стабильному обучению агента, позволяя ему развивать эффективные стратегии для достижения своих целей. В итоге достигнуто значительное увеличение производительности агента, что подтверждено результатами испытаний.

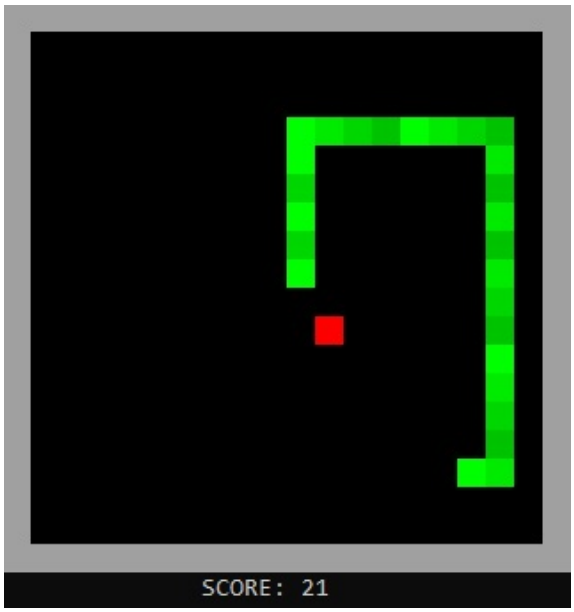

Рис. 4. Процесс игры обученной змейки

Заключение. В ходе исследования рассмотрено применение методов машинного обучения к задаче управления поведением агента с целью оптимизации получаемой награды. Основной фокус направлен на применение методов обучения с подкреплением, особенно в условиях недетерминированной среды. В качестве практической платформы для экспериментов выбрано игровое приложение «Змейка», разработанное на языке программирования C++. В основе выбранного метода машинного обучения лежит алгоритм Double Deep Q-Network. Благодаря использованию обученной нейронной сети агент демонстрирует способность эффективно обрабатывать наборы данных, выделяя и анализируя сложные зависимости для достижения поставленных задач. Тем не менее отметим, что выбор и настройка гиперпараметров играют критическую роль: неправильная конфигурация приводит к переобучению, снижая адаптивность агента к новым условиям и ситуациям. В дальнейших планах

предусмотрено глубокое изучение вопроса подбора оптимальных гиперпараметров, а также проработка возможности интеграции других методов машинного обучения для достижения еще более высокой эффективности.

Литература

[1] Козов А.В. Сравнение эффективности некоторых модификаций алгоритма эволюционной стратегии. *Политехнический молодежный журнал*, 2018, № 5 (22). http://dx.doi.org/10.18698/2541-8009-2018-5-309

[2] Воронцов К. *Математические методы обучения по прецедентам (теория обучения машин)*. URL: http://www.machinelearning.ru/wiki/images/6/6d/Voron-ML-1.pdf (дата обращения 15.10.2023).

[3] Sutton R.S., Barto A.G. *Reinforcement Learning: An Introduction*. London, MIT Press, 1998, pp. 1–11.

[4] Littman M.L. Markov decision processes. *International Encyclopedia of the Social and Behavioral Sciences*, 2012, pp. 573–575. http://doi.org/10.1016/b0-08-043076-7/00614-8

[5] Кузьмин В. Использование нейронных сетей в алгоритме Q-learning. *Транспорт и телекоммуникации*, 2003, т. 4, № 1, с. 74–86.

[6] Melo F.S. *Convergence of Q-learning: a simple proof. Institute for Systems and Robotics*. URL: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.413.2350&rep=rep1&type=pdf (accessed October 15, 2023).

[7] Herrmann M. *RL 5: On-policy and off-policy algorithms*. University of Edinburgh, School of Informatics. URL: https://www.inf.ed.ac.uk/teaching/courses/rl/slides15/rl05.pdf (accessed October 15, 2023).

[8] Hasselt H. van, Guez A., Silver D. Deep Reinforcement Learning with Double Q-learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2015, vol. 30 (1). http://doi.org/10.1609/aaai.v30i1.10295

Федотов Михаил Андреевич — студент кафедры «Системы автоматизированного проектирования», МГТУ им. Н.Э. Баумана, Москва, Российская Федерация.

Чапаев Артем Юрьевич — студент кафедры «Системы автоматизированного проектирования», МГТУ им. Н.Э. Баумана, Москва, Российская Федерация.

Научный руководитель — Козов Алексей Владимирович, старший преподаватель кафедры «Системы автоматизированного проектирования», МГТУ им. Н.Э. Баумана, Москва, Российская Федерация. E-mail: alexey.kozov@bmstu.ru

Ссылку на эту статью просим оформлять следующим образом:

Федотов М.А., Чапаев А.Ю. Применение методов машинного обучения для управления поведением агента. *Политехнический молодежный журнал*, 2023, № 11 (88). http://dx.doi.org/10.18698/2541-8009-2023-11-952

APPLYING THE MACHINE LEARNING METHODS IN THE AGENT BEHAVIOR CONTROL

M.A. Fedotov mixxxaile@gmail.com
A.Yu. Chapaev ayuchapaev@gmail.com

Bauman Moscow State Technical University, Moscow, Russian Federation

Abstract

The paper is devoted to introducing the machine learning methods in agent control. It considers the reinforcement learning method. The following reinforcement learning algorithms are compared: Q-learning, SARSA, EV-SARSA, and DDQN. DDQN appears to be the most suitable algorithm in controlling the agent behavior in the non-deterministic environment. The DDQN algorithm is implemented in the C++ programming language. Developed implementation of the machine learning method is used to control the agent in the Zmeyka gaming application. Computational experiments are presented to study effectiveness of the developed machine learning method in controlling the agent behavior. Experiments demonstrate the benefits of using DDQN under the changing environmental conditions, which confirms the algorithm effectiveness in solving problems of the agent behavior control.

Keywords

Artificial neural network, machine learning methods, reinforcement learning, Double Deep Q-learning, optimization algorithm, agent control, hyper-parameters, learning rate

Received 15.11.2023

References

[1] Kozov A.V. Comparing the efficiency of some modifications of the evolutionary strategy algorithm. *Politekhnicheskiy molodezhnyy zhurnal*, 2018, no. 5 (22). (In Russ.). http://dx.doi.org/10.18698/2541-8009-2018-5-309

[2] Vorontsov K. *Matematicheskie metody obucheniya po pretsedentam (teoriya obuche-niya mashin)* [Mathematical methods of learning by precedent (theory of machine learning).]. URL: http://www.machinelearning.ru/wiki/images/6/6d/Voron-ML-1.pdf (accessed October 15, 2023).

[3] Sutton R.S., Barto A.G. *Reinforcement Learning: An Introduction*. London, MIT Press, 1998, pp. 1–11.

[4] Littman M.L. Markov decision processes. *International Encyclopedia of the Social and Behavioral Sciences*, 2012, pp. 573–575. http://doi.org/10.1016/b0-08-043076-7/00614-8

[5] Kuz'min V. Using neural networks in the Q-learning algorithm. *Transport and Telecommunication*, 2003, vol. 4, no. 1, pp. 74–86. (In Russ.).

[6] Melo F.S. *Convergence of Q-learning: a simple proof. Institute for Systems and Robotics*. URL: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.413.2350&rep=rep1&type=pdf (accessed October 15, 2023).

[7] Herrmann M. *RL 5: On-policy and off-policy algorithms*. University of Edinburgh, School of Informatics. URL: https://www.inf.ed.ac.uk/teaching/courses/rl/slides15/rl05.pdf (accessed October 15, 2023).
[8] Hasselt H. van, Guez A., Silver D. Deep Reinforcement Learning with Double Q-learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2015, vol. 30 (1). http://doi.org/10.1609/aaai.v30i1.10295

Fedotov M.A. — Student, Department of Computer-Aided Design Systems, Bauman Moscow State Technical University, Moscow, Russian Federation.

Chapaev A.Yu. — Student, Department of Computer-Aided Design Systems, Bauman Moscow State Technical University, Moscow, Russian Federation.

Academic advisor — Kozov A.V., Senior Lecturer, Department of Computer-Aided Design Systems, Bauman Moscow State Technical University, Moscow, Russian Federation. E-mail: alexey.kozov@bmstu.ru

Please cite this article in English as:
Fedotov M.A., Chapaev A.Yu. Applying the machine learning methods in the agent behavior control. *Politekhnicheskiy molodezhnyy zhurnal*, 2023, no. 11 (88). (In Russ.). http://dx.doi.org/10.18698/2541-8009-2023-11-952