Пример ВКР «SPbPU-student-thesis-template». Версия от 24 мая 2020 г.

Министерство науки и высшего образования Российской Федерации
Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий

Работа допущена к защите
Руководитель ОП
_____ В.Г. Пак
«_____» _____ 2020 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ ИССЛЕДОВАНИЕ ПРИМЕНЕНИЯ ТЕХНОЛОГИИ ОБУЧЕНИЯ С ПОДКРЕПЛЕНИЕМ В УПРАВЛЕНИИ МУЛЬТИАГЕНТНОЙ СИСТЕМОЙ В ИГРОВОЙ СРЕДЕ

по направлению подготовки XX.XX.XX Наименование направления подготовки
Направленность (профиль) XX.XX.XX_YY Наименование направленности (профиля) образовательной программы

Выполнил
студент гр. в3540203/70277 О.Ю. Григорьев

Руководитель
должность,
степень, звание[1] В.Г. Пак

Консультант[2]
должность, степень И.О. Фамилия

Консультант
по нормоконтролю[3] Ю.Д. Заковряшин

[1]Должность указывают сокращенно, учёную степень и звание — при наличии, а подразделения — аббревиатурами. «СПбПУ» и аббревиатуры институтов не добавляют.

[2]Оформляется по решению руководителя ОП или подразделения. Только 1 категория: «Консультант». В исключительных случаях можно указать «Научный консультант» (должен иметь степень). Без печати и заверения подписи.

[3]Обязателен, из числа ППС по решению руководителя ОП или подразделения. Должность и степень не указываются. Сведения помещаются в последнюю строчку по порядку. Рецензенты не указываются.

Санкт-Петербург
2020

САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО

Институт компьютерных наук и технологий

УТВЕРЖДАЮ
Руководитель ОП
_____ В.Г. Пак
«_____» _____ 2020г.

ЗАДАНИЕ
на выполнение выпускной квалификационной работы

студенту Григорьеву Олегу Юрьевичу гр. в3540203/70277

1. Тема работы: Исследование применения технологии обучения с подкреплением в управлении мультиагентной системой в игровой среде.
2. Срок сдачи студентом законченной работы[4]: 05.2020.
3. Исходные данные по работе[5]:
   3.1. Фреймворки для разработки алгоритмов обучения с подкреплением (pytorch и другие).
   3.2. Программные среды тестирования алгоритмов (OpenAI Gym и другие).
   3.3. Алгоритмы обучения с подкреплением.
   3.4. Теория управления MAC.
4. Содержание работы (перечень подлежащих разработке вопросов):
   4.1. Обзор технологии обучения с подкреплением.
   4.2. Обзор подходов к управлению мультиагентными системами (MAC).
   4.3. Обзор аналогичных исследований.
   4.4. Постановка задачи выпускной квалификационной работы.
   4.5. Разработка алгоритмов управления агентами в MAC. Их исследование.
   4.6. Внедрение разработанных алгоритмов в компьютерную игру, тестирование.

[4]Определяется руководителем ОП, но не позднее последнего числа преддипломной практики и/или не позднее, чем за 20 дней до защиты в силу п. 6.1. «Порядка обеспечения самостоятельности выполнения письменных работ и проверки письменных работ на объем заимствований».

[5]Текст, который подчёркнут и/или выделен в отдельные элементы нумерационного списка, приведён в качестве примера.

4.7. Экспериментальное исследование алгоритмов.

4.8. Заключение, выводы об эффективности исследованных алгоритмов.

5. Перечень графического материала (с указанием обязательных чертежей):

5.1. Графики результатов экспериментов.

5.2. Блок-схемы алгоритмов.

5.3. Скриншоты

6. Консультанты по работе[6]:

6.1. Старший преподаватель ВШИСиСТ, Ю.Д. Заковряшин (нормоконтроль).

7. Дата выдачи задания[7]: 03.02.2020.

Руководитель ВКР _____ В.Г. Пак

Задание принял к исполнению 03.02.2020

Студент _____ О.Ю. Григорьев

[6]Подпись консультанта по нормоконтролю пока не требуется. Назначается всем по умолчанию.

[7]Не позднее 3 месяцев до защиты (утверждение тем ВКР по университету) или первого числа преддипломной практики или по решению руководителя ОП или подразделения (открытый вопрос).

РЕФЕРАТ

На 47 с., 14 рисунков, 1 таблицу, 0 приложений.

КЛЮЧЕВЫЕ СЛОВА: СТИЛЕВОЕ ОФОРМЛЕНИЕ САЙТА, УПРАВЛЕНИЕ КОНТЕНТОМ, PHP, MYSQL, АРХИТЕКТУРА СИСТЕМЫ[8]

В данной работе изложена сущность подхода к созданию динамического информационного портала на основе использования открытых технологий Apache, MySQL и PHP. Даны общие понятия и классификация IT-систем такого класса. Проведен анализ систем-прототипов. Изучена технология создания указанного класса информационных систем. Разработана конкретная программная реализация динамического информационного портала на примере портала выбранной тематики.[9]

В данной работе изложена сущность подхода к созданию динамического информационного портала на основе использования открытых технологий Apache, MySQL и PHP. Даны общие понятия и классификация IT-систем такого класса. Проведен анализ систем-прототипов. Изучена технология создания указанного класса информационных систем. Разработана конкретная программная реализация динамического информационного портала на примере портала выбранной тематики.

ABSTRACT

On 47 p., 14 figures, 1 tables, 0 appendices.

KEYWORDS: STYLE REGISTRATION, CONTENT MANAGEMENT, PHP, MYSQL, SYSTEM ARCHITECTURE

In the given work the essence of the approach to creation of a dynamic information portal on the basis of use of open technologies Apache, MySQL and PHP is stated. The general concepts and classification of IT-systems of such class are given. The analysis of systems-prototypes is lead. The technology of creation of the specified class

[8]Всего **слов**: от 3 до 15. Всего **слов и словосочетаний**: от 3 до 5. Оформляются в именительном падеже множественного числа (или в единственном числе, если нет другой формы), оформленных по правилам русского языка.

[9]До 600 печатных знаков (ГОСТ Р 7.0.99-2018 СИБИД) на русский или английский текст. Текст реферата повторён дважды на русском и английском языке для демонстрации подхода к нумерации страниц. *Внимание! Эта сноска размещена после точки. Это пример как не нужно оформлять сноску.*

of information systems is investigated. Concrete program realization of a dynamic information portal on an example of a portal of the chosen subjects is developed.

In the given work the essence of the approach to creation of a dynamic information portal on the basis of use of open technologies Apache, MySQL and PHP is stated. The general concepts and classification of IT-systems of such class are given. The analysis of systems-prototypes is lead. The technology of creation of the specified class of information systems is investigated. Concrete program realization of a dynamic information portal on an example of a portal of the chosen subjects is developed.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В этой работе исследуется совместная работа нескольких агентов с использованием алгоритмов и методов глубокого обучения в 2D игровых средах. Имея это в виду, мы изучили и адаптировали современные алгоритмы и методы обучения глубокого обучения с подкреплением к настройкам игры с несколькими агентами.

Глубокое обучение с подкреплением - это новая область исследований алгоритмов и методов, которая сочетает в себе обучение с подкреплением и глубокое обучение. Предыдущие работы в основном были направлены на адаптацию глубоких нейронных сетей для усиления алгоритмов обучения. Например, глубокая Q-сеть (Deep Q-network, DQN) [18] интегрирует глубокие нейронные сети в Q-обучение, классический табличный алгоритм обучения с подкреплением. Обученные сети могут играть в различные игры Atari 2600 [36] лучше человека. Это считается первой успешной попыткой научиться играть в видеоигры с прямым визуальным вводом большого размера. Алгоритм глубокого детерминированного градиента политики (deep deterministic policy gradient, DDPG) [6] является еще одним примером использования глубоких нейронных сетей в контексте обучения с подкреплением и непрерывного пространства действий.

Большинство исследований посвящено обучению одного агента. Однако проблемы, связанные с мультиагентным сотрудничеством или конкуренцией, также очень распространены в социальной, экономической и инженерной областях. Игры представляют собой упрощенные версии реальных задач, которые можно сделать идеальными тестовыми площадками для экспериментов. Поэтому в этой работе игровые сценарии используются для изучения алгоритмов глубокого обучения с подкреплением для совместной работы нескольких агентов.

0.1. Вопросы исследования

В этой работе мы исследуем следующие вопросы:

1. Как несколько агентов могут научиться сотрудничать друг с другом во время обучения в определенных игровых сценариях?
2. Может ли после обучения появиться язык между агентами в определенных игровых сценариях?
3. Как можно оптимизировать и ускорить процесс обучения?

Сначала мы изучим современные алгоритмы и методы глубокого обучения с подкреплением. Затем мы проведем эксперименты игровых сценариев в модифицированной среде от компании OpenAI [26]. Наконец, мы рассмотрим и применим различные приемы для оптимизации процесса обучения.

0.2. Сценарии

Чтобы исследовать вышеизложенные вопросы, мы используем несколько сценариев, где агенты общаются друг с другом и физически перемещаются к определенным целям. Некоторые из сценариев являются точными копиями экспериментов, проведенных в недавней работе [24], а другие представляют собой новые сценарии, которые расширяют исходные с целью дальнейшего изучения многоагентного сотрудничества и коммуникации.

0.2.1. Сценарий 1: Simple Speaker Listener

В этом сценарии есть три ориентира, представленные как красные, зеленые и синие ориентиры, как показано на рисунке. Два агента с разными функциями должны сотрудничать для достижения общей цели. *Говорун* (серый агент) не может двигаться, но видит цвет цели и может говорить с другим агентом. *Слушатель* (отображаемый таким же цветом, что и его цель) видит все ориентиры и их цвета (но не видит собственный цвет, т.е. не знает, какой из объектов является его целью), а так же слышит *говоруна* и пытается перейти к правильному ориентиру. Более подробные настройки среды этого сценария можно найти в разделе 4,1.

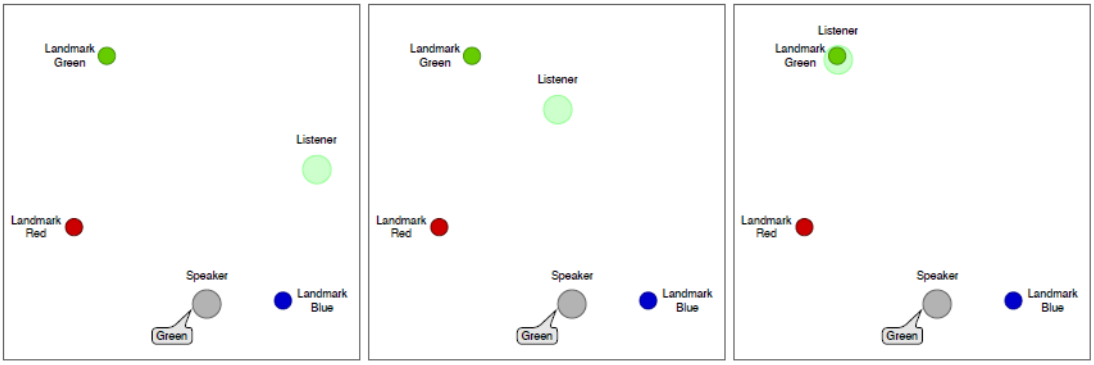

Рис.0.1. Сценарий 1. *Simple Speaker Listener*. Скриншоты слева направо показывают этапы игрового эпизода. *Говорун* (серый) выдает коммуникационное действие, представляющее «зеленый», а *слушатель*, слушает сообщение *говоруна* и направляется к цели.

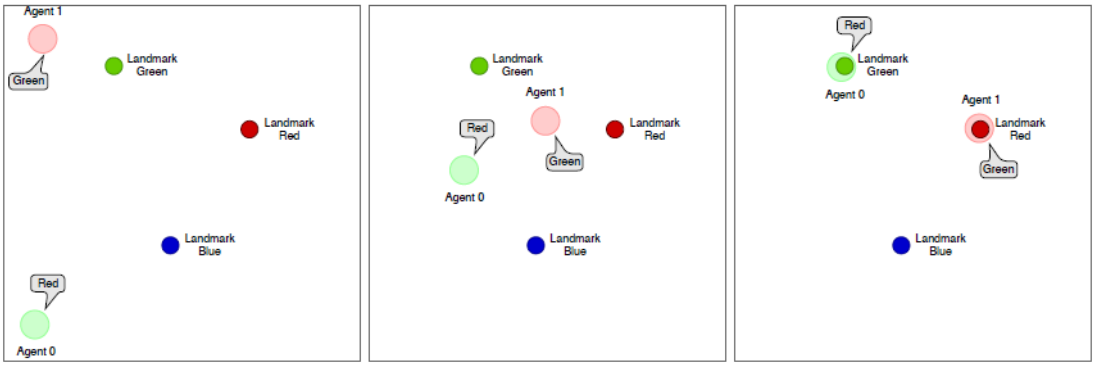

Рис.0.2. Сценарий 2: *Simple Reference*. В этом эпизоде агент 0 (отображаемый в том же цвете, что и его целевой ориентир) выдает коммуникационное действие, представляющее «красный», и слушает агента 1. А агент 1 (отображается в том же цвете, что и его целевой ориентир), слушает и издает «зеленый» для Агента 0. Скриншоты слева направо показывают, как ведут себя два агента в соответствии с оптимальными политиками, слушая друг друга и перемещаясь к целям.

0.2.2. Сценарий 2: Simple Reference

Этот сценарий расширяет предыдущий, поскольку оба агента являются одновременно и *говорунами* и *слушателями*. Ориентиры остаются прежними, отображаются как три разноцветных объекта, (как изображено на рис.0.2). Каждый из двух агентов пытается достичь своего целевого ориентира, который известен только другому агенту. Таким образом, он должен научиться сообщать другому агенту его цель и перемещаться к своей собственной. Что отличает сценарий от двух копий сценария *Simple Speaker Listener*, так это то, что единое общее вознаграждение, присуждаемое агентам, основано на общей производительности. Таким образом, агенты должны выяснить, что идет хорошо, а что нет. Настройка среды подробно описана в разделе X.X.

0.3. Практическая значимость

Проект сосредоточен на обучении нескольких агентов совместной работе с использованием глубокого обучения в игровой среде. Результаты исследования могут быть дополнительно разработаны и широко использованы во многих практических реальных приложениях в области экономики, управления, техники и т. д. Например, он может применяться в робототехнике, к автономным транспортным средствам, производственным линиям, фондовым рынкам и т. д.

Перед применением в этих областях, алгоритмы и методы, которые исследуются и разрабатываются в этой работе, должны быть хорошо протестированы и проверены, поскольку эти приложения непосредственно влияют на безопасность

человека, социальную и финансовую безопасность. В долгосрочной перспективе применение мультиагентных алгоритмов приведет к появлению все большего числа автономных систем во многих областях.

ГЛАВА 1. ОБЗОР ТЕХНОЛОГИИ ОБУЧЕНИЯ С ПОДКРЕПЛЕНИЕМ.

1.1. Введение.

Алгоритмы и методы глубокого обучения с подкреплением (Deep Reinforcement Learning, DRL) – это методы, которые мы используем для исследования вопросов дипломной работы. DRL – это подраздел машинного обучения, который лежит на пересечении Обучения с подкреплением (Reinforcement Learning) и глубокого обучения (Deep Learning).

Прежде чем говорить о DRL, нужно кратко рассмотреть понятия, связанные с искусственным интеллектом и машинным обучением.

На рис.1.1 представлен обзор концепций, которые мы вводим в этой главе.

Сначала мы рассмотрим искусственный интеллект, машинное обучение и его три категории. Затем мы углубимся в глубокое обучение. Наконец, мы рассмотрим основные алгоритмы DRL. Таким образом, мы можем понять, почему и как DL и RL объединяются в DRL, который позволяет агентам взаимодействовать с более сложной средой и вести себя более разумно.

Рис.1.1. Граф, показывающий концепты МО и ИИ и их отношения.

1.2. Искусственный Интеллект

Искусственный интеллект [8] (ИИ), как следует из названия, в отличие от естественного интеллекта человека и других животных, является искусственным. Именно такой тип интеллекта люди реализуют в машинах. Конечной целью ИИ является создание таких автономных систем, которые способны учиться

методом многочисленных проб и ошибок, чтобы найти оптимальное поведение для достижения максимально возможных целей в сложившемся окружении. [28]

С 21 века, благодаря нескольким прорывам, ИИ процветает в викторинах и настольных играх, превосходя уровень игры людей [37] [1]. Наряду с увеличением вычислительной мощности, улучшения алгоритмов и доступности больших наборов данных, ИИ развивался революционными темпами. В ближайшем будущем, ИИ ещё сильнее повлияет на работу и повседневную жизнь человека.

1.3. Машинное Обучение

Если ИИ – это более широкая концепция машинного интеллекта для выполнения задач, которые пока выполняют люди, то Машинное обучение (ML) - это основной метод разработки ИИ, который не требует явного программирования [29]. ML позволяет компьютерам строить модели и применять алгоритмы при изучение больших объемов данных. Модели ML обучаются с использованием методов статистики, чтобы понять структуру набора данных или последовательности экспериментов. Обученные модели могут распознавать паттерны и делать очень точные прогнозы учитывая не очевидные данные или обрабатывать определенные задачи в неочевидных сценариях [5].

Основные категории алгоритмов ML показаны на рисунке рис.1.2. А именно обучение с учителем, обучение без учителя и обучение с подкреплением. Категории определяются тем, как алгоритмы и модели наполняются данными и как данные анализируются.

1.3.1. Обучение с учителем (Supervised Learning)

TODO: может быть добавить рисунок

Один из способов машинного обучения, в ходе которого испытуемая система принудительно обучается с помощью примеров «стимул-реакция». С точки зрения кибернетики, является одним из видов кибернетического эксперимента. Между входами и эталонными выходами (стимул-реакция) может существовать некоторая зависимость, но она неизвестна. Известна только конечная совокупность прецедентов — пар «стимул-реакция», называемая обучающей выборкой. На основе этих данных требуется восстановить зависимость (построить модель отношений

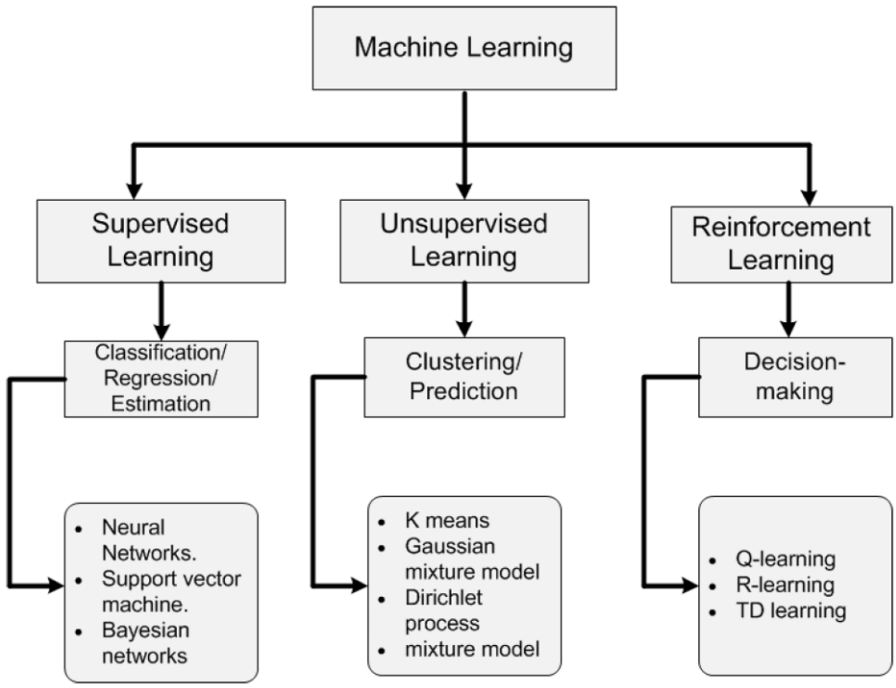

Рис.1.2. Категории ML и соответстующие алгоритмы. Основано на [30].

стимул-реакция, пригодных для прогнозирования), то есть построить алгоритм, способный для любого объекта выдать достаточно точный ответ [2].

Обучение с учителем можно разделить на *регрессию* и *классификацию*, в зависимости от того, являются ли выходные переменные количественными или качественный. Количественные переменные принимают числовые значения, в то время как качественные переменные принимают значения в одном из K различных классов или категории [2]. Например, прогноз цены на жилье по данным параметрам, таким как местоположение дома, общая площадь и количество комнат и т. д., это проблема регрессии. Диагноз рака - проблема классификации поскольку выходной сигнал либо положительный, либо отрицательный.

В обучении с учителем набор данных делится на обучающий, набор для валидации и тестовый набор. Обучающий набор представляет собой пары ввода и вывода переменных, которые напрямую вводятся в модель для обучения. Валидационный набор используется для контроля за переобучением модели. Наконец, тестовый набор используется для подтверждения того, что обученная модель обобщена и точна

Обучение с учителем - наиболее распространенная категория в машинном обучении, но оно требует больших наборов данных с правильными «ответами». Это может быть очень дорого, в некоторых случаях не практично.

1.3.2. Обучение без учителя (Unsupervised Learning)

TODO: может быть добавить рисунок

Это ещё одна важная категория в машинном обучении. В отличие от обучения с учителем, алгоритмы этой категории используют наборы данных, не размеченные «ответами». Задача в этой категории — обнаружить скрытую структуру в данных и распределить их по группам.

Эта категория алгоритмов получила своё название из-за отсутствие меток или выходных переменных. Кластеризация является типичным инструментом, который используется чтобы понять связь между наблюдениями и распределить их в разные группы [15]

При обучении без учителя данные не делятся на обучающие, проверочные и тестовые. Набор данных подается в модель напрямую и группируется в отдельные группы.

1.3.3. Обучение с подкреплением (Reinforcement Learning)

Последняя категория машинного обучения является междисциплинарной областью, которая сочетает в себе машинное обучение, неврологию, поведенческую психология, теорию управления и т. д. Цель RL заключается в достижении целей без четких инструкций, но с наградами или штрафами, получаемыми от взаимодействий с окружающей средой.

Агент RL изучает оптимальную политику, последовательность действий, которая максимизируют общую будущую награду (Reward) [31].

В обучении с подкреплением агент RL наблюдает состояние s_t на этапе времени t, затем он взаимодействует с окружающей средой, выполняя действие a_t. Среда переходит в следующее состояние s_{t+1}, учитывая текущее состояние и выбранное действие, которое ведёт к получению агентом вознаграждения r_t. Цель агента узнать политику π, которая сопоставляет состояния с действиями, так что последовательность действий, выбранная агентом, максимизирует ожидаемое будущее вознаграждение. На каждом шаге взаимодействия со средой агент генерирует переход $\{s_t, a_t, s_{t+1}, r_t\}$, который даёт информацию, необходимую для улучшения политики, см. рис.1.3.

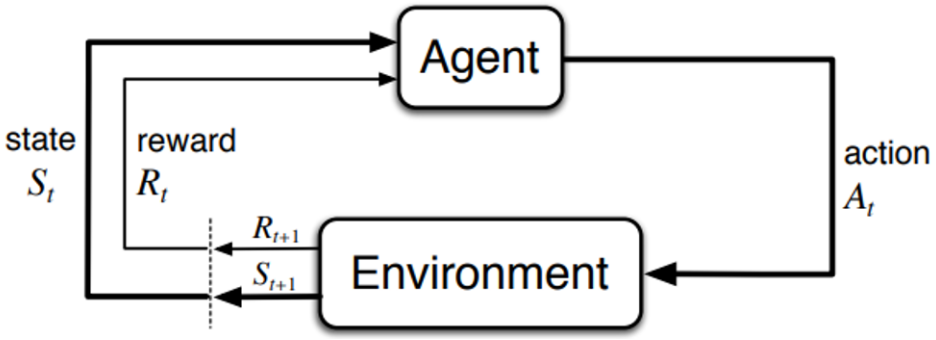

Рис.1.3. Агент взаимодействует с окружающей средой, сначала наблюдая состояние, затем совершая действие и, наконец, получая вознаграждения за выбранное действия. Через многочисленные попытки и ошибки, агент учится формулировать оптимальную политику [32].

1.4. Глубокое обучение

Глубокое обучение (Deep Learning, DL) - подраздел машинного обучения, в настоящее время достигает выдающихся результатов во многих областях, в том числе распознавание изображений, компьютерное зрение, распознавание речи и естественное языковая обработка. Глубокое обучение, как бы имитирует биологический мозг, обрабатывает информацию с помощью искусственных нейронных сетей.

Идея симуляции работы нейронов мозга человека зародилась десятилетия назад. Тем не менее, прорыв произошел в последние годы, когда стали доступны большие наборы данных и вычислительные мощности [12]. Теперь можно строить модели с большим количеством слоев искусственных нейронов, чем когда-либо прежде. С сильным увеличением глубины, сети достигают исключительной производительности в области распознавания изображений и речи. Считается, что глубокое обучение является одним из наиболее перспективных подходов для решения текущих задач ИИ.

1.4.1. Искусственная нейронная сеть

Мозг человека и животных - чрезвычайно сложный орган, который до сих пор до конца не изучен. Тем не менее, некоторые аспекты его структуры и функции были расшифрованы. Фундаментальным рабочим элементом в мозге является нейрон. Многочисленные нейроны связаны между собой сложным образом, что даёт возможность, запоминания, мышления и принятия решений.

Хотя нейроны сложны и функционируют разными способами, все они имеют некоторые базовые компоненты, такие как: ядро, дендриты, аксон и синапсы.

Дендриты действуют как входные каналы, через которые нейроны получают информацию от синапсов других нейронов. Затем ядро обрабатывает эти сигналы и превращает в вывод, который затем отправляется в другие нейроны. Связь этих компонентов играет роль линии передачи в нейронных сетях.

Хоть ИНС и не так сложны, как человеческий мозг, они имитируют базовую структуру, которая включает входные, выходные слои, а также, обычно, скрытые слои. В каждом слое есть искусственные нейроны. Нейроны в одном слое обычно связаны с каждым нейроном в следующем слое. Они передают числовые сигналы через соединения с другими нейронами, примерно, как это происходит в биологической нейронной сети [17].

Поскольку выходные сигналы нейронов ИНС представляются в виде вещественных чисел, выход обычно сравнивается с порогом. Только если порог превышен, нейрон передаёт сигнал следующим подключенным нейронам.

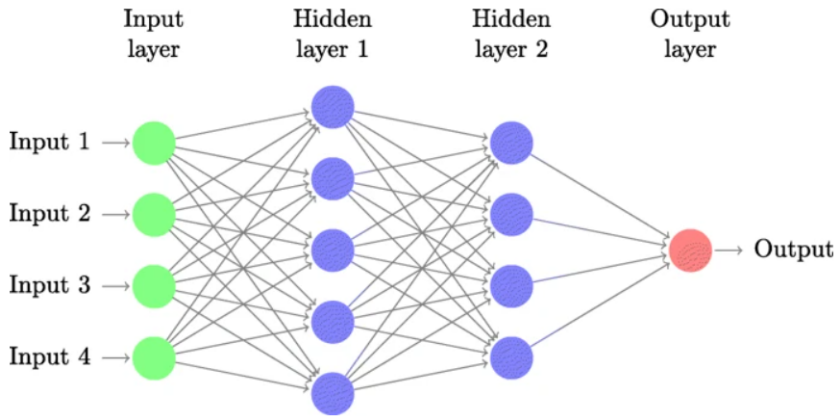

Рис.1.4. Базовая структура ИНС. ИНС обычно состоит из входного и выходного слоев, а также пары скрытых слоев. Основано на [21] [22].

Связи между нейронами параметризованы весами, которые обновляются во время обучения, чтобы отрегулировать мощность сигналов, проходящих через сеть [24]. рис.1.4 показывает базовую структуру ИНС.

1.4.2. Глубокая нейронная сеть

Как уже упоминалось выше, концепция нейронной сети не нова.

На ранних этапах из-за ограничений вычислительной мощности, нейронные сети имели очень малую глубину. Обычно он содержит только входной, выходной слои и пару скрытых слоев. Кроме того, количество нейронов в каждом слое также было ограничено. Глубокие нейронные сети не находили практического

применения до последних лет, когда стали доступны огромные вычислительные мощности и большие объемы данных.

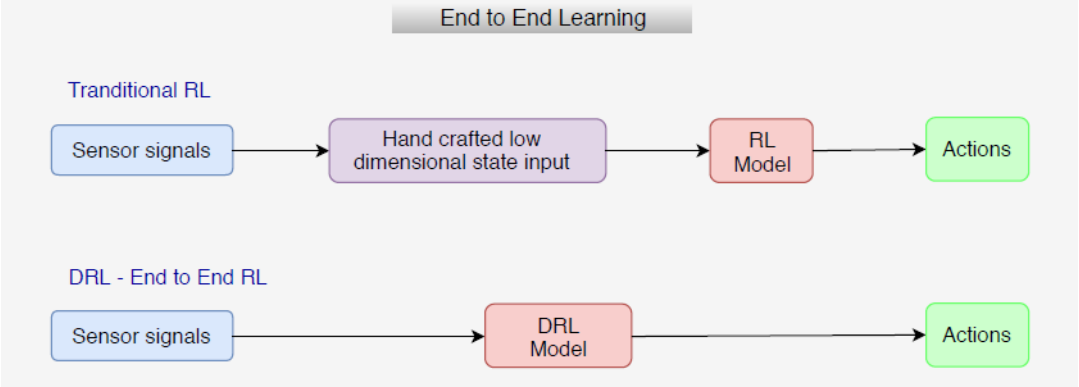

Рис.1.5. Сравнение DRL и традиционной RL, где необходимо явно указать, какие действия производить в каких состояниях. Использование глубокой нейронной сети в DRL позволяет обучаться и принимать решения на основе необработанного сенсорного ввода.

Глубокие нейронные сети с несколькими слоями хороши для извлечения скрытых свойств [20]. Каждый слой, решает свою собственную задачу. Узлы в каждом слое учатся на конкретных наборах свойств, которые приходят из предыдущего слоя. Теоретически чем глубже нейронная сеть, тем более сложны и абстрактны особенности, которые она может распознать, поскольку каждый нейрон агрегирует выводы из нейронов предыдущего слоя [4]. Например, в задаче распознавания изображений вход представляет собой матрицу пикселей. Первый слой извлекает начальные объекты, такие как ребра из пикселей, затем следующий слой кодирует расположение краев и следующий слой распознает глаза, рот, уши, ноги, крылья, хвост и т. д. Наконец, последний слой распознает на изображении кошку, собаку или птицу. Таким образом, глубокие нейронные сети не нуждаются во вмешательстве людей, а сами изучают иерархию объектов.

С математической точки зрения нейронная сеть определяет функцию $y = f(x; \theta)$. Она описывает соответствие входных данных x выходным y, где y — это категория в задаче классификации или выходное значение в задачах регрессии. Тренировка модели с использованием набора данных ведет к вычислению параметра θ.

После завершения обучения предполагается, что нейронная сеть аппроксимирует целевую функцию f^* [14]. Правильно обученная ИНС может лучше соответствовать набору данных, а также делает прогноз, учитывая не очевидные зависимости от данных.

Глубокие нейронные сети могут быть реализованы по-разному в зависимости от конкретных практических задач. Например, свёрточные нейронные сети специализируются на компьютерном зрении, а рекуррентные нейронных сети лучше подходят для обработки естественного языка.

Функцию отображения $f(x)$ можно рассматривать как цепочку из многих связанных функции в виде $f(x) = f^{(n)}(f^{(n-1)}(...f^{(2)}(f^{(1)}(x))))$. n связанных функций соответствуют глубине ИНС. Функция стоимости определяется на основе сравнения вывода y из $f(x)$ с целевым значением t из тренировочного набора данных. Цель обучения нейронной сети в том, чтобы приблизить функцию $f(x)$ к такой, чтобы минимизировать функцию стоимости. Минимизация функции стоимости является задачей оптимизации, и для этого часто используют алгоритм *градиентного спуска*. В *обратном распространении* (backpropagation), градиент используется для итеративного обновления нейронной сети во время обучения. Оптимизатор решает, какие параметры должны быть обновлены, а *скорость обучения* (learning rate) задает размер шага, на который параметр обновляется на каждой итерации.

1.5. Обучение с подкреплением

Хотя алгоритмы RL эффективно решают различные задачи, им не хватает масштабируемости и размерности.

С ростом глубоких нейронных сетей в последние годы, RL начинает использовать их функции приближения и представления свойств [16]. Это помогает преодолеть недостатки алгоритмов RL.

Это устраняет необходимость описывать свойства вручную, позволяя обучать модели, способные непосредственно выводить оптимальные действия, на основе необработанного и высокоразмерного ввода с сенсоров. Это проиллюстрировано на рис.1.5.

Таким образом, использование ИНС в обучении подкреплением, создает новую область – глубокое обучение с подкреплением (Deep Reinforcement Learning, DRL).

1.5.1. Алгоритмы глубокого обучения с подкреплением

Здесь нужно рассказать о Марковском процессе принятия решений (Markov Decision Process, MDP).

1.5.1.1. Марковский процесс принятия решений

Свойство Маркова означает, что следующее состояние зависит только от текущего состояния, тем самым, при принятии решения, мы можем игнорировать все прошлые состояния и учитывать только текущее.

Процесс *обучения с подкреплением* является формой МППР. Он состоит из нескольких элементов:

- набор состояний окружающей среды S;
- набор действий A;
- динамика перехода $T(s_{(t+1)}|s_t, a_t)$, которая описывает распределение новых состояний s_{t+1}, в которые может попасть агент, совершив действие a_t в состоянии s_t;
- функция награды $R(s_t; a_t; s_{t+1})$
- дискаунт фактор $\gamma \in [0, 1]$ для экспоненциального снижения будущих наград.

Политика π сопоставляет состояния вероятностям распределения действий: $\pi : S \rightarrow p(A = a|S)$

Эпизод — это предопределенный период времени, когда среда, начиная со случайного состояния, порождает серию переходов. Переходы в эпизоде можно рассматривать как траекторию политики.

Сумма наград, собранных в траектории политики $R = \sum_{t=0}^{T-1} \gamma^t r_{t+1}$

Цель обучения с подкреплением состоит в том, чтобы изучить оптимальную политику π^*, которая максимизирует ожидаемую награду из всех состояний, $\pi^* = argmax_x \mathbb{E}[R|\pi]$ [27]

1.5.2. Подход, основанный на функции состояния (Value Based подход)

Этот подход состоит в том, чтобы оптимизировать значение функции $V(s)$.

value-функция — это функция, которая сообщает нам максимальное ожидаемое будущее вознаграждение, которое агент получит в каждом состоянии.

Значение каждого состояния — это общая сумма вознаграждений, на которую агент может рассчитывать в будущем, начиная с этого состояния.

$$V^{\pi}(s) = \mathbb{E}^{\pi}[R_{t+1} + \gamma Rt + 2 + \gamma^2 Rt + 3 + ...|S_t = s] \tag{1.1}$$

Q-обучение Функция состояния-значения (state-value funcrion) $V^{\pi}(s) = \mathbb{E}[R|s, \pi]$ — ожидаемая награда с данными состоянием s и политикой π. В то же время, в RL такой переход T не всегда возможен, поэтому, обычно используется другая функция состояния-значения или функция качества $Q^{\pi}(s,a) = \mathbb{E}[R|s, a, \pi]$. Оптимальная политика получается через выбор на каждом шаге действия, которое максимизирует Q-функцию [33]. Q-обучение – это алгоритм без политики (off–policy), так как он жадно выбирает действие исходя из текущего состояния, вместо того, чтобы следовать политике.

Для рекурсивного изучения Q^{π}, применяется *уравнение Беллмана*:

$$Q^{\pi}(s_t, a_t) = \mathbb{E}_{s_{t+1}}[r_{t+1} + \gamma Q^{\pi}(s_{t+1}, \pi(s_{t+1}))] \tag{1.2}$$

Традиционное Q-обучение способно сформулировать оптимальную политику обучением состояние-действие-значение функции. Однако, оно предназначено для дискретных пространств действий с малым количеством измерений и не способно решать более-менее сложные задачи.

Глубокая Q-сеть (DQN) Глубокая Q-сеть (DQN) — вариант Q-обучения, который использует глубокую сверточную нейронную сеть для вычисления Q-функции. Является прорывом в обучении с подкреплением. [11]

DQN была применена в игре Atari и достигла производительности, сопоставимой с человеческим уровнем. [18]

Для решения проблемы нестабильности и расхождения нелинейных функций аппроксиматоров, таких как нейронные сети, был применен метод *Experience Replay* [17]. Идея Experience Replay в том, чтобы равномерно рандомизировать предыдущие переходы при обучении модели, которое нарушает корреляцию последовательности наблюдений. Опыт агента $e_t = (s_t, a_t, r_t, s_{t+1)}$ на каждом шаге t сохраняется в буфер $D_t = e_1, ..., e_t$. Во время тренировки модели случайным образом извлекается небольшая часть опыта $(s; a; r; s')$ $U(D)$.

Двойная глубокая Q-сеть (Double DQN, DDQN) Это ещё один вариант DQN, в котором используется две Q-сети.

Текущая Q-сеть $Q(s, a; \theta_i)$ обновляется итеративно во время обучения, а целевая Q-сеть $Q(s', a'; \theta_i^-)$ используется для получения целевого Q-значения и обновляется только периодически. Целевая Q-сеть уменьшает смещение, вызванное неточностями Q-сети в начале обучения.

На каждой итерации i Q-сеть обновляется на ошибку темпоральной разницы (temporal difference, TD):

$$L_i(\theta_i) = \mathbb{E}^{\pi}[R_{t+1} + \gamma Rt + 2 + \gamma^2 Rt + 3 + ...|S_t = s] \tag{1.3}$$

DQN подошел к проблеме низкоразмерных входных данных наблюдения с применением глубоких нейронных сетей для извлечения свойств из высокомерного необработанного сенсорного сигнала, такого как пиксели изображения в играх. Тем не менее, он все еще ограничен его дискретным и низкоразмерным пространством действия.

1.5.3. Линия поведения (Policy Based)

Вместо получения оптимальной политики путем поддержания Q-функции, алгоритм напрямую ищет оптимальную политику путем максимизации ожидаемого значения $\mathbb{E}[R|\pi_\theta]$. Мы хотим итеративно подгонять параметр θ сети, так чтобы максимизировать $\mathbb{E}[R|\pi_\theta]$.

Оптимизация, основанная на градиенте, используется чаще, так как это более эффективно, при работе с большими сетями со множеством параметров. [10]

1.5.3.1. Policy Gradients

(TODO: перевод?)

Нейронная сеть, которая представляет параметризованную политику обновляется при изучении сигналов в методах Policy Gradients. В RL без модели, для оценки градиента на примерах, сгенерированных в траектории политикой, используется правило *REINFORCE* или функция оценки. Предположим, что f(x) – это функция оценки, где x - случайная величина для одного перехода. Градиент политики может быть рассчитан с использованием отношения правдоподобия:

$$\nabla_\theta \mathbb{E}_x[f(x)] = \mathbb{E}_x[f(x)\nabla_\theta \log px|\theta] \tag{1.4}$$

Теперь рассмотрим траекторию τ с переходами (a_t, s_t, r_t, s_t+1) в соответствии с политикой, тогда градиент политики это:

$$\nabla_\theta \mathbb{E}[R_\tau] = \mathbb{E}[\sum_\tau R_\tau \nabla_\theta \log \pi a_\tau | s_\tau; \theta] \tag{1.5}$$

Недостатком policy gradient является низкая скорость работы — требуется большое количество вычислений для подсчета награды. Так же policy gradient может застрять в локальном оптимуме, не найдя глобального.

1.5.3.2. Актор-критик

Так как value-функция может предоставить обучающие сигналы для прямого оптимального поиска политики, естественным было объединить два подхода.

В DRL две нейронные сети, представляющие актора и критика соответственно, используются для приближения функции, где актор (политика) учится по Q-значениям, оцененным критиком (value-функция) [10]. рис.1.6 показывает, как актор и критик сети взаимодействуют с окружающей средой.

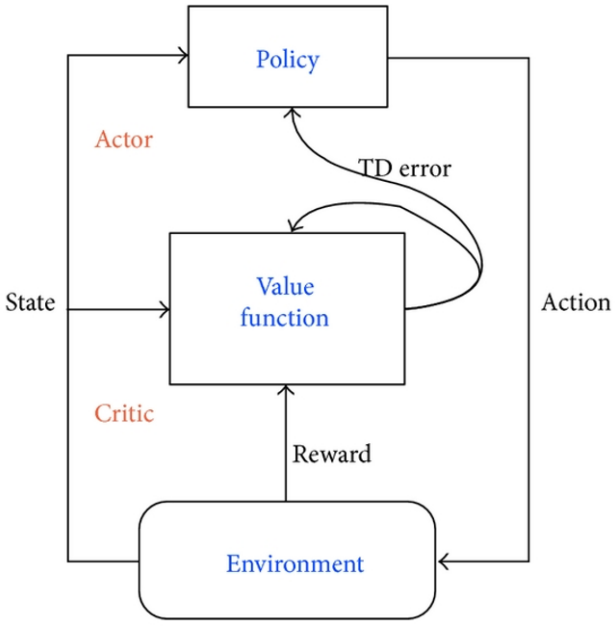

Рис.1.6. *Актор* получает состояние из окружающей среды и реагирует на действие, *критик* получает состояние и награду и рассчитывает TD ошибку для обновления себя и *актора*. Основано на [10]

1.5.3.3. Глубокий детерминированный градиент политики (DDPG)

DDPG - актор-критик алгоритм без модели и без политики [6]. Он расширяет DPG использованием глубоких нейронных сетей. Так же, он использует хорошо за-

рекомендовавший себя приём DQN с текущей и целевой Q-сетями (сети критиков) и experience replay, чтобы стабилизировать обучение. И, наконец, он использует детерминистическую политику (сеть акторов) вместо политики стохастического поведения. В отличие от стохастической политики, которая определяет вероятность распределения через действия, в данном состоянии, детерминированность в DDPG подразумевает, что конкретное действие аппроксимируется в данном состоянии. Соответственно, конкретное состояние для следующего шага, так же детерминировано. Следовательно, вместо использования рекурсивного программирования, как в уравнении *Беллмана*, используется детерминированная политика $\mu : S \leftarrow A$ [6]

$$Q^{\mu}(s_t, a_t) = \mathbb{E}_{r_t, s_{t+1}\ E}[r(s_t, a_t) + \gamma Q^{\mu}(s_{t+1}, \mu(s_{t+1})] \tag{1.6}$$

что очень похоже на Q-обучение – алгоритм с жадной политикой. Сеть критика обновляется по функции потерь:

$$L(\theta^{Q}) = \mathbb{E}[(Q(s_t, a_t|\theta^{Q}) - y_t)^2] \tag{1.7}$$

где

$$y_t = r(s_t, a_t) + \gamma Q(s_{t+1}, \mu(s_{t+1}) j \theta^{Q}) \tag{1.8}$$

Сеть актора обновляется функцией потерь:

$$\begin{aligned} \nabla_{\theta\mu} J &\approx \mathbb{E}[\nabla_{\theta\mu} Q(s, a\theta^{Q})|_{s=s_t, a=\mu(s_t|\theta^{\mu})}] \\ &= \mathbb{E}[\nabla_a Q(s, a\theta^{Q})|_{s=s_t, a=\mu(s_t)\nabla_{\theta\mu}\mu(s|\theta^{\mu})|s=s_t}] \end{aligned} \tag{1.9}$$

Как и в DQN, чтобы избежать расхождения, DDPG применяет мягкое обновление для целевых сетей критика и актора. Они обновляются только раз в указанное количество шагов.

ГЛАВА 2. ОБУЧЕНИЕ МУЛЬТИАГЕНТНЫХ СИСТЕМ

2.1. Введение

В этой главе мы хотели бы представить соответствующую работу по теме «Мультиагентное глубокое обучение с подкреплением». Прежде всего, мы ссылаемся на современные алгоритмы, которые соответствуют нашим мультиагентным сценариям.

2.2. Мультиагентные алгоритмы

Методы обучения с подкреплением, которые специализируются на решении задач с одним агентом, плохо адаптированы для многих реальных задач, таких как автономные транспортные средства. Поэтому необходимо расширить эти алгоритмы или даже создать новые для более сложных сценариев с настройками для нескольких агентов. Некоторые алгоритмы, которые вдохновили эту работу, иллюстрируются в следующих разделах. Этими алгоритмами являются Multi Agent Deep Deterministic Policy Gradient [24], Counterfactual baseline for multi-agent policy gradient [7] и emergent grounded compositional language [23].

2.2.1. Детерминированная политика для нескольких агентов

Multi Agent Deep Deterministic Policy Gradient (MADDPG) - это расширение DDPG, применяемое к настройкам нескольких агентов. Чтобы учесть все состояния среды и политики всех агентов RL в игровом сценарии, алгоритм учитывает совместные наблюдения и действия всех агентов при обучении сетей актеров и критиков. Когда дело доходит до принятия решения, сеть актора каждого агента учитывает только локальные наблюдения. Эта структура централизованного обучения и децентрализованного исполнения позволяет каждому агенту изучать оптимальную политику по консистентному градиентному сигналу. [24]

Совместные наблюдения всех агентов обозначаются через $\mathbf{x} = (o_1, ..., o_N)$, совместные действия - через $a = (a_1, ..., a_N)$. Они вместе с наградами r хранятся в буфере D в виде $(\mathbf{x}, a, r, \mathbf{x}')$. Случайная выборка из S выборок $(\mathbf{x}^j, a^j, r^j, \mathbf{x}'^j)$ извлекается из D, а критик каждого агента обновляется путем минимизации потерь:

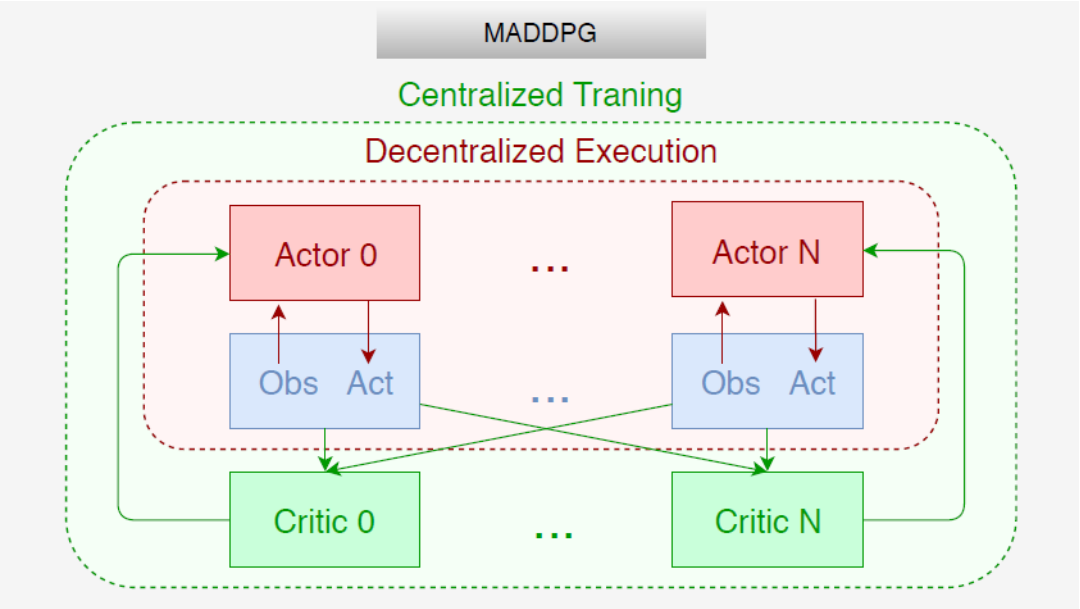

Рис.2.1. у каждого агента есть децентрализованная сеть акторов, которая получает доступ только к своим локальным наблюдениям. Между тем, каждый агент имеет централизованную сеть критиков, которые имеют доступ к наблюдениям и действиям всех агентов. Критик сети обучен обновлять себя и актора. Основано на [24].

$$L(\theta_i) = \frac{1}{S}\sum_j (y^j - Q_i^{\mu}(\mathbf{x}^j, a_1^j, ..., a_N^j))^2 \tag{2.1}$$

И актор обновляется семплированым градиентом политики:

$$\nabla_{\theta_i} J \approx \frac{1}{S}\sum_j \nabla_{\theta_i} \mu_i(o_i^j) \nabla_{a_i} Q_i^{\mu}(x^j, a_1^j, ..., a_i, ..., a_N^j)|_{a_i=\mu_i(o_i^j)} \tag{2.2}$$

Подобно DDPG, целевые сети обновляются «мягко» на каждом шаге на $\theta' \leftarrow \tau\theta + (1-\tau)\theta'$ с $\tau \ll 1$.

Алгоритм MADDPG хорошо подходит для сценариев, в которых агенты учатся взаимодействовать в одномерном пространстве действий, в основном с физической навигацией. Мы хотели бы продолжить изучение этого алгоритма с помощью многомерного пространства действий, особенно для сценариев, в которых агенты могут взаимодействовать при выполнении физических действий.

2.2.2. Counterfactual Multi-Agent Policy Gradient

TODO: перевести

Counterfactual Multi-Agent Policy Gradient (COMA) - это мультиагентный вариант метода *актор-критик*, который использует единого централизованного

критика для приближения к Q-функции и отдельных децентрализованных *акторов* для оптимизации политики каждого агента. [7]

В кооперативных задачах с несколькими агентами сложность сотрудничества возрастает с увеличением количества агентов. Таким образом, было бы нецелесообразно и неэффективно иметь одну единственную оптимальную политику для всех агентов. Вместо этого децентрализованные политики для каждого агента формулируются в виде проблем с несколькими агентами. В COMA один централизованный критик и отдельные акторы тренируются совместными действиями и совместными наблюдениями. Когда дело доходит до исполнения, каждый актор генерирует действия на основе собственной истории наблюдения, см. рис.2.2.

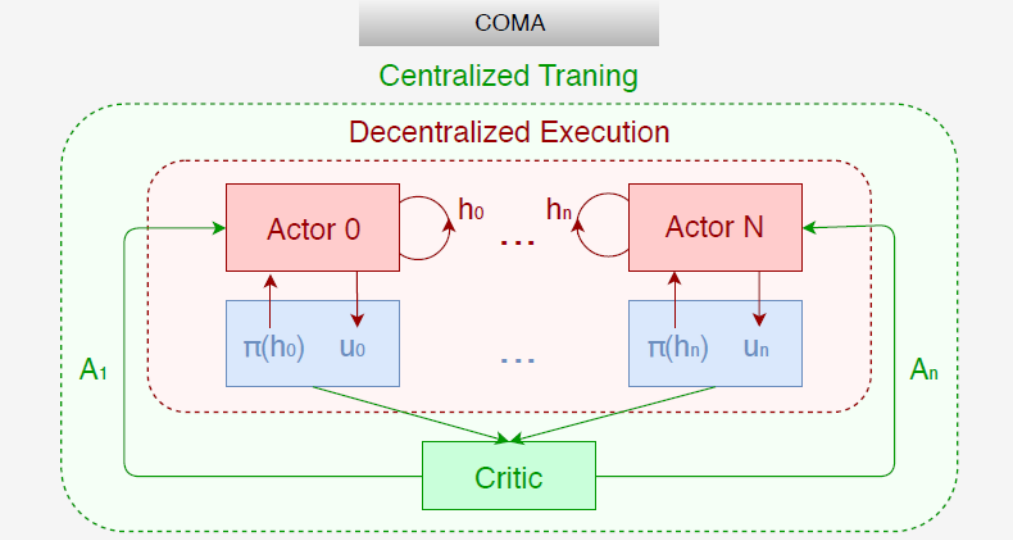

Рис.2.2. Структура сетей алгоритма COMA и информационный поток между централизованным критиком и децентрализованными акторами. В COMA отдельно происходит обучение одного централизованного критика и каждого из акторов. Каждый актор децентрализован, при принятии решений использует локальные истории наблюдения. [7]

COMA предназначен для мультиагентных совместных сценариев с глобальной функцией награды для всех агентов. Однако агенты, обученные COMA, изучают отдельные политики без явного общения. [7]

2.2.3. Emergent Language

Grounded compositional language обозначает простой язык, где агенты связывают конкретные символы с конкретными объектами, а затем объединяют эти символы в значимые понятия [34]. Язык представлен в виде абстрактных дискретных символов, произнесенных агентами, которые не имеют заранее определенного значения, но возникли и сформировались в процессе обучения в соответствии со

средой и целями [23]. В отличие от естественного языка, для работы с которым извлекаются языковые шаблоны из большого набора текстовых данных, этот язык, возникший во время обучения с подкреплением, понятен только агентам и используется ими для сотрудничества друг с другом в достижении общих целей.

2.3. Действия и награды

2.3.1. Архитектура ветвления действий (Action Branching)

Обычно довольно трудно исследовать проблемы с многомерными пространствами действий. Архитектура *ветвления действий* предназначена для решения таких проблем. Например, в среде с N-мерным пространством действий и d_n дискретных последующих действий для каждого измерения n необходимо рассмотреть в общей сложности $\Pi_{n=1}^{N} d_n$ возможных действий [35]. Правильно спроектированные разветвленные архитектуры действий могут эффективно и результативно исследовать такое большое многомерное пространство действий [35].

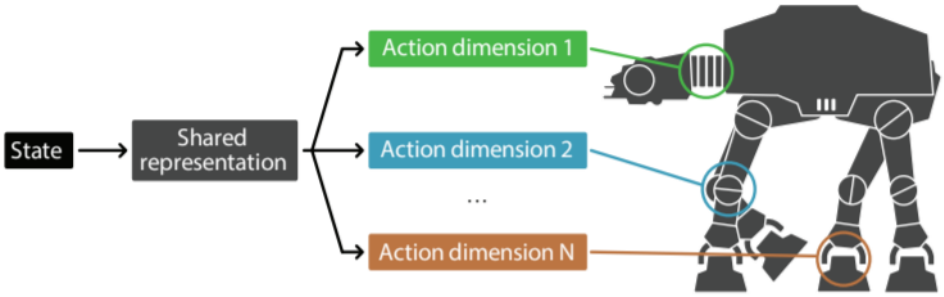

Рис.2.3. Архитектура сети для ветвления действий, например, для робота, где сеть принимает состояние в качестве входных данных и разделяет нижние уровни для извлечения представлений признаков, а затем расходится по ветвям для относительно независимых под-действий, включая, например, высказывание, действие рукой, действие ногой и т. д. Основано на [35].

Основной идеей архитектуры является модуль совместного принятия решений, который извлекает скрытое представление из входного наблюдения и создает отдельные выходные ветви для каждого измерения действия. На рисунке рис.2.3 представлена архитектура *ветвления действий*. n измерений действий представляют собой n относительно независимых под-действий. В архитектурах *ветвления действий* значения Q рассчитываются для каждого измерения действия. Тем не менее, мы бы хотели, чтобы многомерное действие оценивало одно единственное Q-значение, выводимое сетью критика.

2.3.2. Архитектура гибридного вознаграждения

Как показано в [19], жизненно важно иметь точную оптимальную value–функцию в обучении с подкреплением, поскольку она оценивает ожидаемый return как сигнал для оптимизации политики. Как только оптимальная value-функция изучена, из неё можно получить оптимальную политику.

Рис.2.4. Глубокая нейронная сеть с одной головой, аппроксимирует одну единственную Q-функцию с глобальной функцией награды (слева), и Q-сеть с гибридной архитектурой вознаграждения (справа). Сеть имеет n голов, аппроксимирующих n Q-функций. Каждая Q-функция оценивает Q-значение с помощью соответствующей разложенной функции награды. Основано на [19].

Более того, две разные value-функции могут привести к одной и той же политике, когда агент действует жадно в соответствии с ними [19]. Следовательно, можно было бы изучить несколько более простых value-функций, если изучить сложную value-функцию сложно или даже невозможно. В таком случае глобальная функция награды может быть соответственно разложена на несколько различных функций награды:

$$R_{rev}(s, a, s') = \sum_{k=1}^{n} R_k(s, a, s') \tag{2.3}$$

где R_{rev} - глобальная функция награды, которая разбита на n функций награды. Каждая разложенная функция награды зависит от подмножества состояний и имеет свою собственную функцию Q-значения. Глубокая Q-сеть с общими нижними слоями и n головами используется для аппроксимации n Q-значений, обусловливающих текущее состояние и действие с различными функциями разложения награды рис.2.4.

$$Q_{HRA}(s, a; \theta) := \sum_{k=1}^{n} Q_k(s, a; \theta) \tag{2.4}$$

Q-сеть затем итеративно улучшается за счет оптимизации функции потерь

$$L_i(\theta_i) = \mathbb{E}_{s,a,r,s'}[\sum_{k=1}^{n}(y_{k,i} - Q_k(s,a;\theta_i))^2] \tag{2.5}$$

где

$$y_{k,i} = R_k(s,a,s') + \gamma \max_{a'} Q_k(s',a';\theta_{i-1}) \tag{2.6}$$

θ_i — это веса Q-сети в текущей итерации i, а θ_{i-1} - веса отдельной целевой сети в предыдущей итерации. В гибридной архитектуре вознаграждений n Q-значений выводятся n головами из одной единственной Q-сети. Однако для того, чтобы иметь консистентный градиент для улучшения политики для каждого измерения действий, следует создавать отдельные Q-сети для оценки Q-значений для каждого измерения действий.

2.4. Учебная программа

Обучение по *учебной программе* — это стратегия обучения, основанная на процессе обучения человека. В системе образования люди обучаются путем ознакомления с концепциями, которые строятся от простых до сложных уровней, от конкретных до более абстрактных уровней, от простых структур до сложных модулей.

Машинное обучение заимствует такой подход. *Учебная программа* направляет обучение программы от небольших подзадач или простых аспектов к более тяжелым и сложным задачам. Ожидается, что он достигнет высокой эффективности и результатов обучения [25].

В методе *учебная программа* простой аспект проблемы проливает свет на общую картину, а сложные факторы постепенно добавляются и раскрывают больше деталей целой картины. Поиск решения для сглаженной версии проблемы и затем переход к более детализированной может помочь обучению решить проблему постепенно [3].

Математически $C_\gamma(\theta)$ определяется как функции стоимости задачи, в которой γ - уровень сложности, а θ - параметры. Начальную сглаженную версию C_0 обычно легко оптимизировать, доведя θ до локального минимума. Локальный минимум затем используется в качестве основы для следующего уровня сложности. С увеличением γ, C_γ становится менее сглаженной при дальнейшем поиске

следующего локального минимума на основе предыдущего локального минимума. [9]

Учебный план также можно рассматривать как последовательное пере-взвешивание, в начале на наборах данных из простых примеров, а затем на полном наборе данных. С увеличением уровня сложности в набор обучающих данных добавляются немного более сложные примеры, которые используются для повторного взвешивания распределения. В конце используется полный набор обучающих данных. [9]

ГЛАВА 3. ПРИМЕНЯЕМЫЕ МЕТОДЫ

Основным алгоритмом, применяемым в этой работе, является *градиент глубокой детерминированной политики* (Deep Deterministic Policy Gradient, DDPG), который использует архитектуру актор-критик и может работать в пространстве непрерывных действий. Поскольку сценарий игры предполагает совместную работу нескольких агентов, используется мультиагентная модификация алгоритма DDPG - *мультиагентный градиент глубокой детерминированной политики* (MADDPG). Основное его отличие, делающее его более подходящим для работы с несколькими агентами, состоит в том, что MADDPG имеет N наборов ИНС критиков-акторов, где N соответствует количеству агентов в сценарии, благодаря чему, каждый агент имеет свой собственный механизм оптимизации политики. В оригинальном дизайне [24] это сделано для адаптации как к совместной работе, так и к конкуренции между агентами. Поскольку эта ВКР фокусируется на совместной работе нескольких агентов, тестируются несколько вариантов MADDPG, подходящих для совместной работы.

3.1. Основные методы

3.1.1. Архитектура актор-критик

Как уже упоминалось выше, алгоритм MADDPG имеет набор сетей актор–критиков для каждого агента в игровой среде. Для облегчения многоагентной совместной работы, во время тренировки, алгоритм учитывает наблюдения и действия всех агентов. Политики оптимизируются путем оценки качества, поведения всех агентов в разных состояниях среды. Таким образом, в условиях среды, требующей взаимодействия агентов, будет разработана политика, оптимальная для сотрудничества.

Глубокая нейронная сеть может рассматриваться как аппроксиматор нелинейных функций. В решении сложных задач глубокого обучения с подкреплением нейронная сеть нестабильна [6]. Эту проблему решает использование целевой (target) сети, поскольку мягкое и разреженное обновление целевой сети замедляет ее приближение к исходной сети и уменьшает влияние ошибок. Таким образом, это нарушает корреляцию между текущими и целевым Q-значениями. Хотя это

снижает скорость обучения, но и улучшает стабильность обучения. В алгоритме MADDPG и у актера, и у критика есть свои целевые сети.

3.1.2. Воспроизведение опыта (Experience Replay)

Воспроизведение опыта — это механизм, когда во время тренировки в игровой среде буфер используется для сбора опыта, образцы которого из этого буфера затем случайным образом извлекаются для обучения модели.

Опыт генерируется последовательно, во время взаимодействия агентов со средой в хронологическом порядке. Неизбежно то, что собранные последовательности опыта коррелируют друг с другом. Из-за этого сеть легко переобучается на имеющиеся последовательности опыта. В итоге, переобученная сеть не может обеспечить разнообразный опыт для последующего обучения.

Опыт случайно отбирается в небольшие блоки. Это нужно не только для эффективного использования аппаратного обеспечения и увеличения скорости обучения, но также нарушает корреляцию последовательности в буфере. Таким образом, сети имеют возможность формулировать более обобщенные политики по независимым друг от друга данным.

Для применения воспроизведение опыта, в буфер, складывается опыт в виде кортежей переходов, включающих наблюдения, действия, награды и последующие наблюдения, то есть $e_t = (\mathbf{s}, \mathbf{a}, \mathbf{r}, \mathbf{s}')$. Поскольку это многоагентная игровая среда, $\mathbf{s}$ - это совместные наблюдения, которые представляют собой совокупность наблюдений n агентов, $\mathbf{s} = [s_0, s_1, ..., s_n]$. То же самое относится к действиям, следующим наблюдениям и наградам. Буфер имеет ограниченную емкость, и при переполнении старый опыт вытесняется новым.

3.1.3. Тренировка ИНС

В то время, как агенты исследуют игровую среду, опыт перехода на каждом шаге собирается в буфер памяти. Обучение происходит только тогда, когда количество кортежей в буфере превышает определенную величину. Когда начинается обучение, *актор* и *критик* каждого из *n* агентов обновляется на каждом шаге, в то время как *целевой актор* и *целевой критик* обновляются с задержкой (раз в определенное количество шагов).

На рисунке рис.3.1 показано, как обновляется набор сетей акторов-критиков. Текущее Q-значение $Q_i(\mathbf{s}, \mathbf{a}|\theta_i^Q)$ оценивается по сети *критика*, при этом на вход

подаются совместные текущие наблюдения **s** и совместные действия **a**. Целевое Q-значение y_i рассчитывается из вознаграждения и дисконтированного следующего Q-значения $Q'_i(\mathbf{s}', \mathbf{a}'|\theta^{Q'_i})$ аппроксимированного по *целевому критику*. Входные данные для сети *целевого критика* – это совместные последующие наблюдения **s**' из буфера и совместные последующие действия **a**', оценённые *целевыми* сетями *акторов*:

$$\begin{aligned}\mathbf{a}' &= [a'_0, a'_1, ..., a'_n] \\ &= [\mu'_0(s'_0|\theta^{\mu'_0}), \mu'_1(s'_1|\theta^{\mu'_1}), ..., \mu'_n(s'_n|\theta^{\mu'_n})]\end{aligned} \tag{3.1}$$

Целевое Q- значение:

$$y_i = r_i + \gamma Q'_i(\mathbf{s}', \mathbf{a}'|\theta^{Q'_i}) \tag{3.2}$$

где r_i это награда – i-го агента, и γ - это коэффициент дисконтирования или скорость затухания влияния будущих наград.

Наконец, сеть *критика* обновляется путем минимизации функции потерь:

$$L_i(\theta^{Q_i}) = \frac{1}{S}\sum(y_i - Q_i(\mathbf{s}, \mathbf{a}|\theta^{Q_i}))^2 \tag{3.3}$$

где S - это размер тренировочного набора.

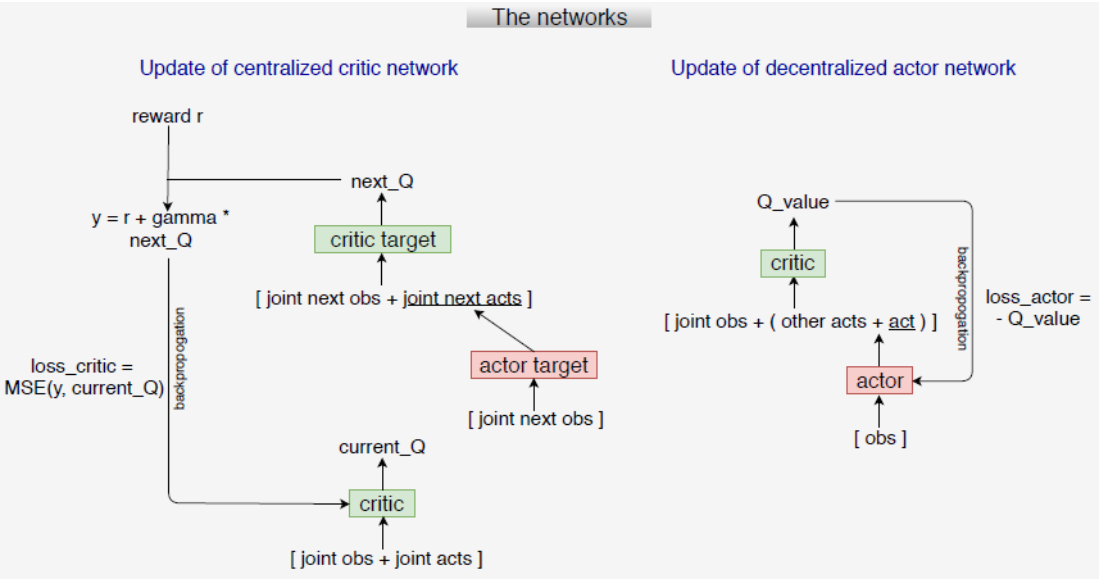

Рис.3.1. Обновление сети *актора* и *критика* для каждого агента в MADDPG.

Сеть *актора* i-го агента обновляется путем максимизации Q-значения сетью *критика*. *Критик* принимает совместные текущие наблюдения **s** из буфера и совместные текущие действия, в которых i-е действие подменяется последним действием оцененным *актором*. То есть:

$$[a_0, a_1, a_i, ..., a_n] = [a_0, a_1, \mu_i(s_i|\theta^{\mu_i}), ..., a_n] \tag{3.4}$$

Для оптимизации уместно преобразовать задачу максимизации в задачу минимизации. Следовательно, функция потерь для обновления сети актора:

$$L_i(\theta^{\mu_i}) = -Q_i(\mathbf{s}, [a_0, a_1, \mu_i(s_i|\theta^{\mu_i}), ..., a_n]|\theta^{Q_i}) \tag{3.5}$$

Сети *целевых актора* и *критика* обновляются не полным копированием сетей *актора* и *критика*, а с использованием *мягкого обновления* (soft update):

$$\begin{aligned} \theta^{\mu'_i} &= \tau\theta^{\mu_i} + (1-\tau)\theta^{\mu'_i} \\ \theta^{Q'_i} &= \tau\theta^{Q_i} + (1-\tau)\theta^{Q'_i} \end{aligned} \tag{3.6}$$

где τ обычно очень мала, например 0.01 в этой работе.

3.2. Прикладные методы

3.2.1. Архитектура ветвления действий (Action Branching)

В сценариях, подразумевающих совместную работу нескольких агентов наряду с физическими действиями важно иметь и коммуникационные действия. В некоторых случаях количество действий может быть большим, как в [35]. В игровых сценариях этой работы агент имеет двумерные действия, которые представляют физическое движение и действие, выбранное для общения.

Полное действие в таких сценариях включает в себя два относительно независимых действия. Сеть *акторов* проектируется таким образом, что она выделяет скрытые представления из наблюдений и имеет две «головы» на выходе. Одна возвращает физическое действие, вторая – действие для общения.

В соответствии с теорией архитектуры ветвления действий [35], можно оптимизировать каждое измерение действия относительно независимо. Полное действие в итоге представляет собой объединение двух действий.

3.2.2. Исследовательский шум (Exploration Noise)

Исследование и *эксплуатация* (Exploration and exploitation) — это дилемма в обучении с подкреплением. *Эксплуатация* — это когда агент следует текущей политике, чтобы совершать жадные действия, которые приносят наибольшую

награду. *Исследование* берет на себя риски, чтобы попробовать другие действия, которые потенциально могут принести лучшую награду в долгосрочной перспективе. *Исследование* необходимо агенту, ищущему оптимальную политику, хотя оно кажется неоптимальным в нынешней ситуации и дает меньшее вознаграждение. В обучении с подкреплением агент обычно больше *исследует* окружающую среду в начале обучения. В ходе оптимизации политики с течением времени, агент постепенно уменьшает и стабилизирует *исследование* до низкого уровня, и в итоге больше придерживается получившейся оптимальной политики.

Чтобы включить *исследование*, на действия накладываются шумы. Какой шум применять, определяется настройкой среды. *Процесс Орнштейна-Уленбека* используется в DDPG в [2] для получения коррелированных по времени *исследований*. Он считаются, эффективным для проблем физического контроля с инерцией. Гауссовское распределение шума используется для физических движений в [4]. В этой работе для наложения шума на действия, генерируемые сетью *акторов* выбирается стандартное *Гауссовское распределение*.

$$\mu_i'(s_i) = \mu_i(s_i|\theta^{\mu_i}) + \mathcal{N} \tag{3.7}$$

где μ' - политика исследования, а $\mathcal{N}$ - шум исследования, который можно выбрать в зависимости от настроек среды. $\mathcal{N}$ затухает на каждом шаге со скоростью ε, то есть $\mathcal{N} \leftarrow \varepsilon\mathcal{N}$.

3.3. Варианты алгоритмов

3.3.1. MADDPG с декомпозированным вознаграждением (Decomposed Reward)

TODO: перевести

Предполагается сценарий игры, в котором агенты выполняют относительно независимые под-действия, каждое под-действие может воздействовать на среду и получать вознаграждение, которое отделено от глобального вознаграждения. Идея декомпозирования награды MADDPG в том, что для каждого под-действия может быть сформулирована независимая политика. Каждый агент, имеет n независимых наборов сетей *актор-критиков*, каждый из которых соответствует одному виду действия. Ожидается, что политики для видов действий могут быть оптимизированы путем обучения соответствующих групп критиков.

В этой работе планируется использовать этот вариант в сценарии *Simple Reference*. Позже, в разделе TODO будет полее подробно описано, как декомпозируется глобальная награда и как формулируется оптимальная политика для каждого вида действия.

3.3.2. MADDPG с общим мозгом

Другой вариант MADDPG вдохновлен [23]. В этом варианте есть только один набор сетей *акторов-критиков*, который используется всеми агентами. Это подразумевает, что все агенты имеют одинаковую оптимальную политику. Этот подход использует предположение о том, что все агенты имеют одно и то же пространство действий и пространство наблюдений, и они также имеют общую глобальную награду. Этот подход особенно эффективен для коммуникационных действий, поскольку композиционный язык постоянно появляется среди всех участников игрового сценария. В варианте с общим набором сетей все агенты говорят на одном языке, в отличие от стандартного MADDPG, где каждый агент может интерпретировать один и тот же ориентир по-разному. Это особенно важно, когда сотрудничают более двух агентов, поскольку им нужно общаться на одном языке.

ГЛАВА 4. ЭКСПЕРИМЕНТЫ

Эксперименты проводятся в мультиагентной среде multiagent-particle-envs [13] от компании OpenAI.

4.1. Сценарий 1. Simple Speaker Listener

Сценарий *Simple Speaker Listener* воспроизводится и тестируется с использованием алгоритма MADDPG [24].

Сценарий упоминается в разделе 0.2.1. В этом сценарии два агента имеют разные пространства действий и наблюдений. Наблюдение *говоруна* o_s - это цвет цели, обозначенный 3- канальным вектором $d \in \mathbb{R}^3$. Наблюдение *слушателя* o_l – это вектор конкатенации его собственной скорости $\upsilon \in \mathbb{R}^2$ его расстояния до трех ориентиров $p = [p_1, p_2, p_3]$, $p_i \in \mathbb{R}^2$ и сигнал, произнесенный *говоруном* на предыдущем временном шаге. Это:

$$o_s = [d] o_l = [\upsilon, p, c] \tag{4.1}$$

Коммуникационное действие *говоруна* обозначается «one-hot encoding» вектором $[1; 0; 0]$ или $[0; 1; 0]$ или $[0; 0; 1]$, для обозначения трех ориентиров соответственно. Физическое действие *и* слушателя - это 5-канальный вектор, каждый из которых представляет одно направление движения (вверх, вниз, влево, вправо или без движения). Наблюдения и действия *говоруна* и *слушателя* и их взаимосвязь показаны на рисунке рис.4.1.

Два агента имеют общую награду r, которая является отрицательным евклидовым расстоянием между *слушателем* и его целью. Проблема, которую необходимо решить в этом сценарии, заключается в поиске оптимальных политик для *говоруна* и *слушателя*, чтобы максимизировать ожидаемую награду, то есть $\max_\pi R(\pi)$, где

$$R(\pi) = \mathbb{E}[\sum_{t=0}^{T} r(s_t, a_t)] \tag{4.2}$$

Во время обучения *говорун* учится различать три ориентира и передавать целевой ориентир *слушателю*. И *слушатель* должен изучить закодированные высказывания, говоруна, и перейти к правильной цели.

TODO: уточнить параметры

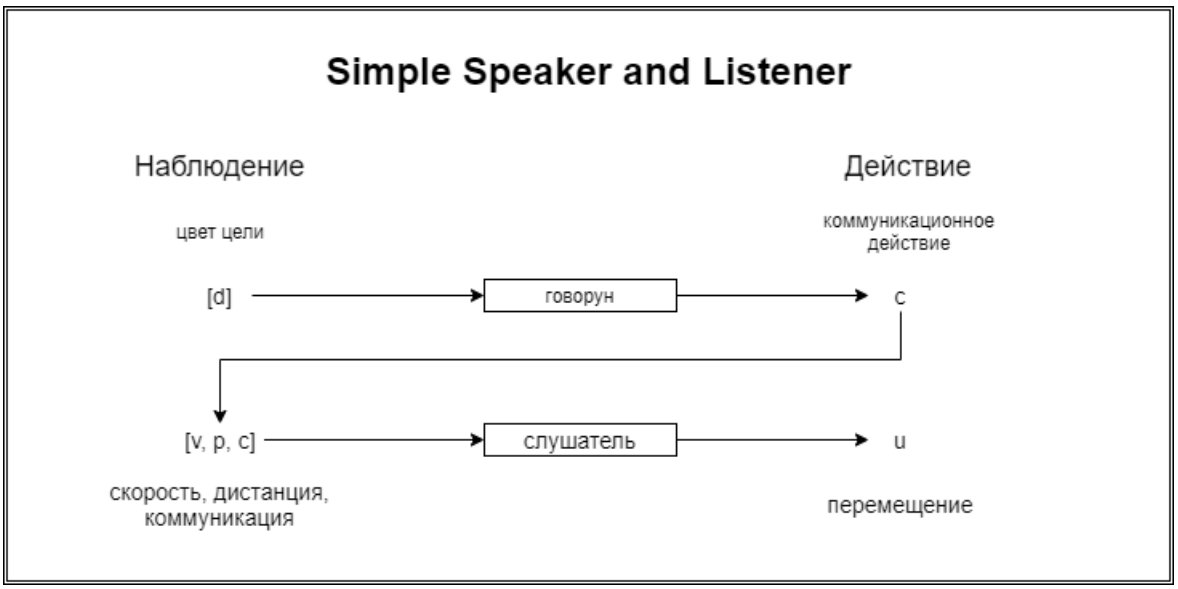

Рис.4.1. *Говорун* наблюдает цвет целевого ориентира *d* и издает коммуникационое действие, которое будет получено *слушателем*. *Слушатель* производит физическое движение.

Архитектура актор-сети *говоруна* аналогична архитектуре *слушателя*, каждый из которых содержит два полно связанных слоя с 64 нейронами и использует функцию активации relu. Однако их выходные слои различаются с точки зрения количества единиц и функций активации. Для *говоруна* используется функция активации Gumbel-Softmax, а для слушателя - tanh. Сети критиков имеют структуру, аналогичную сетям акторов, за исключением того, что они выдают скалярное Q-значение.

4.2. Сценарий 2

TODO

4.3. Сценарий 3

TODO

ГЛАВА 5. РЕЗУЛЬТАТЫ

TODO

5.1. Сценарий 1

TODO

5.2. Сценарий 2

TODO

5.3. Сценарий 3

TODO

ЗАКЛЮЧЕНИЕ

Данная работа способствует дальнейшему пониманию совместной работы множества агентов в игровой среде. Проведенные эксперименты доказывают, что агенты должны вырабатывать оптимальную политику сотрудничества с учетом полного наблюдения за состоянием окружающей среды и политиками других агентов. Очевидно, что в результате обучения возникает композиционный язык, который улучшает сотрудничество. К сожалению, в данной работе алгоритм работает в определенных сценариях, но не удалось реализовать все задуманные приемы, и не все реализованные варианты алгоритма удалось заставить работать, вероятно, из-за нехватки времени и ресурсов.

По сравнению с предыдущими работами эта дипломная работа расширяет MADDPG до более сложного сценария, когда агентам необходимо сотрудничать в многомерных пространствах действий. Вкратце, агенты одновременно выполняют физическое движение и коммуникационное высказывание.

Некоторые варианты MADDPG исследуются для разных игровых сценариев.

MADDPG с разложенным вознаграждением может быть адаптирован к сценариям со сложным глобальным вознаграждением.

MADDPG с одним мозгом может применяться, если агенты в сценарии симметричны в пространстве наблюдения и действия и имеют одинаковое глобальное вознаграждение. Кроме того, агенты, обученные MADDPG с одним мозгом, говорят на одном языке, что важно для сценариев, с более чем двумя агентами, которые должны общаться между собой для достижения цели.

Проблема насыщения сети постоянно встречается в процессе дипломной работы. Возможным объяснением этого может быть то, что для изучения оптимальных политик в пространстве непрерывных действий нейронные сети должны быть достаточно сложными, чтобы извлекать скрытые функции, оценивать Q значения или оптимизировать политики.

Дальнейшая работа Естественным продолжением дальнейшего изучения мультиагентного взаимодействия было бы исследование сотрудничества большего количества агентов. Они могут обладать более многомерным пространством действий. Они должны говорить на одном языке, если необходимо общение между более чем двумя агентами. Следовательно, алгоритмы и сети необходимо модифи-

цировать и адаптировать с учетом новых ситуаций. Приведенные выше выводы в этой дипломной работе могут способствовать дальнейшим исследованиям.

Кроме того, все еще стоит дополнительно изучить использование алгоритма детерминированного градиента политики для задач с несколькими агентами в сценариях трехмерных игр. Основной проблемой для алгоритма в сценариях 3D-игр может быть высокоразмерный ввод пикселей. Дальнейшая работа должна быть сделана соответственно над структурой нейронных сетей, например, сверточных слоев в актере и настройке гиперпараметров.

После отработки алгоритма на 3D средах с высокоразмерным вводом, можно пробовать применить алгоритм к обучению нескольких роботов для достижения общей цели в реальной среде. Например, сбор мусора, различные задачи на преследование и т. д.

TODO: Последним абзацем в заключении можно выразить благодарность всем людям, которые помогали автору в написании ВКР.

СПИСОК СОКРАЩЕНИЙ И УСЛОВНЫХ ОБОЗНАЧЕНИЙ

ML Машинное обучение (Machine Learning).

DL Глубокое обучение (Deep Learning).

RL Обучение с подкреплением (Reinforcement Learning).

DPG Детерминированный градиент политики (Deterministic Policy Gradient).

DDPG Глубокий детерминированный градиент политики (Deep Deterministic Policy Gradient).

MADDPG Мультиагентный глубокий детерминированный градиент политики (Multiagent Deep Deterministic Policy Gradient).

ИНС Искусственная нейронная сеть (Artificial Neural Network).

МППР Марковский процесс принятия решений (Markov Decision Process, MDP).

СЛОВАРЬ ТЕРМИНОВ

TeX — язык вёрстки текста и издательская система, разработанные Дональдом Кнутом.

LaTeX — язык вёрстки текста и издательская система, разработанные Лэсли Лампортом как надстройка над TeX.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. AlphaGo. — URL: https://en.wikipedia.org/wiki/AlphaGo.

2. An Introduction to Statistical Learning: with Applications in R / G. James [и др.]. — Springer New York, 2014. — (Сер.: Springer Texts in Statistics). — URL: https://books.google.se/books?id=at1bmAEACAAJ.

3. Automated Curriculum Learning for Neural Networks / A. Graves [и др.]. — 2017. — arXiv: 1704.03003 [cs.NE].

4. *Bengio Y.*, *Courville A.*, *Vincent P.* Representation Learning: A Review and New Perspectives. — 2012. — arXiv: 1206.5538 [cs.LG].

5. *Bishop C.* Pattern Recognition and Machine Learning. — Springer, 2006. — 738 с. — URL: http://research.microsoft.com/en-us/um/people/cmbishop/prml.

6. Continuous control with deep reinforcement learning / T. P. Lillicrap [и др.]. — 2015. — arXiv: 1509.02971 [cs.LG].

7. Counterfactual Multi-Agent Policy Gradients / J. Foerster [и др.]. — 2017. — arXiv: 1705.08926 [cs.AI].

8. *Crevier D.* AI: The Tumultuous History of the Search for Artificial Intelligence. — 1993. — с. 1—386. — URL: https://www.researchgate.net/publication/233820788_AI_The_Tumultuous_History_of_the_Search_for_Artificial_Intelligence.

9. Curriculum Learning / Y. Bengio [и др.] // Proceedings of the 26th Annual International Conference on Machine Learning. — Montreal, Quebec, Canada: Association for Computing Machinery, 2009. — с. 41—48. — (Сер.: ICML '09). — DOI 10.1145/1553374.1553380. — URL: https://doi.org/10.1145/1553374.1553380.

10. Deep Reinforcement Learning: A Brief Survey / K. Arulkumaran [и др.] // IEEE Signal Processing Magazine. — 2017. — т. 34, № 6. — с. 26—38. — DOI 10.1109/msp.2017.2743240. — URL: http://dx.doi.org/10.1109/MSP.2017.2743240.

11. *Dimitri P. Bertsekas J. N. T.* Neuro-dynamic Programming. — Athena Scientific, 1996. — (Сер.: Anthropological Field Studies). — URL: https://books.google.ru/books?id=WxCCQgAACAAJ.

12. *Edwards C.* Growing Pains for Deep Learning // Commun. ACM. — New York, NY, USA, 2015. — т. 58, № 7. — с. 14—16. — DOI 10.1145/2771283. — URL: https://doi.org/10.1145/2771283.

13. github: openai - multiagent-particle-envs. — URL: https://github.com/openai/multiagent-particle-envs.

14. *Goodfellow I.*, *Bengio Y.*, *Courville A.* Deep Learning. — MIT Press, 2016. — (http://www.deeplearningbook.org).

15. *Hastie T.*, *Tibshirani R.*, *Friedman J.* The Elements of Statistical Learning: Data Mining, Inference, and Prediction. — Springer, 2001. — (Сер.: Springer series in statistics). — URL: https://books.google.ru/books?id=VRzITwgNV2UC.

16. *Hornik K.* Approximation capabilities of multilayer feedforward networks // Neural Networks. — 1991. — т. 4, № 2. — с. 251—257. — DOI https://doi.org/10.1016/0893-6080(91)90009-T. — URL: http://www.sciencedirect.com/science/article/pii/089360809190009T.

17. *Huang S. H.*, *Hong-Chao Zhang*. Artificial neural networks in manufacturing: concepts, applications, and perspectives // IEEE Transactions on Components, Packaging, and Manufacturing Technology: Part A. — 1994. — т. 17, № 2. — с. 212—228.

18. Human-level control through deep reinforcement learning / V. Mnih [и др.] // Nature. — 2015. — т. 518. — с. 529—33. — DOI 10.1038/nature14236.

19. Hybrid Reward Architecture for Reinforcement Learning / H. van Seijen [и др.]. — 2017. — arXiv: 1706.04208 [cs.LG].

20. *Kanter J. M.*, *Veeramachaneni K.* Deep feature synthesis: Towards automating data science endeavors // 2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA). — 2015. — с. 1—10.

21. *Khajanchi A.* Artificial Neural Networks : The next intelligence //. — 2003. — URL: https://www.semanticscholar.org/paper/Artificial-Neural-Networks-%3A-The-next-intelligence-Khajanchi/312a65e33ebba4cbff154a79f57e2a0ff386e6f6.

22. *Mitchell T.* Machine Learning. — McGraw-Hill, 1997. — (Сер.: McGraw-Hill International Editions). — URL: https://books.google.ru/books?id=EoYBngEACAAJ.

23. *Mordatch I.*, *Abbeel P.* Emergence of Grounded Compositional Language in Multi-Agent Populations. — 2017. — arXiv: 1703.04908 [cs.AI].

24. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments / R. Lowe [и др.]. — 2017. — arXiv: 1706.02275 [cs.LG].

25. *Narvekar S.* Curriculum Learning in Reinforcement Learning // Proceedings of the 26th International Joint Conference on Artificial Intelligence. — Melbourne, Australia: AAAI Press, 2017. — с. 5195—5196. — (Сер.: IJCAI'17).

26. OpenAI Gym. — URL: https://github.com/openai/gym.

27. *Otterlo M. van*, *Wiering M. A.* Reinforcement Learning and Markov Decision Processes // Reinforcement Learning. — 2012.

28. *Russell S.*, *Norvig P.* Artificial Intelligence: A Modern Approach. — 3rd. — USA: Prentice Hall Press, 2009. — с. 1—5.

29. *Samuel A. L.* Some Studies in Machine Learning Using the Game of Checkers // IBM Journal of Research and Development. — 1959. — т. 3, № 3. — с. 210—229. — URL: http://dx.doi.org/10.1147/rd.33.0210.

30. *Sultan K.*, *Ali H.*, *Zhang Z.* Big Data Perspective and Challenges in Next Generation Networks // Future Internet. — 2018. — т. 10, № 7. — с. 56. — DOI 10.3390/fi10070056. — URL: http://dx.doi.org/10.3390/fi10070056.

31. *Sutton R. S.*, *Barto A. G.* Reinforcement Learning: An Introduction. — 2-е изд. — The MIT Press, 2008. — с. 2. — URL: https://books.google.ru/books?id=VRzITwgNV2UC.

32. *Sutton R. S.*, *Barto A. G.* Reinforcement Learning: An Introduction. — 2-е изд. — The MIT Press, 2008. — с. 50. — URL: https://books.google.ru/books?id=VRzITwgNV2UC.

33. *Sutton R. S.*, *Barto A. G.* Reinforcement Learning: An Introduction. — 2-е изд. — The MIT Press, 2008. — с. 107—108. — URL: https://books.google.ru/books?id=VRzITwgNV2UC.

34. *Szabó Z. G.* Compositionality // Stanford Encyclopedia of Philosophy. — 2008.

35. *Tavakoli A.*, *Pardo F.*, *Kormushev P.* Action Branching Architectures for Deep Reinforcement Learning. — 2017. — arXiv: 1711.08946 [cs.LG].

36. The Arcade Learning Environment: An Evaluation Platform for General Agents / M. G. Bellemare [и др.] // Journal of Artificial Intelligence Research. — 2013. — т. 47. — с. 253—279. — DOI 10.1613/jair.3912.

37. Watson (computer). — URL: https://en.wikipedia.org/wiki/Watson_(computer).