МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра АПУ

ОТЧЕТ

по преддипломной практике

Тема: Применение методов обучения с подкреплением в задачах управления

Студентка гр. 5391 _____ Шпаковская И.И.

Руководитель д.т.н., профессор _____ Душин С.Е.

Санкт-Петербург
2021

ЗАДАНИЕ
НА ПРЕДДИПЛОМНУЮ ПРАКТИКУ

Студентка Шпаковская И.И.
Группа 5391
Тема практики: Применение методов обучения с подкреплением в задачах управления
Задание на практику: Необходимо проанализировать методы обучения с подкреплением на предмет применения алгоритмов для задачи разработки систем управления.

Сроки прохождения практики: 10.02.2021 - 25.05.2021
Дата сдачи отчета: 20.05.2021
Дата защиты отчета: 25.05.2021

Студентка _____ Шпаковская И.И.

Руководитель д.т.н., профессор _____ Душин С.Е.

СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

В настоящей пояснительной записке применяют следующие термины с соответствующими определениями:

ДП – динамическое программирование

МППР – Марковский процесс принятия решения

ИИ – искусственный интеллект

ОУ – объект управления

RL – обуение с подкреплением (англ. Reinforcement learning)

MPC – управление с прогнозирующими моделями (англ. Model Predictive Control)

PI – алгоритм итерации по стратиегиям (англ. Policy Iteration)

VI – алгоритм итерации по ценности (англ. Value Iteration)

1 ОСНОВНЫЕ ПОЛОЖЕНИЯ ОБУЧЕНИЯ С ПОДКРЕПЛЕНИЕМ

Цель главы – дать общее представление о терминах и алгоритмах обучения с подкреплением. В параграфе 1.1 приводится актуальность изучения новой области инженерами в области автоматизации. В параграфе 1.2 приведены основные положения обучения с подкреплением в терминологии свойственной для специалистов в области искусственного интеллекта, именно на базе введенной в данном параграфе терминологии формулируются методы и алгоритмы обучения с подкреплением в параграфе 1.6.

1.1 Актуальность обучения с подкреплением

Обучение с подкреплением (англ. Reinforcement learning, RL) – это широкая область, объединяющая исследователей из самых разных областей: искусственный интеллект (ИИ), управление, робототехника, исследования операций, экономика, нейробиология. По данной теме было опубликовано множество книг и обзорных статей представляющих самые разные области: ИИ, где классический учебник – это учебник Саттона и Барто (1998) [1] со вторым изданием [2], но также и другие [3; 4]; теория управления [5]; оптимальное управление [6; 7]; робототехника [8]; Некоторые исследования сосредоточены на конкретных задачах RL: как методы градиента стратегии, аппроксимация функций, байесовские формулировки RL, иерархический RL, многоагентные подходы, глубокий RL, безопасный RL и так далее.

В то же время, необходимо упомянуть о том, что количество приложений методов RL для разработки систем управления техническими объектами мало в сравнении с количеством и масштабам приложений в области разработки рекомендательных систем, игр, обработки информации и исследований операций.

С одной стороны сложность применения к техническим системам затруднена в силу сложности обеспечения безопасности. В отличие от симуляции, в физическом мире действия имеют реальные последствия. В результате любой алгоритм, развернутый в реальных системах, должен отвечать нормам безопасности и качества. Безопасность в известных средах давно рассматривается и формализуется сообществами контроля и формальных методов, где можно синтезировать стратегии контроля, соответствующие заданной спецификации. Все методы определения устойчивости в области управления направлены на работу с моделями объектов. Тогда как при применении RL нет необходимо-

сти в математическом моделировании объетка. основаны на известной модели системы.

Другая сложность RL для специалистов в области управления – особенность терминологии и акцента на данные, которые привычны специалистам в области ИИ. Несмотря на эти и другие сложности, наблюдается рост исследований методов RL со стороны инженеров систем управления. Это подтверждается увеличением количества докладов и статьей, опубликованных в профильных журналах по автоматическому управлению (рис.1.1). Международная федерация по автоматическому управлению на конгрессе в 2020 году (IFAC-V 2020) вынесла на пленарное заседание доклад «Reinforcement Learning for Process Control and Beyond» (автор Jay H. Lee), что говорит о важности и влиятельности области обучения с подкреплением.

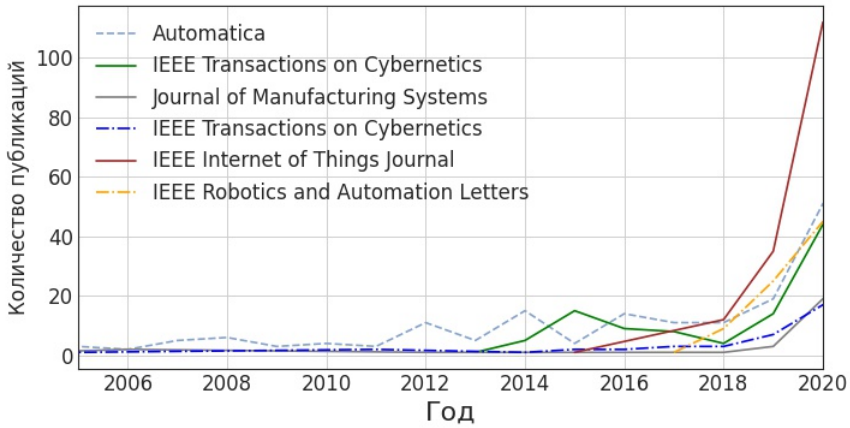

Рисунок 1.1 — Тендеции количества публикаций на тему RL в специализорованных журналах по управлению в технических системах

Учитывая существующий запрос на разработку интеллектуальных систем управление, что связано с возрастающей сложностью объектов управления, можно предполагать, что регуляторы с применением обучения с подкреплением станут неотъемлемой частью любого автоматизированного технологического процесса.

1.2 Терминология обучения с подкреплением

Обучение с подкреплением – это класс методов машинного обучения, которые основанны на взаимодействии алгоритма со средой и изменения стратегии действия или политики управления на основе стимулов, полученных в ответ на действия алгоритма, с целью получения желаемого результата. Цель применения методов RL — определить последовательность входных сигналов, которая

обеспечивает желаемую работу динамической системы, начиная с минимальных знания о работе системы.

Агент – интеллектуальная сущность (система/робота/алгоритм) принимающая решения, взаимодействует с объектом, который называется окружением или средой (англ., environment). Агент и окружение взаимодействуют на каждом шаге последовательности $t = 0, 1, 2, \ldots$ При этом окружение задается, зависящим от времени состоянием $S_t \in S$, где S – множество всех возможных состояний. Агенту в каждый момент времени в общем случае доступно только некоторое наблюдение (англ., observation) текущего состояния среды. На основании наблюдений состояния агент выполняет процедуру выбора действия $A_t \in A(S_t)$, $A(S_t)$ – множество действий доступных агенту в состоянии S_t. Процедура выбора действий агентом называются стратегией или политикой (англ., policy) и обозначается как π_t, описывая вероятность выбора действия $A_t = a$ в состоянии $S_t = S$. По результатам применения действия $A(S_t)$ в среде, на вход агента поступает численная награда $R_{t+1} \in R$ (англ. reward) и новое состояние среды $A(S_{t+1})$. Методы RL определяют способ выбора стратегии агентом в результате полученного опыта.

Необходимо уточнить, что понимается под термином окружающая среда – это все то, что не является агентом. Среда принимает текущее состояние и действие агента в качестве входных данных и возвращает вознаграждение агента и состояние.

Марковские процессы. Одна из структур для RL основана на марковских процессах принятия решений (МППР). Задача RL удовлетворяет условию Марковости – процесс зависит только от текущего состояния и не зависит от всей предыдущей истории. МППР позволяет формализовывать основные элементы RL, такие как функции ценности, награды, а далее основные алгоритмы RL. Многие задачи принятия решений могут быть сформулированы как МППР, включая системы управления с обратной связью. МППР представляет собой четверку (S, A, P, r), где:

S – конечное пространство состояний;

A – конечное пространство действий;

P – функция переходов, определяющая вероятность перейти в состояние s' из состояния s посредством действия a;

r – функция награды. При условии любого состояния s и действия a, вероятность каждого возможного следующего состояния равна:

$$p_{ij}(a,t) = P\{s_{t+1} = j | s_t = i, a_t = a\}$$

На рис. 1.2 представлена общая структурная схема RL в терминах МППР.

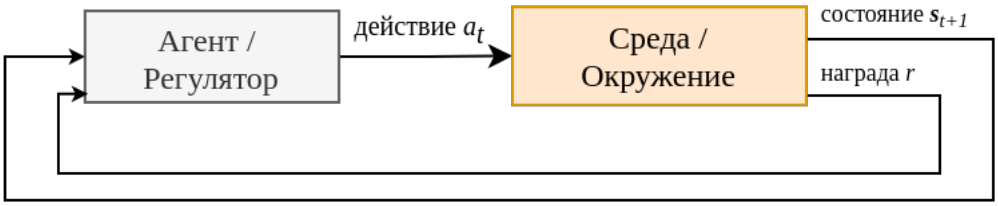

Рисунок 1.2 — Общая структура обучения с подкреплением

В задаче обучения с подкреплением, цель агента формализуется в сигнале награды, которую получает агент от среды на каждом временном шаге. Использование сигнала награды для формализации цели – одна из отличительных характеристик задачи обучения с подкреплением. Алгоритмы RL оптимизируют ту награду, которая была предоставлена ему на входе, при этом принимая истинной «гипотезы награды», которая утверждает, что любую интеллектальную задачу можно определить (задать) при помощи функции награды. Но, как несложно догадаться, на практике дизайн функции награды оказывается сложной задачей и качество работы алгоритмы, напрямую зависит от способности формализовать цель управления.

Пусть последовательность наград после временного шага t, записывается как r_t, r_{t+1}.... Тогда в рамках задачи обучения с подкреплением, необходимо максимизировать ожидаемую награду R_t, которая определена, как сумма всех последующих наград: Тогда в рамках задачи обучения с подкреплением, необходимо максимизировать ожидаемую награду R_t, которая определена как сумма всех последующих наград:

$$R_t = r_t + r_{t+1} + r_{t+2} + \ldots$$

Такое выражение удовлетворительно для задач с конечным временным шагом, В системах с конечным количеством шагов взаимодействия, вводится понятие разбиения взаимодействие агента и окружения на последовательности – эпизоды. Каждый эпизод заканчивается особым терминальным состоянием, за которым следует переход к заданному начальному состоянию или к выбору начального состояния из распределения начальных состояний. Среда называется эпизо-

дичной, если для любой стратегии процесс взаимодействия гарантированно завершается не более чем за некоторое конечное число шагов.

Для ряда задач управления характерно продолжительное действие – взаимодействие агента и окружения не разбивается на эпизоды. В этом случае ожидаемая награда, которую необходимо максимизировать, может достигать бесконечности. Для решения таких задач вводится коэффициент обесценивания (дисконтирования), тогда награда формируется следующим образом:

$$R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots + \gamma^{k-1} r_{t+k-1}$$

где $\gamma \in (0, 1]$ – коэффициент дисконтирования.

Предполагается, что на каждом шаге с вероятностью $1 - \gamma$ взаимодействие обрывается, и итоговым результатом агента станет та награда, которую он успел собрать до прерывания.Это обеспечивает приоритет получению награды в ближайшее время перед получением той же награды через некоторое время. Математически смысл дисконтирования, во-первых, в том, что данный коэффициент позволяет гарантировать ограниченность оптимизируемого функционала, а во-вторых, выполнение условий некоторых теоретических результатов, которые явно требуют $\gamma < 1$.

Учитывая терминалогию приведенную выше, задача RL для заданного МППР может быть сформулирована как поиск стратегии π^*, максимизирующей среднюю дисконтированную суммарную награду.

$$J(\pi) = E_\pi \sum_{t=0}^{\infty} \gamma^t r(s_t, \pi(s_t)) \rightarrow \max_\pi \tag{1.1}$$

В основе многих алгоритмов RL лежит понятие функции ценности и функции ценности действия. Это в некотором роде «обобщение» функционала 1.1, варьируя начальное состояние. *Функция ценности* (англ. Value Function) или оценочная функция состояния $V^\pi(s)$ показывает сколько набирает в среднем агент из состояния s_t при стратегии π. Функция ценности определяется для отдельных стратеги π и равна сумме наград:

$$V^\pi(s) = E_\pi\{R_t | s_t = s\} = E_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, \pi(s_t)) \middle| s_t = s \right] \tag{1.2}$$

где E_π – ожидаемое значение награды R_t при действии агента в соответсвии со стратегией π на каждом t.

Функция ценности действия $Q^{\pi}(s,a)$ (англ. Value Action) для стратегии π характеризует сколько набирает в среднем агент из состояния s_t после выполнения действия a.

$$Q^{\pi}(s,a) = E_{\pi}\{R_t | s_t = s, a_t = a\} = E_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, \pi(s_t)) \middle| s_t = s, a_t = a\right]$$

Функцию ценности и функцию ценности действий также называют V-функцией и Q-функцией, соответственно.

Исходя из задания V-фунции (1.2) задача RL формулируется как определение политики $\pi(s,a)$, которая максимизирует награду:

$$\pi^*(s,a) = \arg\max_{\pi} V_t^{\pi}(s) = \arg\max_{\pi} E_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, \pi(s_t)) \middle| s_t = s\right]$$

Политика $\pi^*(s,a)$ называется оптимальной политикой, и соответствующее оптимальное значение V-функции задается как:

$$V_t^*(s) = \max_{\pi} V_t^{\pi}(s) = \max_{\pi} E_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, \pi(s_t)) \middle| s_t = s\right]$$

Легко заметить, что для максимизации $V_t^{\pi}(s_0)$, где s_0 – стартовое состояние, необходимо промаксимизировать $V_t^{\pi}(s)$. То есть задача имеет подзадачи эквивалентной структуры. Ряд методов RL основывается на рекурсивном свойстве V-функции. Тоже верно и для Q-функции. Это означает то, что для этих функций может быть записано управнение Беллмана, которое выражает отоношение между значениями функций в текущем состояний и значением в последующих состояний.

1.3 Историческая справка

Термин «подкрепление»(англ. reinforcement) унаследован из поведенческой психологии, а именно из работ физиолога И.П. Павлова. Здесь подкрепление обозначает награду или наказание за результат, который зависит как от принятых решений так и от внешних, в общем случае, не контролируемых воздействий. Под обучением здесь понимается поиск способов достичь желаемого результата методом проб и ошибок, то есть попытки решить задачу и использовать накопленный опыта для усовершенствования своей стратегии выбора действий в будущем.

Специалисты в области обучения с подкреплением на раннем этапе истории выделяют два основных направления развития. Одно направление связано

с обучением методом проб и ошибок. Второе направление связано с задачей оптимального управления и решением ее с помощью функций стоимости и динамического программирования.

Главной психологической идеей, которая используется в обучении с подкреплением, является метод проб и ошибок, предложенный ученым и философом Александром Бэном в 1855 году. Ученый объясняет возникновение произвольных движений – вводит представление о спонтанной активности нервной системы. Когда какое-либо движение более одного раза совпадает с состоянием удовольствия, то некоторая сила духа устанавливает между ними ассоциации. На базе идеи метода проб и ошибок начиная с 1930 годов было сделано ряд автоматов, демонстрирующий этот подход. Самая ранняя демонстрация — машина Томаса Росса 1933-ого года, которая могла пройти простейший лабиринт и запомнить последовательность переключателей. В 1952 году Клод Шеннон продемонстрировал лабиринт с роботом-мышью, который передвигался по лабиринту с помощью трех колес и магнита с обратной стороны лабиринта, он мог запомнить путь по лабиринту, исследуя его тем самым методом проб и ошибок. Появление таких электро-механических машин открыло путь к написанию компьютерных программ, способных к разным типам обучения, некоторые из которых были способны к обучению методом проб и ошибок.

Помимо психологического подхода, независимо развивался подход математического программирования. Термин «оптимальное управление» появился в конце 1950-х годов и применялся для описания задачи проектирования устройства управления, которое должно было максимизировать заданную характеристику поведения динамической системы во времени. Один из подходов к решению этой задачи был разработан в середине 1950-х годов Ричардом Беллманом и другими учеными путем обобщения теории Гамильтона–Якоби, созданной в XIX веке. В этом подходе понятия состояния динамической системы и функции стоимости, используются для вывода функционального уравнения – уравнение Беллмана. Класс методов решения задач оптимального управления путем решения уравнения Беллмана называется динамическим программированием [9]. В работе [10] описана дискретная стохастическая версия задачи оптимального управления, известная под названием «марковский процесс принятия решений» (МППР, англ. MDP). В работе Ховарда Роналда [11] предложен метод итерации по стратегиям для МППР. Все это – существенные элементы, лежащие в основе теории и алгоритмов обучения с подкреплением. Общепризнанно, что дина-

мическое программирование – единственный практически применимый способ решения общих стохастических задач оптимального управления. Большим недостатков динамического программирования является «проклятием размерности», т. е. требования к вычислительной мощности растут экспоненциально с ростом числа переменных состояния. Тем не менее динамическое программирования гораздо более эффективно и распространено, чем любой другой общий метод.

Развитие оптимального управления и обучения с подкреплением происходило независимо, так как данные области ставят перед собой разные цели. Так же влияет тот факт, что динамическое программировании представляется как пакетный метод вычислений, который сильно зависит от наличия точной модели системы и аналитических решений уравнения Беллмана. К тому же простейшая форма динамического программирования – вычисление, происходящее в обратном направлении по времени, поэтому трудно понять, как его можно применить в процессе обучения, который по необходимости протекает в прямом направлении.

Некоторые из первых работ по динамическому программированию, содержат в себе и идеи обучения например [12]. В работе [13] приводятся явные аргументы в пользу более тесной связи между динамическим программированием и методами обучения и доказывается, что динамическое программирование имеет прямое отношение к пониманию работы нейронов и когнитивных механизмов.

Явная связь методов динамического программирования с обучением для дискретных стахостических систем отражена в работе Криса Уоткинса 1989 года [14], в которой обучение с подкреплением изложено с позиций формализма МППР. Методы оптимального управления и обучения с подкреплением активно разрабатываются многими исследователями, в особенности Димитрием Бертсеркасом и Джоном Цициклисом, которые предложили термин «нейродинамическое программирование», описывающий комбинацию динамического программирования с нейронными сетями [6]. Еще один термин, широко употребляемый в настоящее время, -– «приближенное (адаптимное) динамическое программирование».

С одной стороны, почти все традиционные методы требуют полного знания об управляемой системе, кажется не совсем верно утверждать, что они так же относятся к обучению с подкреплением. С другой стороны, многие алгоритмы динамического программирования инкрементные и итеративные.

Как и методы обучения, они постепенно приходят к правильному ответу путем последовательных приближений.

1.4 Примеры применения обучения с подкреплением

Первоначально методы RL применялись только к простым задачам, но применение глубоких нейронных сетей позволило применять RL для систем другого уровня сложности. Сейчас RL применяется в самых разных областях: робототехника, финансы, автономные транспортные средства, медицина и здравоохранения, оптимизация процессов и обнаружение неисправностей. Ниже рассмотрены основные области применения RL:

- Промышленная робототехника – активно развивающаяся область применения RL, поскольку является естественным внедрением этой парадигмы в практику [15]. Например, использование глубокого обучения и обучения с подкреплением позволяет обучать роботов, способных захватывать различные объекты - даже те, которые не видны во время обучения. Или другой пример – обучение работа повторять команды за человеком, выполняя перемещение объектов [16]. Данные приложения могут быть использованы при сборке продуктов на сборочной линии.
- Здравоохранение. RL в здравоохранении относится к методам динамического лечения (англ. Dynamic Treatment Regimes), так как алгоритмы RL позволяют находить решения для оптимального лечения пациентам в каждый момент времени. Например, вход алгоритма – набор клинических наблюдений и оценок пациента, выход - варианты лечения для каждого этапа.[17]
- Обработка естественного языка. RL активно применяется при решении таких задач, как построение диалоговых систем [18], резюмированние и сокращение текстов [19] машинный перевод и др.
- Проектирование архитектуры глубоких нейронных сетей. С одной стороны, применение RL для подбора архитектуры – вычислительно затратное решение, но такой подход позволяет создавать лучшие архитектуры глубоких нейронных сетей.
- Автономные транспортные средства. Алгоритмы RL используются как для организации дорожного движения - автономные светофоры, так и для решения задач автономного вождения, например, оптимизация

траектории движения, динамическое определение пути, смена полосы движения, парковка [20].

– Оптимизация производственных процессов. RL применяется для прогнозирования технического обслуживания, диагностики оборудования режиме реального времени и управлять производственной деятельностью, а так же для оптимизации энергопотребления.[21] Например, компания Royal Dutch Shell применяет RL в задачах по разведке и бурению. Алгоритмы RL, обученные на исторических данных бурения, а также дообученные на физических моделях, используются для управления газовыми буровыми установками при их движении по геологической среде.

1.5 Классификация алгоритмов обучения с подкреплением

В данном разделе приведена основополагающая классификация алгоритмов RL. На рис.1.3 представлена лишь одна из возможных таксономий алгоритмов RL. Одна из наиболее важных классификаций основана на наличии доступа или возможность исследования агентом модели среды.

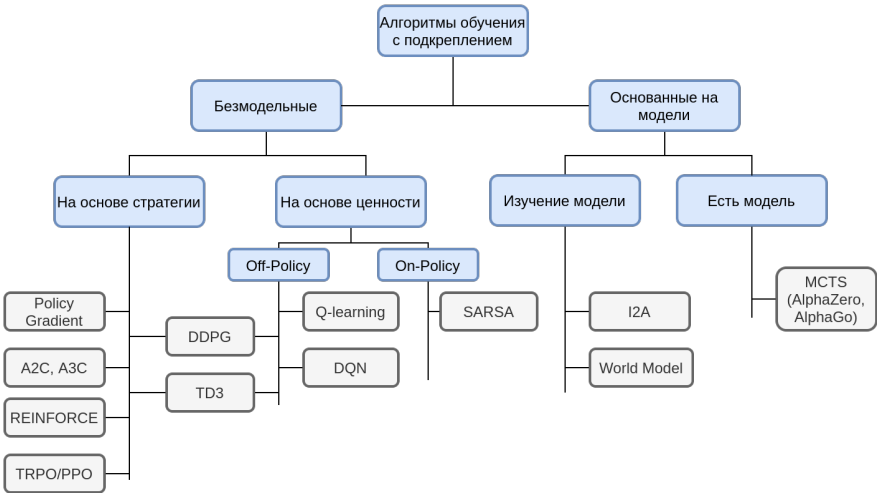

Рисунок 1.3 — Классификация алгоритмов Rl

Алгоритмы модельно-ориентированного обучения (англ. model-based RL) на первом этапе решают задачу идентификации системы либо используют уже составленную модель для решения задач управления. Безмодельное обучение (англ. model-free RL) направленно на прямой поиск управления на основе наблюдений и действий. Другими словами, безмодельные методы направлены на решение задач управления путем исследования системы и улучшения стратегий на основе прошлых вознаграждений и состояний. Модельные алгоритмы

RL имеют такой же недостаток, как и классические подходы в управлении – сложность получения достоверной модели среды (объекта).

Безмодельные методы в свою очередь делятся на два подхода: оптимизация политики и на основе функции ценности. Алгоритмы градиента стратегии (англ. Policy-based) , использую оценки градиента функционала по параметрам политики, что означает, что каждое обновление использует данные, полученные на последней политики. Алгоритмы на основе ценности (англ. value-based) позволяют получать стратегию неявно через теорию оценочных функций. В наиболее часто встречающейся постановке для аппроксимации функции применяются глубокие нейронные сети и другие современные подходы, которые позволяют справиться с высокой дисперсией и общей неустойчивостью. Стоит отметить, что алгоритмы градиента стратегии применимы для сред с непрерывной областью действий, тогда как алгоритмы функций ценности применяются для дискретных систем. Поскольку такие алгоритмы используют целевую функцию основанную на уравнении Беллмана. Модельные и безмодельные алгоритмы не лишены недостатков, поэтому ведется активное развитие гибридных безмодельных алгоритмов RL.

Наколенные данные от эксперта, то есть записи взаимодействия со средой некоторой стратегии, не обязательно оптимальной, могут упростить задачу поиска условного оптимального решения ряда алгоритмов RL. Исходя из способности алгоритма использовать опыт взаимодейтсвия со средой произвольной стратегии, выделяют on-policy и off-policy алгоритмы. Алгоритм off-policy способен использовать для обучения опыт взаимодействия произвольной стратегии. То есть появляется возможность проводить очередной шаг обучения на произвольных траекториях, сгенерированных произвольными стратегиями. Для алгоритма on-policy на очередной итерации требуется опыт взаимодействия конкретной, предоставляемой самим алгоритмом, стратегии. В то же время, накопленные данные от эксперта могут быть использованы для инициализации on-policy алгоритмов. Важно, что off-policy алгоритмы способны на данных произвольного эксперта условно сойтись к оптимуму при достаточном объёме и разнообразии экспертной информации, не требуя дополнительного взаимодействия со средой.

1.6 Методы и алгоритмы обучения с подкреплением

Динамическое программирование. Динамическое программирование (ДП) – один из фундаментальных модельных методов в RL, который основан на урав-

нении Беллмана. Базовыми алгоритмами ДП являются итерирование стратегий (англ., Policy iteration, PI) и итерирование ценности (англ., Value iteration, VI). Алгоритм PI во время одной итерации вычисляет функцию награды для текущей стратегии, потом улучшает эту стратегию, то есть оценивание и улучшение стратегии выполняется циклически, тогда как в алгоритме VI – оценивание и улучшение стратегии выполняется за одно обновление. В литературе по RL процесс вычисления функции ценности состояния V^{π} для произвольной стратегии π называют оценкой стратегии (англ. Policy Evaluation) или задачей прогнозирования.Функция ценности для стратегии вычисляется для того, чтобы далее с помощью нее найти новую, улучшенную стратегию. Процесс формирования новой, улучшающей исходную, стратегии называется улучшением стратегии (англ., Policy Improvement). Так, вычисленное значение функции ценности позволяет улучшить стратегию, в частостии, жадной стратегией:

$$\pi' = argmax_a Q_{\pi}(s,a) = argmax_a \sum_{s',r} p(s'|s,a)[r + \gamma V_p i(s')]$$

Алгоритм итерация по стратегиям представлен на рис.1.4. Алгоритм итерация по ценности представлен на рис.1.5.

1. Инициализация $V(s) \in \mathcal{R}$ и $\pi(s) \in \mathcal{A}(s)$ для всех $s \in \mathcal{S}$
2. Оценка стратегии Пока $\Delta < \theta$
  $v := V_k(s)$
  $V_{k+1}(s) := \sum_{s',r} p(s'|s,\pi(a))[r + \gamma V_k(s')]$
  $\Delta := \max(\Delta, |v - V_{k+1}(s)|)$
Конец цикла
3. Улучшение стратегии
Стратегия устойчива:=истинно
Цикл по $s \in \mathcal{S}$
  $b := \pi(s)$
  $\pi(s) = argmax_a Q_{\pi}(s,a) = argmax_a \sum_{s'r} p(s'|s,a)[r + \gamma V_{k+1}(s)]$
  Если $b \neq \pi(s)$ тогда
  Стратегия устойчива:=ложно
Конец цикла
Если Стратегия устойчива тогда
  выход
иначе переход к шагу 2

Рисунок 1.4 — Алгоритм итерирование стратегии

Недостатком методов ДП является то, что они требуют вычислений в каждом состоянии. В случае, если множество состояний велико, это может

1. Инициализация $V_0(s)$ произвольно для всех $s \in \mathcal{S}$
2. Оценка стратегии
Пока $\Delta < \theta$
$v := V_k(s)$
$V_{k+1}(s) := \sum_{s',r} p(s'|s,\pi(a))[r + \gamma V_k(s')]$
$\Delta := \max(\Delta, |v - V_{k+1}(s)|)$
Конец цикла
3. Улучшение стратегии
$\pi(s) = argmax_a \sum_{s'r} p(s'|s,a)[r + \gamma V_{k+1}(s)]$

Рисунок 1.5 — Алгоритм итерирование ценности

потребовать больших вычислительных ресурсов. Асинхронные алгоритмы ДП обновляют оценки значений только подмножества состояний, а не всего набора на каждой итерации оценки политики. Такие алгоритмы обновляют значения состояний в любом порядке, используя значения других доступных состояний. Как было сказано ранее, PI состоит из двух чередующихся взаимодействующих процессов: оценивание стратегии, которое обеспечивает совместимость функции ценности с текущей стратегией и улучшение стратегии, который делает стратегию жадной к текущей функции ценности. В VI между двумя улучшениями стратегии только одна итерация оценивания стратегии. В асинхронных методах ДП процесс оценивания и учлучшение стратегии разбивается на более крупные чередующиеся итерации.

Одни из главных недостатков ДП – экспоненциальное возрастание сложности вычислений с увеличением числа состояний («Проклятье размерности») и необходимость модели объекта. Несмотря на это, идеи оценки стратегии и итерирования по стратегии лежат в основе почти всех алгоритмов ОП.

В отличии от ДП методы Монте-Карло (МК) не предполагают полного знания о модели. Такие методы предполагают наличие данных – набор состояний, действий и наград, полученных при взаимодействии с средой. Для применения методов МК требуется, чтобы задача была эпизодическая, так как методы МК инкриментны на уровне эпизодов, а не на уровне шагов. В отличии от ДП оценка для каждого состояния в методах МК – независимы. Подробно не будет останавливаться на методах МК и ДП.

Обучение на основе временных различий. Обучение на основе временных различий (англ., Temporal Difference, TD) или TD-обучение – это метод обучения с подкреплением без использования моделей, где агент обучается на каждом

отдельном действии, которое он предпринимает и при этом обновляет знания агента на каждом временном шаге (действии), а не в каждом эпизоде.TD-методы совмещают в себе идеи методов МК и ДП. Коррекция в простейшем TD-методе производится путем улучшения ценности на небольшую величину в направлении оптимального значения:

$$V(s_t) = V(s_t) + \alpha[r + \gamma V(s_{t+1}) - V(s_t)]$$

где α – параметр, который определяет степень изменения ценности состояния при каждом обновлении. Если $a = 0$, то ценность состояния не изменяется. Если же $a = 1$, то ценность состояния будет равна $r + \gamma V(s_{t+1})$ – старая ценность затирается. К алгоритмам TD-методов относятся: Q-learning, SARSA, R-learning, методы исполнитель-критик

На рис.1.6 – 1.7 представлены алгоритмы Q-learning, SARSA. Алгоритм Q-learning – это табличный безмодельный off-policy алгоритм RL. Это метод основан на временных разностей для вычисления оптимальной Q-функции с $\varepsilon-$жадной стратегией исследования, то есть агент выбирает случайное действие с вероятностью ε, но использует известное лучшее действие. Алгоритм наследует от TD-обучения характеристики одношагового обучения, такие как возможность обучаться на каждом шаге и способность обучаться на опыте, не имея модели окружающей среды. Q-learning отличается от SARSA прежде всего тем, что это алгоритм с разделенной стратегией. Разделенная стратегия означает, что обновление производится независимо от того, какая стратегия использовалась для накопления опыта, то есть алгоритмы с разделенной стратегией могут использовать прежний опыт для улучшения стратегии. Стратегия, которая применяется для улучшение стратегии – целевая, а стратегия для взаимодействия с окружающей средой – поведенческая. Алгоритм состоит из следующих шагов: 1. Инициализация Q-таблицы с нулевыми значениями, то есть в начальный момент времени все стратегии равновероятны и равноценны. 2. Выбор действия с наибольшей ценностью. 3. Отправка на вход среды выбранного действие, на выходе получаем вознаграждение. 4. Обновление Q-таблицы с учетом полученного вознаграждения. Гиперпараметрами алгоритма являются $\alpha \in (0,1]$ – параметр экспоненциального сглаживания, $\varepsilon > 0$ – параметр исследования.

Разница между этими двумя алгоритмами в том, что SARSA выбирает действие, соответствующее той же текущей политике, и обновляет его Q-значения, тогда как Q-обучение выбирает жадное действие, то есть действие, которое

Инициализация $Q(s,a)$ произвольно для всех $s \in \mathcal{S}$, $\alpha \in \mathcal{A}$
Цикл по эпизодам
  Инициализация s
  Цикл по шагам эпизодов k
  Выбор a_k: с вероятностью ε принимаем $a_k \sim Uniform(\mathcal{A})$
  иначе $a_k = argmaxQ(s_k, a_k)$
  Выполнение действие a_k
  Нахождение r_k, s_{k+1}
  Обновление
  $Q(s,a) \leftarrow Q(s,a) + \alpha[r_k + \gamma \max_{a_{k+1}} Q(s_{k+1}, a_{k+1}) - Q(s_k,a_k)]$
  Конец цикла
Конец цикла

Рисунок 1.6 — Алгоритм Q-learning

Инициализация $Q(s,a)$ произвольно для всех $s \in \mathcal{S}$, $\alpha \in \mathcal{A}$
Цикл по эпизодам
  Считывание s_0, находим $a_k \sim Uniform(\mathcal{A})$
  Цикл по шагам эпизодов k
  Нахождение r_k, s_{k+1}
  Выбор a_{k+1}: с вероятностью ε принимаем $a_{k+1} \sim Uniform(\mathcal{A})$
  иначе $a_{k+1} = argmaxQ(s_{k+1}, a_{k+1})$
  Выполненине действия a_k
  Обновление $Q(s,a) \leftarrow Q(s,a) + \alpha[r_k + \gamma Q(s_{k+1}, a_{k+1}) - Q(s_k,a_k)]$
  Конец цикла
Конец цикла

Рисунок 1.7 — Алгоритм SARSA

дает максимальное значение Q для состояния, то есть оно следует оптимальной политике.

Апроксимация V-функции.

Главным недостатком перечисленных выше методов является необходимость хранить в памяти большие объемы данных при увеличении сложности объекта, а так же сложность работы с непрерывным пространством действий. Необходимо ввести аппроксимацию V-функции (Q-функции), что позволит представить функцию в ограниченной области определения, располагая памятью фиксированного объема.Применение аппроксимации функций позволяет заменить пространство состояний набором признаков, порождаемых по исходным со- стояниям

Основная идея аппроксимации функций – воспользоваться набором признаков для оценки значений V-функции (Q-функции). Есть несколько способов

отобразить признаки на значения функции, например линейная аппроксимация, решающие деревья, алгоритм ближайших соседей, искусственные нейронные сети. В случайно линейной аппроксимации функция ценности состояний записывается в виде взвешенной суммы признаков. Как можно ожидать, нейронные сети используются чаще других подходов. В частности, используются глубокие нейронные сети (ГНС).

Вывод по главе 1. В ходе анализа и изучения области обучения с подкреплением, сделаны следующие выводы:

- МППР – нотация для представления задачи обучения с подкреплением;
- Функция ценности и функция ценности действия – основополагающие термины для дальнейших исследований;
- Динамическое программирование рассматривается как один из главных способов обучения агента;
- Область управления и область обучения с подкреплением связаны математической базой, а именно уравнение Беллмана;
- Примемения методов обучения с подкреплением распространено в области рекомендательных систем и игр;

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Introduction to reinforcement learning. T. 135 / R. S. Sutton, A. G. Barto [и др.]. — MIT press Cambridge, 1998.

2. *Sutton R. S.*, *Barto A. G.* Reinforcement learning: An introduction. — MIT press, 2018.

3. Reinforcement learning: A survey / M. L. Littman, A. W. Moore [и др.] // Journal of artificial intelligence research. — 1996. — Т. 4, № 1. — С. 237—285.

4. *Otterlo M. van* Reinforcement learning: State-of-the-Art. — Springer Berlin Heidelberg, 2012.

5. *Lewis F. L.*, *Vrabie D.*, *Vamvoudakis K. G.* Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controllers // IEEE Control Systems Magazine. — 2012. — Т. 32, № 6. — С. 76—105.

6. *Bertsekas D. P.*, *Tsitsiklis J. N.* Neuro-dynamic programming. — Athena Scientific, 1996.

7. *Powell W. B.* Approximate Dynamic Programming: Solving the curses of dimensionality. T. 703. — John Wiley & Sons, 2007.

8. A survey on policy search for robotics / M. P. Deisenroth, G. Neumann, J. Peters [и др.] // Foundations and trends in Robotics. — 2013. — Т. 2, № 1—2. — С. 388—403.

9. *Bellman R.* Dynamic Programming. — 1-е изд. — Princeton, NJ, USA: Princeton University Press, 1957.

10. *Bellman R.* A Markovian decision process // Journal of mathematics and mechanics. — 1957. — Т. 6, № 5. — С. 679—684.

11. *Howard R. A.* Dynamic Programming and Markov Processes. — Cambridge, MA: MIT Press, 1960.

12. *Bellman R.*, *Dreyfus S.* Functional approximations and dynamic programming // Mathematical Tables and Other Aids to Computation. — 1959. — С. 247—251.

13. *Werbos P. J.* Building and Understanding Adaptive Systems: A Statistical/Numerical Approach to Factory Automation and Brain Research // IEEE Transactions on Systems, Man, and Cybernetics. — 1987. — Т. 17, № 1. — С. 7—20. — DOI 10.1109/TSMC.1987.289329.

14. *Watkins C. J. C. H.* Learning from delayed rewards. — 1989.

15. *Kober J.*, *Bagnell J. A.*, *Peters J.* Reinforcement learning in robotics: A survey // The International Journal of Robotics Research. — 2013. — Т. 32, № 11. — С. 1238—1274.

16. Meta Learning via Learned Loss / S. Bechtle [и др.] // International Conference on Pattern Recognition, ICPR, Italy, January 10-15, 2021. — 2021.

17. *Yu C.*, *Liu J.*, *Nemati S.* Reinforcement learning in healthcare: A survey // arXiv preprint arXiv:1908.08796. — 2019.

18. Deep reinforcement learning for dialogue generation / J. Li [и др.] // arXiv preprint arXiv:1606.01541. — 2016.

19. *Paulus R.*, *Xiong C.*, *Socher R.* A deep reinforced model for abstractive summarization // arXiv preprint arXiv:1705.04304. — 2017.

20. Exploring applications of deep reinforcement learning for real-world autonomous driving systems / V. Talpaert [и др.] // arXiv preprint arXiv:1901.01536. — 2019.

21. Optimization of global production scheduling with deep reinforcement learning / B. Waschneck [и др.] // Procedia Cirp. — 2018. — Т. 72. — С. 1264—1269.