Серия «Математика»
2021. Т. 38. С. 65—83

Онлайн-доступ к журналу:
http://mathizv.isu.ru

ИЗВЕСТИЯ
Иркутского
государственного
университета

УДК 004.85
MSC 22E05
DOI https://doi.org/10.26516/1997-7670.2021.38.65

Глубокое обучение адаптивных систем управления на основе логико-вероятностного подхода

А. В. Демин[1]

[1] *Институт систем информатики им. А. П. Ершова СО РАН, Новосибирск, Российская Федерация*

Аннотация. Проблема автоматического выделения подцелей в настоящее время является одной из наиболее актуальных в задачах адаптивного управления, в частности в задачах обучения с подкреплением (Reinforcement Learning). В данной работе предложен логико-вероятностный подход к построению адаптивных обучаемых систем управления, способный обнаруживать глубокие неявные подцели. Подход использует идеи нейрофизиологической теории функциональных систем для организации схемы управления, и логико-вероятностные методы машинного обучения для обучения правил работы системы и выявления подцелей. Работоспособность предложенного подхода демонстрируется на примере решения трехэтапной задачи фуражирования, содержащей две вложенные неявные подцели.

Ключевые слова: система управления, машинное обучение, обнаружение закономерностей, обучение с подкреплением.

1. Введение

В настоящее время понятие «глубокое обучение» (deep learning) устойчиво вошло в терминологию современных исследователей. В области Data Science термин «глубокое обучение» чаще всего ассоциируется с технологией обучения глубоких многослойных нейронных сетей. Между тем понятие «глубокое обучение» можно рассматривать в более широком смысле: как подход к машинному обучению, позволяющий достичь большей эффективности за счет представления данных в виде иерархии вложенных концепций, в которой каждая концепция определяется в терминах более простых концепций, а более абстрактные представления вычисляются в терминах менее абстрактных [7]. К примеру, в работах [14; 15] представлен способ построения глубоких моделей, названный deep forest и основанный на иерархии random-forest классификаторов.

Что касается проблематики адаптивных систем управления, в частности обучение агента путем его взаимодействия с окружающей средой методом «проб и ошибок», которое носит название «обучение с подкреплением» (reinforcement learning, RL) [10], то здесь в последнее время наиболее впечатляющие результаты показывает подход «глубокое обучение с подкреплением» (deep reinforcement learning, DRL). К примеру, алгоритмы DRL позволили машине автоматически обучиться играть в игры Atari, получая на вход лишь сырые пиксельные данные [8], а также побить чемпионов мира в игру го [9]. Однако при более детальном рассмотрении можно заметить, что своим успехом многие современные эффектные DRL приложения в основном обязаны использованию моделей глубоких нейронных сетей для анализа входной сенсорной информации. В частности, Atari DRL эффективно использует глубокие сети для преобразования сырых пиксельных данных в сжатое представление, пригодное для использования reinforcement learning методами. Безусловно, объединение классических методов обучения с подкреплением с глубокими нейронными сетями является большим прогрессом, однако при этом по-прежнему остались нерешенными ряд традиционных проблем обучения с подкреплением, в частности проблема разбиения задачи на подзадачи (обнаружение подцелей), проблема быстрого набора эффективного для обучения опыта, проблема быстрого падения эффективности обучения с ростом размерности пространства состояний и действий, проблема эффективного обучения при редких вознаграждениях от среды.

Проблема автоматического выделения подцелей в настоящее время является одной из наиболее актуальных. Способность объединять серии элементарных действий в более крупные группы, решающие конкретные подзадачи, дает нам возможность рассмотреть решение основной задачи в укрупненном масштабе, в терминах этих подзадач. Это не

только позволяет быстрее обучаться, но и дает возможность эффективно решать задачи, которые нельзя решить в масштабе элементарных действий. Здесь можно легко провести аналогию между иерархией целей, где вышестоящие цели оперируют нижестоящими подцелями, и иерархией вложенных концепций в глубоком обучении, где более абстрактные концепции определяется в терминах менее абстрактных. По нашему мнению, понятие глубокого обучения применительно к задачам адаптивного управления состоит не только и не столько в глубоком анализе входной сенсорной информации, но и в глубоком анализе опыта взаимодействия системы с внешней средой с целью представления этого опыта в виде эффективной иерархии целей и подцелей.

В настоящее время в рамках общего раздела обучения с подкреплением (reinforcement learning, RL) выделилось направление «иерархическое обучение с подкреплением» (hierarchical reinforcement learning, HRL) [5;6], которое объединяет различные подходы к группировке элементарных действий для более эффективного обучения агента и решения им задач. Однако одна из основных проблем большинства этих подходов заключается в необходимости заранее задавать подцели. Таким образом, задача автоматического обнаружения подцелей по-прежнему остается крайне актуальной.

В данной работе предлагается альтернативный подход, который, с одной стороны, использует идеи организации управления из нейрофизиологической теории функциональных систем, а с другой — логико-вероятностный методы машинного обучения для тренировки агента. Одно из преимуществ использования логико-вероятностных методов состоит в том, что обнаруженные в результате обучения закономерности имеют явную форму, т. е. представлены в виде логических формул. Это позволило нам предложить автоматический способ извлечения подцелей, основанный на анализе этих закономерностей. Учитывая сказанное выше замечание относительно понятия глубокого обучения применительно к адаптивному управлению, можно утверждать, что предлагаемый нами подход, позволяющий автоматически обнаруживать скрытые подцели и выстраивать их в иерархию, имеет право называться «глубоким».

2. Экспериментальная среда

Для экспериментальной проверки эффективности предложенного подхода и, в частности, метода извлечения подцелей, мы предлагаем использовать тестовую задачу, которую условно назвали «многоэтапная задача фуражирования».

Существует известная классическая задача фуражирования, которая заключается в том, что агент должен обучиться эффективному по-

иску пищевых объектов в некотором плоском виртуальном мире. Данная задача является одноэтапной, поскольку в ней не могут быть выделены подцели. В связи с этим мы усложнили классическую задачу, сделав ее многоэтапной. N-этапная задача отличается тем, что виртуальный мир содержит объекты N типов, и для поглощения объекта типа k ($k = 2, ..., N$), необходимо найти и поглотить объект типа $(k-1)$. Первый тип объекта ($k = 1$) начинает цепочку, поэтому может быть поглощен после обнаружения без дополнительных условий. При этом если агент поглотил объект типа k ($k = 1, ..., N-1$), то после он может поглотить только объект типа $(k+1)$. Задача считается выполненной, если агент поглотит объект последнего типа ($k = N$), а это возможно только после выполнения всей цепочки последовательных поглощений. Данная задача удобна тем, что содержит в себе очевидную иерархию подцелей, и поэтому может являться хорошим тестом для оценки эффективности обнаружения подцелей. В данной работе в качестве эксперимента мы будем решать трехэтапную задачу.

Опишем более подробно экспериментальную среду и условия задачи. Виртуальная среда представляет собой прямоугольное поле, размером 25 на 25 клеток. Каждая клетка может быть либо пустой, либо содержать один из трех типов объектов: «еда-1» (пищевой объект первого типа), «еда-2» (второго типа) или «еда-3» (третьего). Агент может перемещаться по полю, совершая три типа действий: шаг на клетку вперед («шаг»), поворот налево («налево»), поворот направо («направо»).

В начале эксперимента по полю случайным образом располагается одинаковое количество пищевых объектов всех трех типов. Чтобы поглотить объект агенту достаточно шагнуть на клетку, в которой он располагается, и если при этом будут выполнены описанные выше условия, связанные с последовательностью поглощений, то клетка будет очищена и новый объект того же типа случайным образом появиться в другом месте поля. Таким образом, суммарное количество пищевых объектов в виртуальном мире всегда остается постоянным.

Для ориентации в виртуальном мире агент имеет десять сенсоров, девять из которых информируют его о состоянии окружающих клеток: «впереди-слева», «впереди», «впереди-справа», «слева», «центр», «справа», «сзади-слева», «сзади», «сзади-справа» (рис. 1). Каждый из этих сенсоров информирует о типе объекта, находящегося в соответствующей клетке, и может принимать следующие значения: «пусто», «еда-1», «еда-2», «еда-3». Еще один, десятый, сенсор информирует агента о факте совершения поглощения в текущем такте, и принимает два значения: «да» или «нет».

Отдельно отметим, что в данной постановке задачи агент не содержит никаких дополнительных сенсоров, которые бы в явном виде информировали его о том, какой последний тип пищевого объекта он поглотил. Это существенным образом усложняет задачу по сравнению с

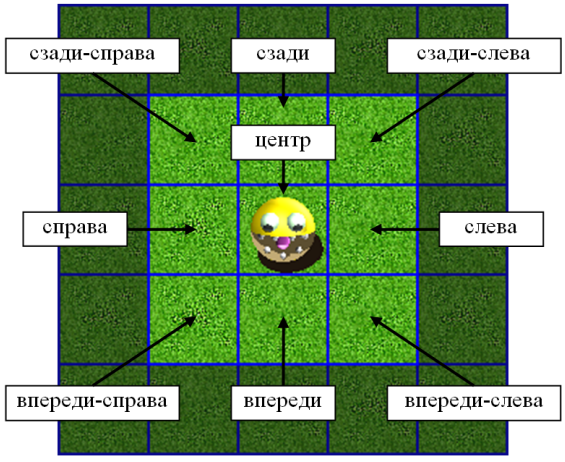

Рис. 1. Сенсорное поле агента

той, которая была в работах [3;11–13], где решалась двухэтапная задача, и агент имел дополнительный сенсор, который сообщал ему о том, был ли поглощен объект первого типа.

3. Модель системы управления

Архитектура предложенной системы управления основана на теории функциональных систем, разработанной известным русским нейрофизиологом П. К. Анохиным [1]. Согласно этой теории единицей деятельности организма является функциональная система, формирующаяся для достижения полезных организму результатов (например, удовлетворение потребностей). Организация функциональных систем при целенаправленном поведении осуществляется в соответствии с двумя правилами: последовательностью и иерархией результатов. Последовательность результатов выстраивается по принципу «доминанты»: доминирующая потребность возбуждает доминирующую функциональную систему и строит поведенческий акт, направленный на ее удовлетворение. По отношению к доминирующей функциональной системе все остальные функциональные системы выстраиваются в иерархию по принципу «иерархии результатов»: когда результат деятельности одной функциональной системы входит в качестве компонента в результат деятельности другой.

Предполагаем, что системы управления функционирует в дискретном времени $t = 1, .., n$, имеет некоторый набор сенсоров для восприятия информации об окружающей среде и набор возможных действий, которые она может совершать. Также считаем, что статистические дан-

ные о взаимодействии системы с окружающей средой хранятся в массиве данных, где для каждого момента времени t записана вся сенсорно-моторная информация системы: показания сенсоров и выбранное действие.

Для обнаружения закономерностей, описывающих работу системы, предлагается использовать логико-вероятностные методы извлечения знаний из данных, основанные на идеях семантического вероятностного вывода и адаптированные для задач управления [2–4]. Для этого вся сенсорно-моторная информация представляется в виде логических предикатов, в терминах которых осуществляется поиск управляющих правил.

Введем наборы базовых логических предикатов следующего типа:

$S_i \in \mathbb{S}$ — сенсорные предикаты из заданного набора предикатов $\mathbb{S}$, описывающие состояние сенсоров агента. Запись $S_i(t)$ означает, что состояние сенсоров агента в момент времени t удовлетворяет условию предиката S_i.

$A_i \in \mathbb{A}$ — активирующие предикаты из заданного набора предикатов $\mathbb{A}$, описывающие действия агента. Запись $A_i(t)$ означает, что в момент времени t агент выполнил действие, удовлетворяющее условию предиката A_i. Для простоты в дальнейшем будем полагать, что каждый предикат A_i взаимооднозначно соответствует одному действию из набора возможных действий системы.

Введем понятие предиката-цели G, как условие достижения некоторого целевого состояния, описываемого конъюнкцией сенсорных предикатов: $G = S_{i1} \wedge S_{i2} \wedge \ldots \wedge S_{ik}$.

Также введем понятие предиката-результата достижения цели G:

$$Res(G)(t) = \exists x : (t_0 \leq x \leq t) \wedge G(x) = true$$

Данный предикат информирует о том, что цель G была достигнута на промежутке времени между тактами $[t_0, t]$ и ее результат актуален на момент времени t. Точка t_0, по сути, является моментом отсечения, после которого мы считаем, что все достижения цели G уже являются неактуальными на данный момент. В данной работе мы предлагаем в качестве t_0 использовать момент времени последнего достижения любой цели, стоящей выше в иерархии подцелей над G. Фактически, это означает, что при достижении какой-либо вышестоящей цели все результаты нижестоящих подцелей «обнуляются».

Архитектура системы управления представляет собой иерархию функциональных систем, при которой функциональные системы верхнего уровня ставят цели системам нижнего уровня. Отдельная функциональная система F определяется следующим набором:

$$F = \langle G, RES, REG \rangle$$

где G — предикат-цель, описывающий цель, достижение которой является задачей данной функциональной системы.

RES — множество предикатов-результатов целей, соответствующих функциональным системам, подчиненным данной системе.

REG — множество правил, описывающих работу данной функциональной системы и имеющих вид:

$$\underbrace{Sit_1(t)}_{situation}, \underbrace{A_1(t)}_{action} \to \underbrace{Sit_2(t+1)}_{situation}, \underbrace{A_2(t+1)}_{action} \to \dots \underbrace{Sit_m(t+m)}_{situation}, \underbrace{A_m(t+m)}_{action} \to$$

$$\to \underbrace{G(t+m+1)}_{goal} \tag{1}$$

где $A_i \in \mathbb{A}$, $Sit_i = S_1^i \wedge \dots \wedge S_n^i \wedge Res(G_1) \wedge \dots \wedge Res(G_k)$ — описание ситуации, конъюнкция из сенсорных предикатов и предикатов-результатов, $S_i \in \mathbb{S}$, $Res(G_i) \in RES$. Запись $Sit(t)$ означает, что в момент времени t показания сенсоров и достигнутые результаты удовлетворяют условию Sit.

Данные правила предсказывают, что если из ситуаций $Sit_1, ..., Sit_m$ последовательно выполнять действия $A_1, ..., A_m$, переходя после выполнения каждого действия в соответствующую ситуацию, указанную в правиле, то с некоторой вероятностью p будет достигнута вышестоящая цель G.

Рассмотрим, каким образом происходит работа системы управления в целом. Каждый такт работы системы включает три этапа: 1) прогнозирование и формирование плана действий; 2) выполнение действий и 3) оценка результатов.

Во время первого этапа происходит инициация запроса к функциональной системе, находящейся на вершине иерархии функциональных систем, о достижении основной цели. Ответом на запрос будет являться максимально вероятный прогноз достижимости этой цели и соответствующий план действий. Рассмотрим этот процесс подробнее.

Предположим, что в некоторый момент времени t перед функциональной системой $F = \langle G, RES, REG \rangle$ ставится запрос о достижении цели G. На вход функциональной системы подается также информация об окружающей среде в виде набора значений сенсорных предикатов $S_1, S_2, ..., S_n$, описывающих текущую ситуацию.

В процессе вычисления прогноза функциональная система F отбирает из множества своих правил REG все правила, применимые в текущей ситуации. То есть отбираются все правила вида (1), у которых набор сенсорных предикатов $S_1^1, ..., S_n^1$ в описании ситуации $Sit_1 = S_1^1 \wedge \dots \wedge S_n^1 \wedge Res(G_1) \wedge \dots Res(G_k)$ выполнен в текущей ситуации. Далее для каждого отобранного правила R рассчитывается оценка вероятности

$f(R)$ достижения цели G по формуле:

$$f(R) = \begin{cases} p(R), \ \textbf{если } \forall i : Res(G_i) = true, i = 1...k, G_i \in Sit_1 \\ p(R) \cdot \prod\limits_{i \in I} f(G_i), \ \textbf{если } \exists i : Res(G_i) = false, i \in I, G_i \in Sit_1. \end{cases}$$

где $p(R)$ — условная вероятность данного правила; $f(G_i)$ – оценки вероятностей достижения подцелей G_i из текущей ситуации.

Расчет оценок $f(G_i), i \in I$ осуществляется рекурсивно путем отправки аналогичных запросов соответствующим функциональным системам, находящимся ниже по иерархии и реализующим эти подцели. Эти запросы активируют в подчиненных функциональных системах аналогичные процессы вычисления прогнозов в текущей ситуации. Если какая-то из подцелей не может быть выполнена в данной ситуации (нет правил предсказывающих достижение подцели в данной ситуации), то в ответ на запрос возвращается отказ и правило, инициировавшее запрос, будет исключено из рассмотрения.

После вычисления оценок для всех отобранных правил функциональная система выбирает правило R_{best}, имеющее максимальную оценку вероятности. Если окажется, что невозможно выбрать ни одно правило, то считаем, что $R_{best} = \varnothing$.

Таким образом, по завершению первого этапа все функциональные системы будут иметь наилучшие правила, определяющие действия по достижению их целей. Совокупность всех этих правил фактически будет являться планом действий системы управления по решению задачи.

Во время второго этапа система в соответствии с выбранным планом осуществляет выполнение действий. Процесс исполнения запускается путем отправки команды доминирующей функциональной системе, которая выполняет действия в соответствии с выбранным на предыдущем этапе наилучшим правилом R_{best}. Если $R_{best} \neq \varnothing$, т.е. $R_{best} = Sit_1, A_1 \to Sit_2, A_2 \to ... \to Sit_m, A_m \to G$ и все результаты из Sit_1 достигнуты, то будет выполнено первое действие A_1.

Если же описание ситуации Sit_1 содержит какие-либо невыполненные результаты $\{Res(G_i)\}, i \in I$, то аналогичная команда на исполнение будет передана нижележащей функциональной системе, реализующей достижение первой подцели G_1 из списка $\{Res(G_i)\}, i \in I$. Эта функциональная система в соответствии со своим наилучшим правилом либо выполнит действие, либо передаст управление еще ниже по иерархии. Данный процесс передачи запросов вниз по иерархии будет распространяться до тех пор, пока какая-либо система не выполнит какое-нибудь действие $A \in \mathbb{A}$.

Если у доминирующей функциональной системы отсутствует наилучшее правило, т. е. $R_{best} = \varnothing$, то система выберет действие случайным образом из арсенала имеющихся действий.

После совершения действия система перейдет в новую ситуацию, будут обновлены показания сенсоров и запустится третий этап, в ходе которого будет произведена оценка полученных результатов. Для каждой функциональной системы $F_i = \langle G_i, RES_i, REG_i \rangle$, будет произведено вычисление значения ее предиката-цели G_i. Если предикат-цель выполняется в новой ситуации, значит, что данная система достигла свою цель.

На этом завершится один такт работы системы управления.

4. Генерация правил

Обучение системы управления заключается в обнаружении для каждой функциональной системы $F = \langle G, RES, REG \rangle$ множества правил REG на массиве данных истории деятельности агента. Но прежде чем приступить к описанию процедуры генерации правил, остановимся на вопросе выбора формы правил (1).

В предыдущих работах [3; 11–13] были использованы правила вида

$$S_1, ..., S_n, Res(G_1), ..., Res(G_k), A \to G$$

которые прогнозируют оценку результата своего действия, если оно будет совершенно в определенной ситуации. Недостаток подобного подхода состоит в том, что для обучения оптимальному поведению системе необходимо встретить в процессе своего обучения все возможные пути достижения цели, чтобы построить оценки достижимости цели из различных ситуаций. В сложных средах это может потребовать непомерно большого времени для обучения.

Другой подход может состоять в том, чтобы сначала построить вероятностную модель среды как набор закономерностей, прогнозирующих реакцию среды при совершении действий:

$$S_1^1, ..., S_n^1, Res(G_1), ..., Res(G_m), A \to S_1^2, ..., S_l^2, Res(G_1), ..., Res(G_k) \quad (2)$$

Данное правило предсказывает, что если в ситуации, описываемой набором предикатов $S_1^1, ..., S_n^1, Res(G_1), ..., Res(G_m)$, совершить действие A, то система перейдет в ситуацию $S_1^2, ..., S_l^2, Res(G_1), ..., Res(G_k)$ с некоторой вероятностью p.

А затем использовать эти закономерности для построения пути достижения цели по принципу обратного вывода, получая правила вида (1), состоящие из последовательностей переходов между ситуациями:

$$Sit_1, A_1 \to Sit_2, A_2 \to ... \to Sit_m, A_m \to G$$

Основное преимущество данного подхода заключается в значительном увеличении скорости обучения, поскольку таким образом могут

быть сгенерированы пути достижения цели, которые не встречались системе ранее в процессе обучения.

Однако у этого подхода также есть существенный недостаток: множество закономерной вида (2), описывающих модель среды, будет экспоненциально возрастать с ростом количества сенсорных предикатов. При этом также будет возрастать сложность нахождения эффективного набора правил вида (1).

В данной работе мы предлагаем подход, который позволяет воспользоваться преимуществами наличия вероятностной модели среды и при этом избежать генерации слишком большого набора правил. В нашем подходе мы не будем пытаться обнаружить все правила вида (2), прогнозирующие реакцию среды, а будем целенаправленно искать только такие, которые помогают в достижении цели, и затем уже из этих правил будем строить правила вида (1), определяющие работу функциональных систем.

Перейдем к описанию процедуры генерации множества правил REG для функциональной системы $F = \langle G, RES, REG\rangle$. Идея алгоритма состоит в том, чтобы шаг за шагом наращивать правила вида (1), начиная с правил, содержащих один переход в ситуацию — цель G, последовательно добавляя в правила новые переходы между ситуациями, которые определяются закономерностями вида (2). Для нахождения всех способов перехода между ситуациями используется другой алгоритм, основанный на идеях семантического вероятностного вывода, описанного в работах [2;4]. Последовательно применяя данный алгоритм для нахождения цепочек переходов между ситуациями, можно получить правила вида (2), определяющие различные способы достижения конечной цели G.

Сначала рассмотрим алгоритм, реализующий поиск закономерностей вида (2). Для удобства изложения будем через $P_i \in \mathbb{S} \cup RES$ обозначать предикаты из совокупности сенсорных предикатов $\mathbb{S}$ и предикатов-результатов RES.

Введем ряд формальных определений.

Определение 1. *Подправилом правила* $R_1 = P_1^1, ..., P_n^1, A \to P_1^2, ..., P_l^2$ *будем называть любое правило* $R_2 = P_1^3, ..., P_m^3, A \to P_1^2, ..., P_l^2$, *для которого выполнено условие* $\{P_1^3, ..., P_m^3, A\} \subset \{P_1^1, ..., P_n^1, A\}$.

Определение 2. *Правило* $R = P_1^1, ..., P_n^1, A \to P_1^2, ..., P_l^2$ *будет являться вероятностной закономерностью, если оно удовлетворяет следующим условиям:*

- *Вероятность* $p(P_1^1, ..., P_n^1, A)$ *определена и* $p(P_1^1, ..., P_n^1, A) > 0$.
- *Условная вероятность правила больше условной вероятности каждого из его подправил, т. е.* $\forall \{P_1^3, ..., P_m^3, A\} \subset \{P_1^1, ..., P_n^1, A\}$,

$$p(P_1^2, ..., P_l^2 | P_1^3, ..., P_m^3, A) < p(P_1^2, ..., P_l^2 | P_1^1, ..., P_n^1, A)$$

Определение 3. *Правило* $R_2 = P_1^1, ..., P_n^1, P_{n+1}^1, A \to P_1^2, ..., P_l^2$ *будем называть уточнением правила* $R_1 = P_1^1, ..., P_n^1, A \to P_1^2, ..., P_l^2$, *если оно получено добавлением в посылку правила* R_1 *любого предиката* $P_{n+1} \in \mathbb{S} \cup RES$, *не содержащегося в правиле* R_1.

Алгоритм 1. Входными параметрами алгоритма являются целевая ситуация $P_1^2, ..., P_l^2$, правила перехода в которую мы хотим найти; множество предикатов $\mathbb{S} \cup RES \cup \mathbb{A}$ и параметр глубины базового перебора d, где $d \geq 1$ — натуральное число.

- На первом шаге генерируем множество RUL_1 всех правил единичной длины, имеющих следующий вид $A_i \to P_1^2, ..., P_l^2$, $A_i \in \mathbb{A}$. Все правила RUL_1 проходят проверку на выполнение условий принадлежности к закономерностям. Правила, прошедшие проверку, будут являться закономерностями. Обозначим через REG_1 множество всех закономерностей, обнаруженных на первом шаге.
- На шаге $k \leq d$ генерируется множество $RUL_k = Spec(RUL_{k-1})$ всех уточнений правил, сгенерированных на предыдущем шаге. Все правила из RUL_k проходят проверку на выполнение условий принадлежности к закономерностям. Обозначим через $MREG_k$ полученное множество закономерностей.
- На шаге $l > d$ генерируется множество $RUL_l = Spec(MREG_{l-1})$ уточнений всех закономерностей, обнаруженных на предыдущем шаге. Все правила из RUL_l проходят проверку на выполнение условий принадлежности к закономерностям. Обозначим $MREG_l$ — множество всех закономерностей, обнаруженных на данном шаге.
- Алгоритм останавливается на шаге $m > d$, когда не обнаружено новых закономерностей, $MREG_m = \varnothing$.
- Результирующее множеством закономерностей является объединение всех множеств обнаруженных закономерностей $MREG = \bigcup\limits_i MREG_i$.

Шаги алгоритма $k \leq d$ соответствуют базовому перебору, а шаги $k > d$ – дополнительному перебору.

Преимущества использования семантического вероятностного вывода заключается в том, что в результате мы получаем оптимальный набор закономерностей с минимальным описанием, которые максимально точно прогнозирует вероятность перехода в целевую ситуацию. Это позволяет избежать проблемы непомерного разрастания количества правил, описывающих модель среды, о которой упоминалось выше.

Прежде чем перейти к описанию алгоритма генерации правил вида (2), введем ряд определений.

Определение 4. *Длиной правила* $R = Sit_1, A_1 \to Sit_2, A_2 \to ... \to Sit_m, A_m \to G$ *будем называть величину* $len(R) = m$, *равную количеству переходов между ситуациями.*

Определение 5. *Правило* $R = Sit_1, A_1 \to Sit_2, A_2 \to ... \to Sit_m$, $A_m \to G$ *будем называть корректным, если для любого перехода* Sit_k, $A \to Sit_{k+1}$, *входящего в правило* R, *если ситуация* Sit_{k+1} *из заключения перехода содержит предикаты-результаты*

$$Res(G_1), ..., Res(G_d) \subseteq Sit_{k+1},$$

то ситуация Sit_k *из посылки этого перехода также содержит эти предикаты-подцели* $Res(G_1), ..., Res(G_d) \subseteq Sit_k$ *и никакие другие.*

Данное требование корректности фактически означает требование сохранения результатов достижения подцелей при переходах между ситуациями.

Определение 6. *Правило* $R_2 = Sit_1, A_1 \to Sit_2, A_2 \to ... \to Sit_m$, $A_m \to G$ *будем называть уточнением правила* $R_1 = Sit_2, A_2 \to ... \to Sit_m, A_m \to G$, *если оно*

- *является корректным;*
- *получено добавлением в начало правила* R_1 *перехода* $Sit_1, A_1 \to Sit_2$.

Определение 7. *Корректное правило* $R_2 = Sit_k, A_k \to ... \to Sit_m$, $A_m \to G^2$ *будем называть альтернативой корректного правила* $R_1 = Sit_h, A_h \to ... \to Sit_n, A_n \to G^1$, *если* $R_2 \neq R_1$, $G^2 = G^1$ *и* $Sit_k \subseteq Sit_h$.

То есть альтернативное правило — это другое правило, которое приводит к той же цели из той же либо более общей ситуации, что и первое правило, но другим образом (другой цепочкой переходов).

По аналогии с определением 3 введем для правил вида (2) определение закономерности.

Определение 8. *Правило* $R = Sit_1, A_1 \to Sit_2, A_2 \to ... \to Sit_m$, $A_m \to G$ *будем называть закономерностью, если оно:*

- *является корректным;*
- *условная вероятность этого правила больше условной вероятности любого другого альтернативного правила меньшей длины.*

Обозначим через $Discovery(Sit)$ вызов алгоритма 1 для обнаружения всех закономерностей перехода в ситуацию Sit. Алгоритм вернет множество всех обнаруженных закономерностей, которое будет обозначать через $MREG(Sit)$: $MREG(Sit) = Discovery(Sit)$.

Также введем функцию $Estimate(R)$ расчета оценки условной вероятности правила R через условные вероятности переходов между ситуациями: если $R = Sit_1, A_1 \to Sit_2, A_2 \to ... \to Sit_m, A_m \to G$, то $Estimate(R) = p(Sit_1, A_1 \to Sit_2) \cdot ... \cdot p(Sit_m, A_m \to G)$.

Алгоритм 2. Входными параметрами алгоритма являются цель G, правила достижения которой мы хотим обнаружить; множество предикатов $\mathbb{S} \cup RES \cup \mathbb{A}$ и параметр глубины перебора d, где $d \geq 1$ – натуральное число.

– На первом шаге генерируем множество $MREG_1(G) = Discovery(G)$ всех закономерностей перехода в целевую ситуацию G. Затем для каждой закономерности $MR = Sit, A \to G$, $MR \in MREG_1(G)$, создаем правило $R = Sit, A \to G$, содержащее один переход в целевую ситуацию. Для правила R вычисляем его условную вероятность $p(R) = Estimate(R)$ и проверяем его на выполнение условий принадлежности к закономерностям. Обозначим через REG_1 множество всех закономерностей, обнаруженных на первом шаге.

– На шаге $k \leq d$ для каждой закономерности $R_i \in REG_{k-1}$, $R_i = Sit_{1,i}, A_{1,i} \to ... \to Sit_{k,i}, A_{k,i} \to G$, полученной на предыдущем шаге, выполняем:

- Генерируем множество $MREG_{k,i}(Sit_{1,i}) = Discovery(Sit_{1,i})$ всех закономерностей перехода в ситуацию $Sit_{1,i}$.
- для каждой закономерности $MR = Sit, A \to Sit_{1,i}$, $MR \in MREG_{k,i}(Sit_{1,i})$, создаем правило $R = Sit, A \to Sit_{1,i}, A_{1,i} \to ... \to Sit_{k,i}, A_{k,i} \to G$ путем уточнения закономерности R_i за счет перехода MR. Для правила R вычисляем его условную вероятность $p(R) = Estimate(R)$ и проверяем его на выполнение условий принадлежности к закономерностям. Обозначим через $REG_{k,i}$ множество всех закономерностей, полученных уточнением R_i переходами из $MREG_{k,i}(Sit_{1,i})$.

Обозначим через $REG_k = \bigcup\limits_i REG_{k,i}$ все закономерности, найденные на шаге k.

– Алгоритм останавливается либо на шаге $m = d$, либо когда не обнаружено новых закономерностей, $REG_m = \varnothing$.

– Результирующее множеством закономерностей является объединение всех множеств обнаруженных закономерностей $REG = \bigcup\limits_i REG_i$.

Таким образом, в результате выполнения алгоритма 2 мы одновременно получим и правила достижения цели вида (1), и модель среды, выраженную в виде набора правил вида (2).

5. Обнаружение подцелей

Общая идея обнаружения ситуаций-подцелей заключается в следующем: 1) сначала на основе исторических данных о действиях агента выделить основные цепочки переходов, приводящие его ранее к цели;

2) затем, перебирая различные комбинации признаков, описывающих ситуации, и оценивая степень изменения вероятности достижения цели при добавлении этих комбинаций в цепочки переходов, выделить те из них, которые значительно увеличивают вероятность, они и будут являться искомыми подцелями.

Перейдем к более детальному описанию.

Для выделения основных цепочек переходов, приводящих агента ранее к цели, можно использовать алгоритм 2, заменив в нем способ расчета оценки условной вероятности правил $Estimate(R)$. Замена способа расчета необходима, поскольку $Estimate(R)$ дает нам оценку вероятности достижения цели при условии использования правила. Сейчас же мы должны оценить, с какой вероятностью уже имеющиеся в прошлом цепочки действий приводили агента к цели. Для этого мы определим вероятность правил (1) классическим частотным методом:

$$p(R) = n(C, G)/n(C)$$

где $n(C)$ – количество раз в истории, когда агент совершал цепочку переходов и действий $C = Sit_1, A_1 \to Sit_2, A_2 \to ... \to Sit_m, A_m$, $n(C, G)$ – количество раз, когда он после совершения этих действий попадал в цель G.

В остальном алгоритм остается таким же, поэтому повторять его нет смысла.

Обозначим через $CH(G)$ множество правил-цепочек переходов, найденных данных алгоритмом.

Алгоритм 3. Входными параметрами алгоритма являются множество правил-цепочек переходов $CH(G)$; множество сенсорных предикатов $\mathbb{S}$, параметр глубины перебора d, где $1 \le d \le |\mathbb{S}|$ — натуральное число и δ, $0 < \delta < 1$ — порог увеличения вероятности.

- Инициализируем $NG = \varnothing$ — множество найденных подцелей.
- Перебираем все комбинации $\{S_1, ..., S_k\}$,$k \le d$,$S_i \in \mathbb{S}$. Для каждой комбинации $\{S_1, ..., S_k\}$ генерируем предикат-цель $G' = \{S_1, ..., S_k\}$ и предикат-результат достижения этой цели $Res(G')$, выполняем:
  - Для каждого правил-цепочек переходов $R_i \in CH(G)$, $R_i = Sit_1, A_1 \to Sit_2, A_2 \to ... \to Sit_m, A_m$ генерируем правило $R'_i = Sit_1 \cup Res(G'), A_1 \to Sit_2 \cup Res(G'), A_2 \to ... \to Sit_m \cup Res(G'), A_m$ путем добавления в каждое описание ситуации каждого перехода из правила R_i предиката-результата $Res(G')$.
  - Если $\arg\max_i (p(R'_i) - p(R_i)) > \delta$, то добавляем G' в множество подцелей $NG := NG \cup G'$.
- Возвращаем NG — искомое множеством подцелей.

Отметим, что алгоритм 3, помимо истинных подцелей, может вернуть и некоторое количество ложных «шумовых» подцелей, которые

случайным образом встречались в истории агента вместе с настоящими подцелями. Отфильтровать эти «шумовые» подцели можно, если проанализировать множество событий, когда была достигнута вышестоящая цель, при достижении подцелей. Для истинных подцелей это множество будет максимальным.

Таким образом, для каждой функциональной системы $F = \langle G, RES, REG\rangle$ при помощи описанного метода выявляются новые подцели. Для каждой обнаруженной подцели G' создается новая функциональная система $F' = \langle G', RES', REG'\rangle$, находящаяся ниже по иерархии системы F и реализующая достижение этой подцели. Для вновь созданной функциональной системы F' при помощи алгоритма генерации правил порождается множество правил REG'. Для этого просматривается все множество данных истории агента и выявляются случаи, когда новая подцель G' была достигнута в прошлом. У функциональной системы F множество предикатов-результатов RES обогащается еще одним предикатом $Res(G')$, а также генерируются новые правила. Тем самым, множество правил REG этой функциональной системы обогащаются правилами, содержащими новый результат $Res(G')$.

6. Результаты экспериментов

Продемонстрируем работу модели системы управления на примере решения трехэтапной задачи фуражирования.

Изначально система управления агентом состояла только из одной функциональной системы, задачей которой являлось достижение главной цели $G = (\text{центр} = \text{еда-3}) \wedge (\text{поглощение})$, т. е. чтобы в центральной клетке находился объект «еда-3», и было совершено поглощение.

После старта эксперимента агент совершал 2000 случайных шагов, накапливая статистику функционирования, и после этого запускал процедуру обнаружения подцелей и генерации правил. Продемонстрируем работу процедуры извлечения подцелей на примере обнаружения подцели второго уровня $G_2 = (\text{центр} = \text{еда-2}) \wedge (\text{поглощение})$.

Сначала алгоритм выделяет основные цепочки переходов, которые приводили агента к цели $G = (\text{центр} = \text{еда-3}) \wedge (\text{поглощение})$ в течение первых 2000 случайных шагов. Среди выявленных цепочек будет, к примеру, следующая:

$R = (\text{слева} = \text{еда-3}), (\text{налево}) \to (\text{впереди} = \text{еда-3}), (\text{шаг}) \to G.$

Вероятность этой цепочки будет небольшая, около 0.2.

Далее, согласно алгоритму 3, осуществляется перебор различных комбинаций сенсорных предикатов. В том числе в процессе перебора возникнет комбинация $G_2 = (\text{центр} = \text{еда-2}) \wedge (\text{поглощение})$. Для этой комбинации будет создан предикат-результат $Res(G_2)$. Этот предикат

будет добавляться в различные цепочки, в том числе, и в правило R, которое примет вид:
$(\text{слева} = \text{еда-3}), Res(G2), (\text{налево}) \to (\text{впереди} = \text{еда-3}), Res(G2), (\text{шаг}) \to G$.

Вероятность этой цепочки будет уже 1.0. Если мы примем порог $\delta = 0.5$, то прирост вероятности будет больше порога, значит, G_2 будет являться подцелью.

Аналогично будет обнаружена и подцель третьего уровня $G_3 = (\text{центр} = \text{еда-3}) \wedge (\text{поглощение})$.

Результаты серии тестовых запусков показали, что предложенная модель успешно решает трехэтапную задачу, стабильно обнаруживая две неявные подцели. Система также показала высокую скорость обучения: оптимальное поведение достигалось уже через 2000 тактов работы.

7. Заключение

В данной работе предложен логико-вероятностный подход к построению адаптивных обучаемых систем управления. Основное отличие данного подхода от других моделей, в том числе предложенных нами ранее моделей [9–12], заключается в следующем.

1. В данном подходе в процесс обучения агента интегрирован механизм обучения модели среды, что позволяет строить пути достижения цели по принципу обратного вывода. Это возможность значительно увеличивает скорость обучения агента, поскольку таким образом могут генерироваться пути, которые не встречались системе ранее в процессе обучения.

2. Предложенные ранее логико-вероятностные модели [3; 11–13] могли работать только с подцелями, явно присутствующими в сенсорном поле (к примеру, в виде отдельного сенсора, говорящего, достигнута подцель или нет). Это существенно ограничивало применимость алгоритма обнаружения подцелей, поскольку система не могла выявлять и работать с неявными подцелями. В новой модели эта проблема была решена за счет разделения понятия «цель» на, собственно, «цель» как ситуацию, которую надо достичь, и «результат» как факт достижения цели. В результате новая модель получила способность выявлять неявные подцели и работать с ними.

3. В новой модели предложен метод обнаружения подцелей, способный эффективно обнаруживать неявные подцели, результаты достижения которых достаточно сильно разнесены по времени с моментом достижения конечной цели. Данные подцели можно назвать «глубокими». Проведенные эксперименты показали, что модель успешно справляется с обнаружением таких «глубоких» подцелей.

Список литературы

1. Анохин П. К. Принципиальные вопросы общей теории функциональных систем // Принципы системной организации функций. М. : Наука, 1973. С. 5–61.
2. Витяев Е. Е. Извлечение знаний из данных. Компьютерное познание. Модели когнитивных процессов. Новосибирск : НГУ, 2006. 293 с.
3. Демин А. В., Витяев Е. Е. Логическая модель адаптивной системы управления // Нейроинформатика. 2008. Т. 3, № 1. С. 79–107.
4. Демин А.В., Витяев Е.Е. Реляционный подход к извлечению знаний и его применения // Материалы Всероссийской конференции с международным участием «Знания – Онтологии – Теории» (ЗОНТ-2013). Новосибирск, 2013. Т. 1. С. 122–130.
5. Al-Emran Mostafa. Hierarchical Reinforcement Learning: A Survey // IJCDS Journal. 2015. Vol. 4, N 2.P. 137–142. https://doi.org/10.12785/ijcds/040207
6. Dietterich T. G. Hierarchical reinforcement learning with the MAXQ value function decomposition // Journal of Artificial Intelligence Research/ 2000. Vol. 3. P. 227–303. https://doi.org/10.1613/jair.639
7. Goodfellow Ian, Bengio Yoshua, Courville Aaron. Deep learning. The MIT Press, 2016. 800 p.
8. Human-level control through deep reinforcement learning / V. Mnih [et al.] // Nature. 2015. Vol. 518. P. 529–533. https://doi.org/10.1038/nature14236
9. Mastering the game of Go with deep neural networks and tree search / D. Silver [et al.] // Nature. 2016. Vol. 529. P. 484–489. https://doi.org/10.1038/nature16961
10. Sutton R. S., Barto A. G. Reinforcement Learning. London : MIT Press, 2012. 320 p.
11. Vityaev E. E., Demin A. V., Kolonin Y. A. Logical probabilistic biologically inspired cognitive architecture // Artificial General Intelligence - 13th International Conference, AGI 2020, Proceedings. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Springer Gabler, 2020. Vol. 12177 LNAI. P. 337–346. https://doi.org/10.1007/978-3-030-52152-3_36
12. Vityaev E. E., Demin A. V. Cognitive architecture based on the functional systems theory // Procedia Computer Science. Elsevier, 2018. Vol. 145. P. 623–628. https://doi.org/10.1016/j.procs.2018.11.072
13. Vityaev E. E., Demin A. V. Recursive subgoals discovery based on the Functional Systems Theory // Biologically Inspired Cognitive Architectures 2011. IOS Press, 2011. P. 425–430.
14. Zhou Zhi-Hua, Feng Ji. Deep Forest: Towards An Alternative to Deep Neural Networks // Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI-17). 2017. P. 3553–3559. https://doi.org/10.24963/ijcai.2017/497
15. Zhou Zhi-Hua, Feng Ji. Deep Forest // National Science Review. 2019. Vol. 6, N 1. P. 74–86. https://doi.org/10.1093/nsr/nwy108

Александр Викторович Демин, кандидат физико-математических наук, Институт систем информатики им. А. П. Ершова СО РАН, Российская Федерация, 630090, г. Новосибирск, просп. Лаврентьева, 6, тел.: (383)3306660, alexandredemin@yandex.ru

Поступила в редакцию 27.10.2021

Deep Learning of Adaptive Control Systems Based on a Logical-probabilistic Approach

A. V. Demin[1]

[1] *Ershov Institute of Informatics Systems SB RAS, Novosibirsk, Russian Federation*

Abstract. The problem of automatic selection of subgoals is currently one of the most relevant in adaptive control problems, in particular, in Reinforcement Learning. This paper proposes a logical-probabilistic approach to the construction of adaptive learning control systems capable of detecting deep implicit subgoals. The approach uses the ideas of the neurophysiological Theory of functional systems to organize the control scheme, and logical-probabilistic methods of machine learning to train the rules of the system and identify subgoals. The efficiency of the proposed approach is demonstrated by an example of solving a three-stage foraging problem containing two nested implicit subgoals.

Keywords: control system, machine learning, knowledge discovery, reinforcement learning.

References

1. Anohin P.K. Fundamental questions of the general theory of functional systems. *The principles of the systemic organization of functions*. Moscow, Science Publ., 1973, pp. 5-61. (in Russian)
2. Vityaev E.E. *Extracting Knowledge from Data. Computer Cognition. Models of Cognitive Processes*. Novosibirsk, Novosibirsk State University Publ., 2006, 293 p. (in Russian)
3. Demin A.V., Vityaev E.E. A Logical Model of an Adaptive Control System. *Neiroinformatika*, 2008, vol. 3, no. 1, pp. 79-107. (in Russian)
4. Demin A.V., Vityaev E.E. Relational approach to knowledge discovery and its application. *Materials of the All-Russian conference with international participation "Knowledge - Ontology – Theories" (KONT-2013)*. Novosibirsk, 2013, vol. 3, pp. 122–130. (in Russian).
5. Al-Emran Mostafa. Hierarchical Reinforcement Learning: A Survey. *IJCDS Journal*, 2015, vol. 4, no. 2, pp. 137-142. https://doi.org/10.12785/ijcds/040207
6. Dietterich T.G. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research*, 2000, vol. 13, pp. 227–303. https://doi.org/10.1613/jair.639
7. Goodfellow Ian, Bengio Yoshua, and Courville Aaron. *Deep learning*. The MIT Press, 2016, 800 p.
8. Mnih V., Kavukcuoglu K., Silver D. et al. Human-level control through deep reinforcement learning. *Nature*, 2015, vol. 518, pp. 529-533. https://doi.org/10.1038/nature14236
9. Silver D., Huang A., Maddison C. et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 2016, vol. 529, pp. 484-489. https://doi.org/10.1038/nature16961
10. Sutton R.S., Barto A.G. *Reinforcement Learning*. London, MIT Press, 2012, 320 p.
11. Vityaev E.E., Demin A.V., Kolonin Y.A. Logical probabilistic biologically inspired cognitive architecture. *Artificial General Intelligence - 13th International*

Conference, AGI 2020, Proceedings. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Springer Gabler, 2020, vol. 12177 LNAI, pp. 337-346. https://doi.org/10.1007/978-3-030-52152-3_36

12. Vityaev E.E., Demin A.V. Cognitive architecture based on the functional systems theory. *Procedia Computer Science*, Elsevier, 2018, vol. 145, pp. 623-628. https://doi.org/10.1016/j.procs.2018.11.072
13. Vityaev E.E., Demin A.V. Recursive subgoals discovery based on the Functional Systems Theory. *Biologically Inspired Cognitive Architectures*, 2011, IOS Press, 2011, pp. 425-430.
14. Zhou Zhi-Hua and Feng Ji. Deep Forest: Towards An Alternative to Deep Neural Networks. *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI-17)*, 2017, pp. 3553-3559. https://doi.org/10.24963/ijcai.2017/497
15. Zhou Zhi-Hua and Feng Ji. Deep Forest. *National Science Review*, 2019, vol. 6, no.1, pp. 74-86. https://doi.org/10.1093/nsr/nwy108

Alexander Demin, Candidate of Sciences (Physics and Mathematics), Ershov Institute of Informatics Systems SB RAS, 6, Lavrentyev av., Novosibirsk, 630090, Russian Federation, tel.: +7 (383) 3306660,
e-mail: alexandredemin@yandex.ru

Received 27.10.2021