Моделирование и анализ данных
2024. Том 14. № 2. С. 7–22
DOI: https://doi.org/10.17759/mda.2024140201
ISSN: 2219-3758 (печатный)
ISSN: 2311-9454 (online)

Modelling and Data Analysis
2024. Vol. 14, no. 2, pp. 7–22
DOI: https://doi.org/10.17759/mda.2024140201
ISSN: 2219-3758 (print)
ISSN: 2311-9454 (online)

◇◇◇◇◇◇◇◇◇◇АНАЛИЗ ДАННЫХ◇◇◇◇◇◇◇◇◇◇

УДК 004.85

Опыт использования архитектуры сети Transformer для аппроксимации политики агента в обучении с подкреплением

Новиков Н.П.*
Московский авиационный институт
(национальный исследовательский университет) (МАИ)
г. Москва, Российская Федерация
e-mail: rtyderson@gmail.com

Виноградов В.И.**
Московский авиационный институт
(национальный исследовательский университет) (МАИ)
г. Москва, Российская Федерация
ORCID: https://orcid.org/0000-0003-3773-9653
e-mail: vvinogradov@inbox.ru

В данной статье рассмотрены основные положения алгоритма глубокого обучения с подкреплением и использование нейронных сетей для аппроксимации политики агента. Рассмотрено сравнение использования полносвязной нейронной сети и сети Transformer в алгоритме обучения с подкреплением.

Ключевые слова: искусственный интеллект, машинное обучение, глубокое обучение с подкреплением, марковские процессы принятия решений, Transformer, оптимизация.

Для цитаты:
Новиков Н.П., Виноградов В.И. Опыт использования архитектуры сети Transformer для аппроксимации политики агента в обучении с подкреплением // Моделирование и анализ данных. 2024. Том 14. № 2. С. 7–22. DOI: https://doi.org/10.17759/mda.2024140201

***Новиков Никита Павлович,** студент магистратуры, институт «Компьютерные науки и прикладная математика» Московский авиационный институт (национального исследовательского университета) (МАИ), г. Москва, Российская Федерация, e-mail: rtyderson@gmail.com

****Виноградов Владимир Иванович,** кандидат физико-математических наук, доцент, кафедра «Математическая кибернетика» Московский авиационный институт (национального исследовательского университета) (МАИ), г. Москва, Российская Федерация, ORCID: https://orcid.org/0000-0003-3773-9653, e-mail: vvinogradov@inbox.ru

1. ВВЕДЕНИЕ

В настоящее время алгоритмы глубокого обучения с подкреплением помогают решать задачи искусственного интеллекта методом проб и ошибок, не обладая априорными знаниями о решаемой задаче. Сам термин подкрепление (reinforcement) обозначает награду или наказание за результат, который может зависеть не только от принятых решений, но и некоторых неподконтрольных факторов. Вместо подачи обучающей выборки на вход алгоритму обучение происходит за счёт взаимодействия с некоторой средой (environment). Под желаемым результатом подразумевается максимизация некоторой скалярной величины, называемой наградой (reward). Саму сущность (систему), которая принимает решения и взаимодействует со средой, принято называть агентом (agent). При взаимодействии агента с миром или средой агенту доступно только некоторое наблюдение (observation) текущего состояния данной среды. Агент взаимодействует со средой посредством действий (actions), которые выбираются благодаря некоторой процедуре. Данную процедуру принято называть стратегией или политикой (policy). Сам процесс взаимодействия агента и среды называется динамикой среды (world dynamics), которая определяет условия, по которым меняются состояния среды во времени и генерируется некоторая награда.

Основываясь на том, что агент взаимодействует со средой, а соответственно, выполняет какие-либо действия благодаря своей стратегии, которая моделируется некоторым распределением, в широкое применение вошло использование глубоких нейронных сетей для аппроксимации данной функции распределения. В связи с прогрессом в области искусственного интеллекта появилось множество новых архитектур нейронных сетей, а именно архитектура сети Transformer, именно она будет использована в данной работе. Благодаря ей стало возможно обрабатывать последовательности векторов и вектор контекста самой последовательности, не теряя при этом входную информацию.

2. СВЯЗЬ С ОПТИМАЛЬНЫМ УПРАВЛЕНИЕМ И МАРКОВСКИМ ПРОЦЕССОМ ПРИНЯТИЯ РЕШЕНИЙ

Буквами *s, a, r* принято обозначать состояния, действия и награды, буква *t* используется для идентификации времени. Для объяснения задачи в терминах оптимального управления нужно свести её в рамки лапласовского детерминизма и определить модель мира [1]. Это означает, что положение, скорости и ускорения всех атомов вселенной задают её исходное состояние, а изменение этого состояния происходит согласно некоторым дифференциальным уравнениям. Исходя из вышеупомянутых высказываний, систему можно описать:

$$\dot{s} = f\left(s,t\right),$$

где f – функция перехода состояний среды, s – текущее состояние среды, $\dot{s}$ – скорость изменения состояния среды.

Поскольку агент обладает некоторой стратегией принятия решений, а соответственно может влиять на среду, то основываясь на уравнении выше можно описать взаимодействие агента со средой следующим образом:

$$\dot{s} = f\left(s, a\left(s,t\right),t\right).$$

Начальное условие: $s_0 = s\left(t_0\right) \in S$ – произвольное состояние в множестве возможных состояний агента, $a\left(s,t\right)$ – это действие, выбранное агентом в определённый момент времени *t,* кроме того, выбранное действие зависит от состояния, в котором находится агент в момент времени *t*. Награда агента также моделируется в каждый момент времени и составляет $r\left(s,a\left(s,t\right),t\right)$. Итоговая награда складывается из сумм наград за каждый временной шаг, именно это значение максимизируется агентом и является целевой функцией:

$$R = \sum_{t=0}^{T} \gamma^t r_t \to \max_{\pi},$$

где γ – коэффициент дисконтирования, r_t – награда агента на шаге t, π – политика (стратегия) агента, T – максимальное количество шагов в среде.

Коэффициент дисконтирования $\gamma \in [0,1]$ – важная переменная, влияющая на то, как оцениваются будущие вознаграждения. Чем больше значение – тем больший вес имеют будущие вознаграждения, полученные агентом.

Описываемую задачу можно представить через определение марковской цепи, где переход из состояния в состояние зависит только от текущего момента времени и состояния агента, но не зависит от всей предыдущей истории взаимодействия агента со средой. Таким образом, предложенное свойство Марковости позволяет свести задачу к MDP – Марковскому процессу принятия решений (Markov Decision Process, MDP). Данное определение не проявляется в усложнении задачи, поскольку его легко представить в виде:

$$\left(S, A, P, r\right),$$

где *S, A, P* – это описание среды, а именно:

S – *пространство состояний*, множество состояний, в которых в каждый момент времени может находится среда;

A – *пространство действий,* множество вариантов действий, из которых агент производит свой выбор на каждом шаге;

P – *функция перехода,* которая задаёт изменение среды после того, как в состоянии *s* было выбрано действие *a;*

r – $S \times A \to \mathbb{R}$: *функция награды*, которая выдаёт скалярную величину за выбор действия *a* в состоянии *s*. Именно она выступает в роли «обучающего сигнала».

3. ОПИСАНИЕ ПОСТАВЛЕННОЙ ЗАДАЧИ И МОДЕЛИ МИРА

Целью данной работы является проверка применимости архитектуры сети Transformer для аппроксимации политики агента и сравнение с полносвязной архитектурой сети в задаче обучения с подкреплением посредством обработки последовательности данных. Идея использования архитектуры сети Transformer основывается на том, что множество алгоритмов обучения с подкреплением страдает от увеличения пространства действий, что прямым образом сказывается на сходимости алгоритма решения. Поэтому использование новой архитектуры сети с возможностью обработки последовательностей позволит уменьшить пространство действий, что приведёт к ускорению сходимости и увеличению суммарной награды.

Задачу можно сформулировать следующим образом. Агенту доступны: вид пространства состояний, вид пространства действий и взаимодействие со средой. Благодаря последнему, агент собирает данные путём выбора некоторого действия, перемещающего его из одного состояния в другое, где совершаемое агентом действие принадлежит некоторой стратегии (политике), которую аппроксимирует нейронная сеть. Требуется определить стратегию агента, максимизирующую суммарную награду на основе его опыта взаимодействия со средой.

Для реализации решения поставленной задачи используется симулятор Mujoco – бесплатный физический движок с открытым исходным кодом, принадлежащий DeepMind. Существует одиннадцать сред Mujoco: Ant, Hopper, Humanoid и др. Для реализации и тестирования поставленной задачи использовалась среда Ant, которая основана на модели мира, представленной Шульманом, Морицем, Левином, Джорданом и Аббилем в работе [6]. Ant – это 3D робот, который состоит из туловища с прикреплёнными к нему четырьмя ногами, где каждая из которых состоит из двух секций. Суть задачи сводится к удержанию роботом равновесия и движения вышеупомянутого вперёд, путём координации его действий с помощью приложения крутящих моментов к каждому шарниру робота. Шарниры находятся на стыках креплений секций ног и соединений самих ног с туловищем.

Таким образом, пространство действий можно представить в виде вектора размерности 8 (Таблица 1), где каждое значение представляет собой крутящий момент, приложенный к шарнирному соединению.

Таблица 1

Пространство действий агента (робота Ant)

Индекс	Действие	Минимальное значение	Максимальное значение
0	Крутящий момент, приложенный к шарниру между туловищем и задним правым бедром	–1	1
1	Крутящий момент, приложенный к шарниру между двумя секциями задней правой ноги	–1	1

Индекс	Действие	Минимальное значение	Максимальное значение
2	Крутящий момент, приложенный к шарниру между туловищем и передним левым бедром	–1	1
3	Крутящий момент, приложенный к шарниру между двумя секциями передней левой ноги	–1	1
4	Крутящий момент, приложенный к шарниру между туловищем и передним правым бедром	–1	1
5	Крутящий момент, приложенный к шарниру между двумя секциями передней правой ноги	–1	1
6	Крутящий момент, приложенный к шарниру между туловищем и задним левым бедром	–1	1
7	Крутящий момент, приложенный к шарниру между двумя секциями задней левой ноги	–1	1

Пространство наблюдений для робота описывается значениями положения различных частей его тела и скоростями этих отдельных частей. Следовательно, для описания состояния робота используется вектор размерности 27, который описан в таблице 2.

Таблица 2

Пространство состояний агента (робота Ant)

Индекс	Наблюдение	Значение
0	Координата туловища по оси z	Позиция (м)
1	x – ориентация туловища	Угол наклона (рад)
2	y – ориентация туловища	Угол наклона (рад)
3	z – ориентация туловища	Угол наклона (рад)
4	w – ориентация туловища	Угол наклона (рад)
5,7, 9, 11	Углы между туловищем и секциями каждой ноги	Угол наклона (рад)
6, 8, 10, 12	Углы между двумя секциями каждой ноги	Угол наклона (рад)
13	Скорость движения туловища по x – координате	Скорость (м/с)
14	Скорость движения туловища по y – координате	Скорость (м/с)
15	Скорость движения туловища по z – координате	Скорость (м/с)
16	x – координата угловой скорости движения туловища	Угловая скорость (рад/с)
17	y – координата угловой скорости движения туловища	Угловая скорость (рад/с)
18	z – координата угловой скорости движения туловища	Угловая скорость (рад/с)
19, 21, 23, 25	Угловая скорость угла между туловищем и секциями каждой ноги	Угол наклона (рад)
20, 22, 24, 26	Угловая скорость угла между двумя секциями каждой ноги	Угол наклона (рад)

Вознаграждение для агента состоит из трёх частей:

1) награда за устойчивость (если агент оставался на ногах до конца эпизода);
2) награда за движение вперёд, которая измеряется как скорость продвижения робота вперёд;

3) отрицательная награда за величину действия (нормировка по действиям).

Кроме того, в начальном состоянии к позиционным значениям состояния робота добавляется некоторый равномерный шум $U(-0.01,0.01)$, а к значениям скоростей добавляется нормальный гауссовский шум $\mathcal{N}(0,0.01)$.

Симуляция, или как её называют иначе – эпизод, заканчивается при достижении агентом (роботом) заданной точки, при перевороте робота или при окончании длины симуляции (1000 временных интервалов).

4. ОПИСАНИЕ АЛГОРИТМА РЕШЕНИЯ ЗАДАЧИ

В алгоритмах обучения с подкреплением сбор данных является важной частью. Определяя политику взаимодействия со средой, можно повлиять на то, для каких состояний s,a мы получаем сэмпл s' из функции переходов. Собираемые данные – траектории (роллауты), используются для обучения алгоритма.

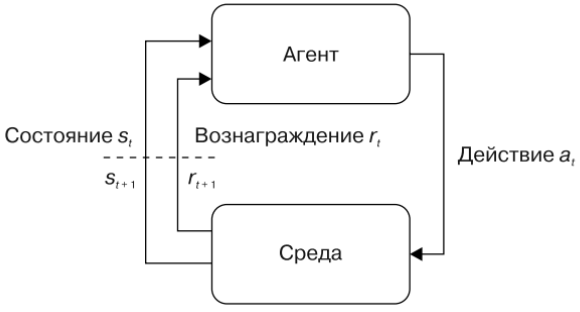

Рис. 1. Схема взаимодействия агента и среды

Для решения задачи использовался алгоритм, основывающийся на Policy Gradient подходе. Такой класс алгоритмов использует оценки градиента функционала по параметрам стратегии (политики). Алгоритм Proximal Policy Optimization (PPO) [2] считается устоявшимся решением в обучении с подкреплением, именно этот алгоритм использовался при создании алгоритма, победившего профессионалов в киберспортивной игре Dota 2.

Основываясь на [2], итоговую процедуру работы алгоритма можно представить следующим образом. Собирается большой набор переходов (s,a,r,s') по политике, где каждый переход – это один временной шаг агента в среде. Текущая версия стратегии (политики) агента π_θ и вероятности выбранных действий сохраняются. Для всех пар состояние-действие (s,a) рассчитывается оценка Advantage функции (функции полезности) методом GAE [2].

Из полученного набора данных (датасета) берутся батчи пар $s \sim d_{\pi^{old}}(s)$, $a \sim \pi^{old}(a|s)$, где $d_{\pi^{old}}(s)$ – распределение состояний, полученное по политике агента π^{old}, и считается Монте-Карло оценка градиента целевой функции:

$\mathrm{E}_{s\sim d_{\pi^{old}}(s)}\mathrm{E}_{\mathrm{a}\sim\pi^{\mathrm{old}}(a|s)}\left[\min\left(\rho(\theta)A^{\pi^{old}}(s,a),\rho^{clip}(\theta)A^{\pi^{old}}(s,a)\right)-C\sqrt{KL(\pi^{old}||\pi_\theta)}\right]\rightarrow\max_\theta$,

где C – константа, KL – KL-дивергенция, $\rho(\theta)=\frac{\pi_\theta(s|a)}{\pi^{old}(a|s)}$, $\rho^{clip}(\theta)=clip(\rho(\theta),1-\epsilon,1+\epsilon),\epsilon\in(0,1)$, $clip$ – усечение в пределах [1 – ϵ, 1 + ϵ].

Алгоритм Proximal Policy Optimization

Гиперпараметры:

M – количество параллельных сред,
N – длина роллаутов (траекторий),
B – размер мини-батчей (пакетов данных),
N_epoch – количество эпох (итераций) обучения,
γ – коэффициент дисконтирования,
λ – параметр GAE оценки,
ϵ – параметр обрезки для актёра (сеть, моделирующая стратегию агента),
$\hat{\epsilon}$ – параметр обрезки для критика (сеть, «оценивающая» принятые решения),
$V_\phi(s)$ – нейронная сеть (произвольной архитектуры) критика с параметрами ϕ,
$\pi_\theta(a\,|s)$ – нейронная сеть (произвольной архитектуры) для стратегии агента (актера) с параметрами θ,
α – коэффициент масштабирования ошибки критика,

Оптимизатор – Stohastic Gradient Descent (используется для оптимизации нейронных сетей критика и актёра).

Псевдокод алгоритма:

Инициализировать θ,ϕ.

На каждом шаге:

В каждой параллельной среде собрать роллаут $(s_0,a_0,r_0\ldots,s_N,a_N,r_N)$ длины N, используя стратегию π_θ, сохраняя вероятности выбора действий как $\pi^{old}(a|s)$, а выход критика на встреченных состояниях как $V^{old}(s)\bullet V_\phi(s)$.

Для каждой пары *s, a* из роллаутов посчитать одношаговую оценку Advantage:

$$\psi_1(s_t,a_t)=r_t+\gamma\left(1-done_{t+1}\right)V_\phi(s_{t+1})-V_\phi(s_t),$$

где *done* – показатель конца эпизода.

Посчитать Advantage оценку:

$$\psi_{GAE}(s_{N-1},a_{N-1})=\psi_1(s_{N-1},a_{N-1})$$

1) Для *t* от (*N - 2)* до 0:

$$\psi_{GAE}(s_t,a_t)=\psi_1(s_t,a_t)+\gamma\lambda\left(1-done_t\right)\psi_{GAE}(s_{t+1},a_{t+1})$$

2) Посчитать целевое значение (таргет) для критика:

$$y(s)=\psi_{GAE}(s,a)+V_\phi(s),$$

где s – из роллаута, собранного в п. 1.

3) Составить датасет из шестёрок:

$$\left(s,a,\psi_{GAE}\left(s,a\right),y\left(s\right),\pi^{old}\left(a|s\right),\ V^{\text{old}}\left(s\right)\right),$$

где $V^{\text{old}}\left(s\right)$ – старая политика критика.

Выполнить N_epochs проходов по роллауту, генерируя мини-батчи шестёрок $\mathbb{T}=\left(s,a,\psi_{GAE}\left(s,a\right),y\left(s\right),\pi^{old}\left(a|s\right),\ \text{V}^{\text{old}}\left(s\right)\right)$, размером *B*.

Для каждого мини-батча:

- Вычислить ошибку критика:

$$Loss_1\left(\mathbb{T},\phi\right)=\left(y\left(s\right)-V_\phi\left(s\right)\right)^2,$$

$$Loss_2\left(\mathbb{T},\phi\right)=\left(y\left(s\right)-clip\left(V_\phi\left(s\right)-V^{old}\left(s\right),-\hat{\epsilon},\hat{\epsilon}\right)\right)^2,$$

$$Loss^{critic}\left(\phi\right)=\frac{1}{B}\sum_{\mathbb{T}}max\left(Loss_1\left(\mathbb{T},\phi\right),Loss_2\left(\mathbb{T},\phi\right)\right).$$

Сделать шаг градиентного спуска по ϕ, используя $\nabla_\phi Loss^{critic}\left(\phi\right)$.

Нормализовать $\psi_{GAE}\left(s,a\right)$ по батчу, чтобы в среднем значения равнялись 0, а дисперсия – 1.

Посчитать коэффициенты для выборки по важности (importance sampling):

$$r_\theta\left(\mathbb{T}\right)=\frac{\pi_\theta(a|s)}{\pi^{old}(a|s)}$$

- Посчитать обрезанную версию градиента:

$$r_\theta^{clip}\left(\mathbb{T}\right)=clip\left(r_\theta\left(\mathbb{T}\right),1-\epsilon,1+\epsilon\right)$$

- Вычислить градиент для нейронной сети актёра:

$$\nabla_\theta^{actor}=\frac{1}{B}\sum_{\mathbb{T}}\nabla_\theta\min\left(r_\theta\left(\mathbb{T}\right)\psi_{GAE}\left(s,a\right),r_\theta^{clip}\left(\mathbb{T}\right)\psi_{GAE}\left(s,a\right)\right)$$

Сделать шаг градиентного подъёма по θ, используя ∇_θ^{actor}.

Работы, рассматривающие решение поставленной задачи, как правило, использовали данный алгоритм, в котором в роли аппроксиматора политики агента выступала полносвязная глубокая нейронная сеть. Архитектура такой сети представляет собой перцептрон, состоящий из двух скрытых линейных слоёв с гиперболическим тангенсом (tanh) в качестве функции активации и количеством нейронов 64, 64 и 8 для актёра и 64, 64 и 8 для критика соответственно. Пример архитектуры такой сети представлен на рисунке 2.

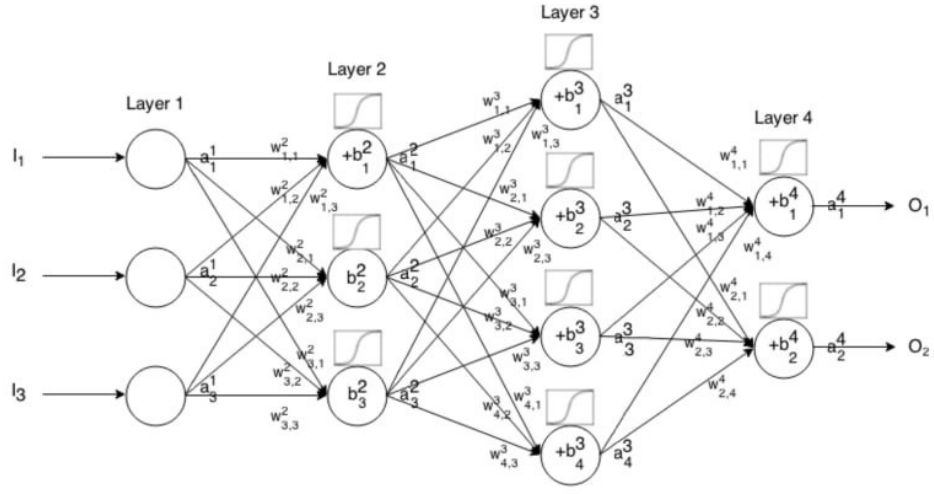

Рис. 2. Схема архитектуры полносвязной нейронной сети

Выходная размерность сети соответствует пространству действий агента, следовательно, при увеличении размерности задачи возникали проблемы с увеличением количества выходных нейронов сети актёра.

В данной работе для борьбы с «проклятием размерности» используется подход, который подразумевает замену представления пространства действий и, соответственно, замену архитектуры сети. Ключевой идеей является использование сети, состоящей из энкодера и декодера. Благодаря такому подходу, становится возможным использование последовательностей входных данных и сохранение внутренней информации самой последовательности без потери качества.

Основой для разработки сети в текущей работе послужила статья [3] об архитектуре сети Transformer, которая полностью основывается на механизме внимания и не использует рекуррентные слои или слои свертки. Ниже, на рисунке 3, представлено устройство сети Transformer.

Основная идея заключается в том, что сеть разбивается на два блока – энкодер и декодер, слева и справа на рисунке 3 соответственно.

Энкодер последовательно применяет к исходной последовательности N блоков, в каждом из которых входные данные проходят через два основных слоя: multi-head attention и feed-forward. Multi-head attention состоит из нескольких наборов self-attention, которые можно по-другому назвать слоями самовнимания, они отличаются от обычного слоя внимания тем, что выходом слоя являются новые представления всё той же последовательности, каждый элемент которой напрямую взаимодействует друг с другом. Сама реализация данного слоя изложена в [3], стоит отметить лишь то, что благодаря распараллеливанию слоёв внимания удается уловить намного больше внутренних зависимостей последовательности, что и реализует слой multi-head attention.

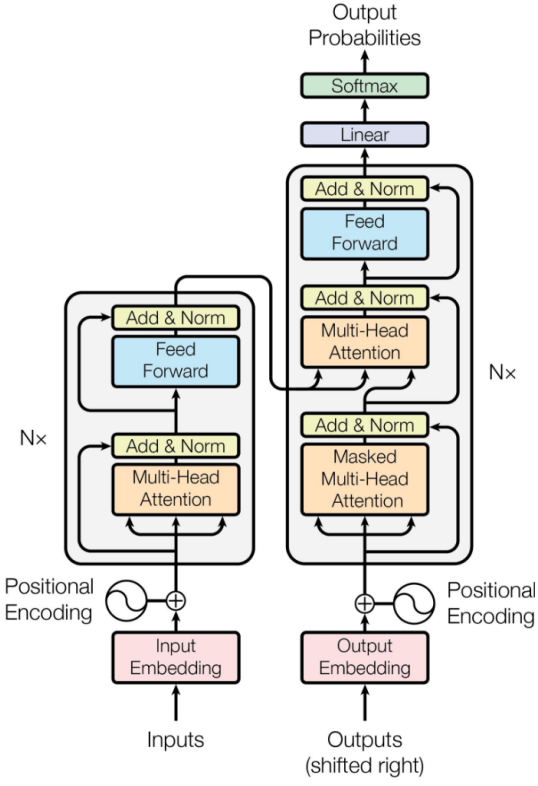

Рис. 3. Схема архитектуры Tranformer

В декодере используется похожая структура, но основное отличие заключается в слое cross-attention (кросс-внимание). В данном слое для формирования результата многоголового внимания используются результаты обработки энкодером входной последовательности. Кроме того, на вход энкодеру подается вся последовательность сразу, в то время как декодер обрабатывает последовательность итерационно, т.е. на каждом шаге он получает информацию о текущем элементе последовательности (токене) и обо всех предыдущих, не заглядывая вперёд.

Для использования описанной ранее архитектуры сети Transformer необходимо преобразовать входные данные в последовательность. Для этого необходимо объединить элементы входного вектора состояний в некоторые представления, основываясь на их информационной ценности. Таким образом, был получен новый входной вектор для нейронной сети $s = \{s_0, s_1, s_2, s_3, s_4\}$. Каждый элемент в данной последовательности отвечал за отдельную часть тела робота:

Элементы, описывающие торс робота:

$$s_0 = \{0, 1, 2, 3, 4, 13, 14, 15, 16, 17, 18\}$$

Элементы, описывающие переднюю левую ногу робота:

$$s_1 = \{5, 6, 19, 20\}$$

Элементы, описывающие переднюю правую ногу робота:

$$s_2 = \{7,8,21,22\}$$

Элементы, описывающие заднюю левую ногу робота:

$$s_3 = \{9, 10, 23, 24\}$$

Элементы, описывающие заднюю правую ногу робота:

$$s_4 = \{11, 12, 25, 26\}$$

Данные векторы $s_0 \ldots s_4$ содержат индексы элементов исходного вектора состояния. Описание каждого элемента состояния и их величин доступно в таблице 2 и в документации среды [4].

После представления вектора состояния в виде последовательности формируются входные эмбеддинги (векторные представления данных) и передаются в энкодер сети Transformer. В отличие от [3] позиционное кодирование не используется, поскольку информация о позициях ног и туловища в последовательности входного вектора не оказывает влияния на дальнейшее обучение, критична лишь их исходная инициализация.

Для использования декодера необходимо определить размерность выходного вектора. Так как агент способен управлять лишь шарнирными соединениями своих ног, то выходной вектор последовательности сокращается до 4 элементов, соответствующих каждой ноге робота. При этом каждый элемент содержит 2 величины крутящего момента для шарниров ноги. Благодаря такому подходу удалось сократить пространство действий до размерности 2, что в четыре раза меньше исходной размерности вектора действий из таблицы 1.

5. РЕЗУЛЬТАТЫ ЭКСПЕРИМЕНТОВ

Обучение агента проводилось на GPU, чтобы ускорить распараллеленные процессы в нейронной сети. Было запущено два эксперимента с одинаковым набором гиперпараметров для алгоритма PPO (см. табл. 3), реализованном в библиотеке StableBaselines на языке Python [5].

Таблица 3

Гиперпараметры алгоритма обучения с подкреплением PPO

Learning rate	0,0003
Количество шагов в эпизоде	1000
Размер мини-батча	64
Коэффициент дисконтирования γ	0,99

Коэффициент для оценки GAE λ	0,95
Размер clip PPO	0,2
Максимальное значение для обрезки градиента	0,5
Размер окна для усреднения оценки политики агента	100

В первом эксперименте использовалась полносвязная нейронная сеть с параметрами, описанными в таблице 4.

Таблица 4

Параметры полносвязной нейронной сети для алгоритма PPO

Количество скрытых слоёв	2
Количество нейронов в скрытых слоях	64
Функция активации	Tanh

Во втором эксперименте использовалась сеть архитектуры Transformer, параметры которой описаны в таблице 5.

Таблица 5

Параметры нейронной сети архитектуры Transformer для алгоритма PPO

Количество голов multi-head attention энкодера	4
Функция активации в полносвязном слое энкодера	ReLu
Количество голов в masked multi-head attention декодера	4
Функция активации в полносвязном слое декодера	Relu
Количество нейронов в слоях энкодера	64
Количество нейронов в слоях декодера	64
Размерность слоя эмбеддинга для энкодера	32
Размерность слоя эмбеддинга для декодера	32

После обучения агента на более чем 1 миллионе шагов в среде были получены следующие результаты. Для агента, обученного с помощью полносвязной сети длина эпизода достигала максимум 910 шагов в среде и среднее значение награды за эпизод 1212, синяя кривая на рисунках 3 и 4. Награда формируется из суммы нескольких значений:

1) награда за пребывание в «здоровом» состоянии, если муравей не перевернулся = 1;
2) награда за продвижение вперёд = 0.05;
3) штраф за слишком большое (по модулю) действие = 0.5 sum (действие агента^2).

Результаты для агента, который был обучен с помощью сети Transformer, были выше, чем у предыдущего агента. Длина эпизода для него сошлась к значению 1000 уже за 475000 шагов в среде, в то время как роботу, политика которого основывалась на полносвязной сети, не удалось достигнуть такого результата за всё время обучения. Также, награда у нового агента достигала отметки в 1380 единиц и продолжала расти. Графики для данного агента описываются красной кривой на рисунках 4 и 5.

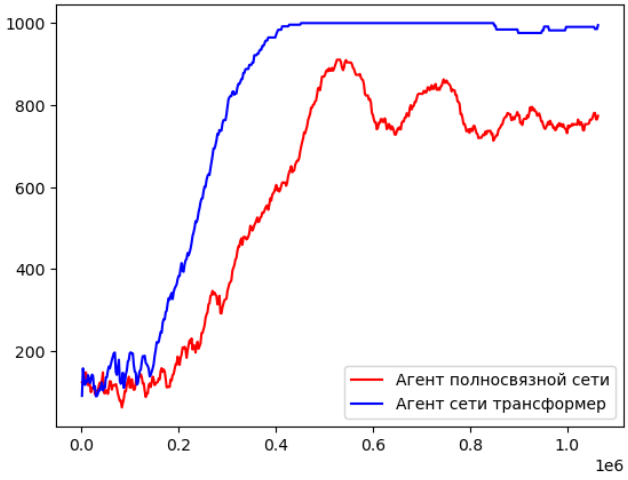

Рис. 4. График зависимости средней длины эпизода агентов от количества шагов обучения

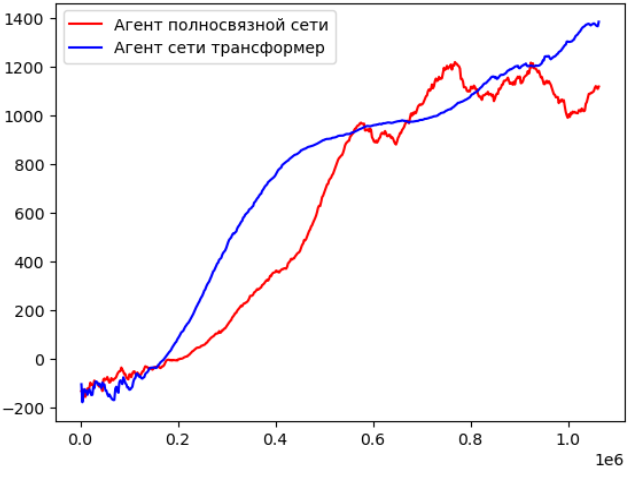

Рис. 5. График зависимости средней награды агентов от количества шагов обучения

6. ЗАКЛЮЧЕНИЕ

Апробация новой архитектуры сети Transformer для аппроксимации стратегии обучения агента позволила добиться положительных результатов. Благодаря использованию архитектуры сети Transformer при обучении робота передвижению, удалось

сузить пространство действий, увеличить продолжительность эпизода и получить награду, превосходящую результат, получаемый для решения аналогичной задачи с использованием глубокой полносвязной нейронной сети. Исходя из всего вышесказанного, можно сделать вывод, что применение архитектуры нейронной сети Transformer для использования в решении задач обучения с подкреплением может привести к улучшению качества результатов обучения и снижению вычислительных нагрузок.

Литература

1. Конспект по обучению с подкреплением // Arxiv URL: https://arxiv.org/pdf/2201.09746.pdf (дата обращения: 02.01.2024).
2. Proximal Policy Optimization Algorithms // Arxiv URL: https://arxiv.org/pdf/1707.06347.pdf (дата обращения: 24.12.2023).
3. Attention Is All You Need // Arxiv URL: https://arxiv.org/abs/1706.03762 (дата обращения: 16.12.2023).
4. Gymnasium URL: https://gymnasium.farama.org/ (дата обращения: 10.12.2023).
5. Stable Baselines Documentation // URL: https://buildmedia.readthedocs.org/media/pdf/stable-baselines/master/stable-baselines.pdf (дата обращения: 08.12.2023).
6. High-dimensional continuous control using generalized advantage estimation // Arxiv URL: https://arxiv.org/pdf/1506.02438.pdf (дата обращения: 07.12.2023).

Experience in Using the Transformer Network Architecture to Approximate Agent's Policy in Reinforcement Learning

Novikov N.P.*
Moscow Aviation Institute (National research university) (MAI), Moscow, Russian
e-mail: rtyderson@gmail.com

Vinogradov V.I.**
Moscow Aviation Institute (National research university) (MAI), Moscow, Russian
ORCID: https://orcid.org/0000-0003-3773-9653
e-mail: vvinogradov@inbox.ru

This paper discusses the basics of the deep reinforcement learning algorithm and the use of neural networks to approximate the agent's policy. The comparison of using a fully connected neural network and a transformer network in the reinforcement learning algorithm is considered.

Keywords: artificial intelligence, machine learning, deep reinforcement learning, Markov decision processes, transformer, optimization.

For citation:
Novikov N.P., Vinogradov V.I. Experience in Using the Transformer Network Architecture to Approximate Agent's Policy in Reinforcement Learning. *Modelirovanie i analiz dannykh = Modelling and Data Analysis,* 2024. Vol. 14, no. 2, pp. 7–22. DOI: https://doi.org/10.17759/mda.2024140201 (In Russ., abstr. in Engl.).

References

1. An outline of reinforcement learning // Arxiv URL: https://arxiv.org/pdf/2201.09746.pdf (circulation date: 02.01.2024).
2. Proximal Policy Optimization Algorithms // Arxiv URL: https://arxiv.org/pdf/1707.06347.pdf (circulation date: 24.12.2023).
3. Attention Is All You Need // Arxiv URL: https://arxiv.org/abs/1706.03762 (circulation date: 16.12.2023).
4. Gymnasium URL: https://gymnasium.farama.org/ (circulation date: 10.12.2023).

***Novikov Nikita Pavlovich,** master's student, Institute of Computer Science and Applied Mathematics, Moscow Aviation Institute (National Research University) (MAI), Moscow, Russia, e-mail: rtyderson@gmail.com

****Vinogradov Vladimir Ivanovich,** PhD, Associate Professor, Department of Mathematical Cybernetics, Moscow Aviation Institute (National Research University) (MAI), Moscow, Russia, ORCID: https://orcid.org/0000-0003-3773-9653, e-mail: vvinogradov@inbox.ru

5. Stable Baselines Documentation // URL: https://buildmedia.readthedocs.org/media/pdf/stable-baselines/master/stable-baselines.pdf (circulation date: 08.12.2023).
6. High-dimensional continuous control using generalized advantage estimation // Arxiv URL: https://arxiv.org/pdf/1506.02438.pdf (circulation date: 07.12.2023).

Получена 06.03.2024
Принята в печать 21.03.2024

Received 06.03.2024
Accepted 21.03.2024