СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Обучение с подкреплением – это один из способов машинного обучения. Отличительной особенностью этого метода является то, что для обучения не требуются наборы тренировочных данных. Обучение происходит в средах, которые можно описать марковским процессом принятия решений (МППР), схема которого изображена на рисунке 1. Либо в средах, которые можно описать частично наблюдаемым марковским процессом принятия решений (ЧНМППР), схема которого изображена на рисунке 2. Разница между данными двумя видами сред заключается в том, что в МППР с помощью предоставляемых средой наблюдений можно однозначно определить ее состояние. В случае с ЧНМППР, наблюдения несут контекстную информацию, то есть одному наблюдению может соответствовать множество состояний среды. Такие среды полезны в случаях, когда в виде МППР задачу нельзя обобщить, либо когда невозможно однозначно описать среду.

При этом важно понимать, что успешность обучения зависит не только от выбранного алгоритма и гиперпараметров, но и от того, как спроектирована среда обучения. А именно: какие данные выбраны наблюдаемыми, как формируется награда и как масштабируется награда.

Целью данной учебной практики является проектирование двух сред, описываемых МППР и ЧНМППР. В дальнейшем первая и вторая среда соответсовенно. А также обучение модели в созданных средах.

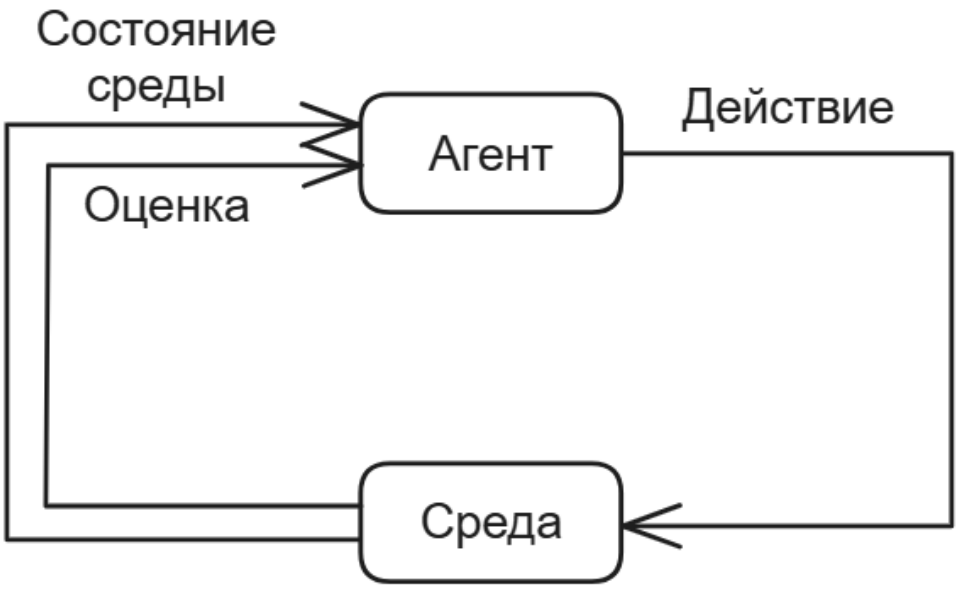

Рисунок 1 – Марковский процесс принятия решений

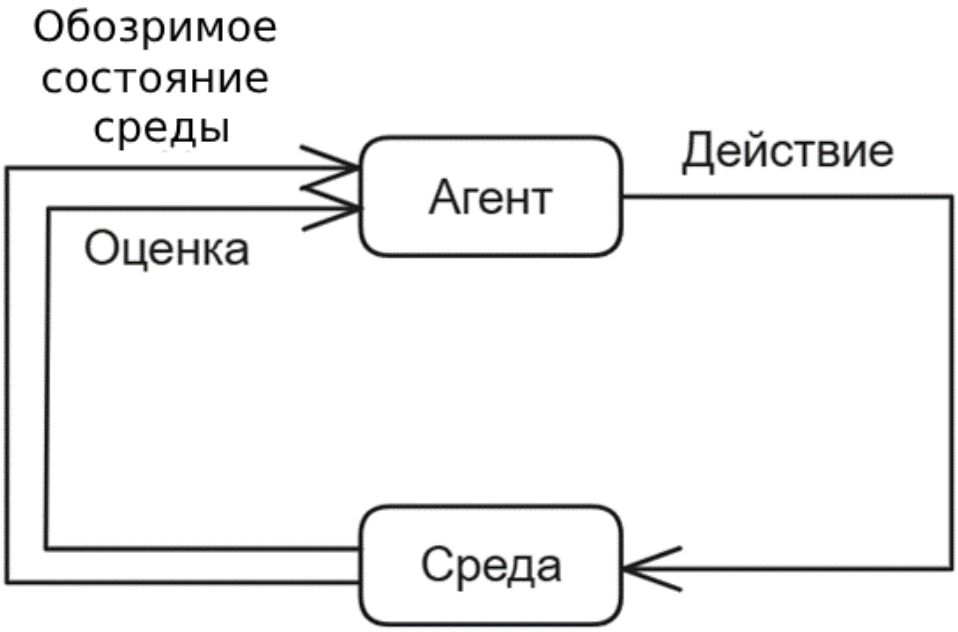

Рисунок 2 – Частично наблюдаемый марковский процесс принятия решений

1. ПРОЕКТИРОВАНИЕ СРЕДЫ ОБУЧЕНИЯ

1.1. ОПИСАНИЕ ПЕРВОЙ СРЕДЫ

В качестве инструмента для создания среды была выбрана платформа для разработки игр Unity с расширением ML-Agents[4]. Для процесса обучения модели была выбрана библиотека PyTorch. В качестве связующего звена между средой и моделью используется библиотека Gymnasium.

Среда обучения представляет собой бесконечное поле, в центре которого находится цель. Задача агента – добраться до цели за наименьшее время. При этом помимо агента на поле присутствуют враги, которые движутся в направлении агента с постоянной скоростью. Эпизод обучения завершается если агент сталкивается с врагом, добирается до цели или превышает максимальное время эпизода.

Точка появления агента в начале эпизода, как и точки появления врагов, случайна в некотором радиусе от цели. При этом враги не могут появиться слишком близко к агенту.

Визуально среда представлена на рисунке 3. Где синим обозначен агент, красным – враги, а зеленым – цель.

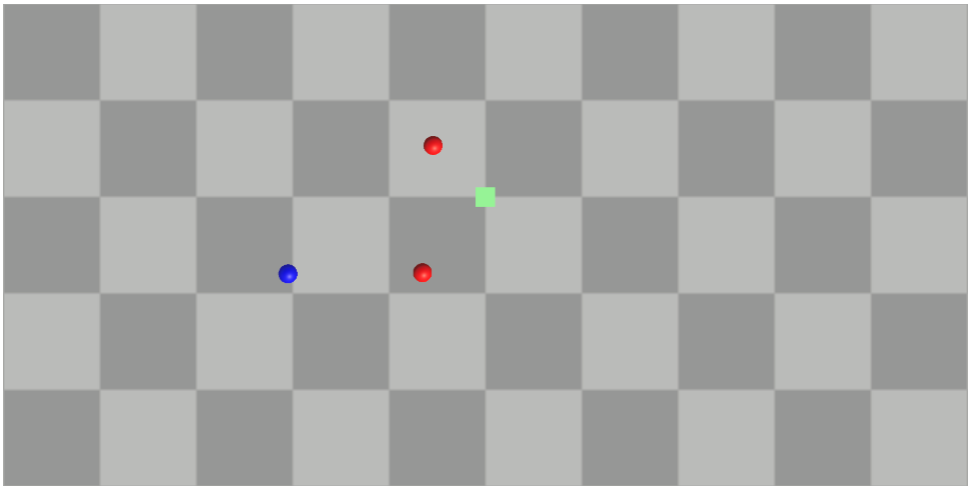

Рисунок 3 – Визуальное представление среды

1.2. НАБЛЮДЕНИЯ В ПЕРВОЙ СРЕДЕ

Состояние среды описывается следующими параметрами:

- абсолютные координаты агента,
- абсолютные координаты врагов,
- абсолютные координаты цели.

В качестве предоставляемых средой наблюдений можно взять состояние таким, какое оно есть. Однако это повлечет за собой проблему. Использование абсолютных координат значительно понижает обобщенность модели. То есть при сдвиге положения участка, на котором происходило обучение, модель потребует переобучения. В такой ситуации целесообразно брать не абсолютные координаты, а относительные. В нашем случае наиболее обобщенная модель получится если брать координаты относительно агента.

В итоге предоставляемые наблюдения выглядят следующим образом:

- относительные координаты врагов,
- относительные координаты цели.

Так как наблюдение однозначно описывает состояние среды, среда описывается МППР.

1.3. ФОРМИРОВАНИЕ НАГРАДЫ В ПЕРВОЙ СРЕДЕ

Для начала необходимо выбрать по какому принципу будет начисляться награда. Существует два главных принципа: разреженная награда и плотная награда. В случае плотной награды осмысленно оцениваются все шаги агента. Например, награда на каждом шаге пропорциональная расстоянию до цели, чем ближе к цели – тем больше награда. В случае разреженной награды агент оценивается только в случае достижения какого-либо результата, будь то достижение цели или провал. Промежуточные шаги либо оцениваются константой, либо не оцениваются вовсе.

Чаще всего используется первый принцип оценки агента, как самый интуитивный. Однако второй принцип имеет неочевидные преимущества[3]. Сравнение данных принципов показывает, что плотная награда обеспечивает более

стабильное и быстрое обучение. Но оценка каждого шага агента в некоторой степени ограничивает его свободу действий. Может так оказаться, что стратегия, форсируемая наградами, не является оптимальной. Разреженная награда не имеет такой проблемы. При успешном обучении агент будет действовать оптимально. При этом сам процесс обучения в этом случае нестабилен и медленен.

Для данной среды будет использован принцип плотной награды, так как формулировка проблемы является простой и позволяет сформировать оптимальную стратегию с помощью наград.

Начисление награды на каждом шаге будет происходить следующим образом:

- $-|L_{\text{цель}}|$,
- $-e^{-(|L_{\text{враг}}|-2)}$ для каждого врага,

где L - вектор координат.

Следующим шагом является масштабирование наград. Масштаб имеет значительное влияние на эффективность обучения[1, с.7]. При малом масштабе политика модели становится почти равномерной, что предотвращает обучение. При большом масштабе модель быстро учится в начале, но в дальнейшем политика модели становится почти детерминированной, что ведет к застреванию в локальных минимумах. Для алгоритма, который будет использован, наилучшей является награда, масштабированная в пределах двухзначных чисел. Таким образом, масштабированная награда, с учетом максимального количества шагов:

- $-10^{-4} \cdot |L_{\text{цель}}|$,
- $-0.003 \cdot e^{-(|L_{\text{враг}}|-2)}$ для каждого врага.

1.4. ПРОСТРАНСТВО ДЕЙСТВИЙ АГЕНТА В ПЕРВОЙ СРЕДЕ

Для полноценного взаимодействия со средой агенту необходимо перемещаться. Для контроля перемещения будет использоваться двумерный вектор (x, y), который будет задавать направление движения агента.

Выходными данными модели будут являться два числа, выступающими значениями вектора перемещения. Значения x и y будут принадлежать континуальному пространству действий, так как в данной ситуации оно обеспечивает боле тонкий контроль, нежели дискретное пространство действий. Причем $x, y \in [-1; 1]$.

1.5. ОПИСАНИЕ ВТОРОЙ СРЕДЫ

Так же, как и для предыдущей среды, используются Unity с расширением ML-Agents, библиотеки PyTorch и Gymnasium.

Среда обучения представляет собой лабиринт с квадратным типом сетки. Лабиринт имеет простую структуру, без циклов и только один выход.

Точка появления агента в начале эпизода, как и точки появления начальной комнаты, фиксирована и находится в начале координат. От начальной комнаты генерируется лабиринт, уникальный для каждого эпизода. Алгоритм генерации лабиринта следующий: при появлении комнаты, с шансом 25%, вместо стены генерируется соседняя комната. Процесс происходит до тех пор, пока не выполнится квота количества комнат. В последней созданной комнате создается выход. Эпизод завершается если агент доходит до цели, количество шагов двухкратно превышает количество комнат или суммарная награда эпизода становится ниже 100.

Визуально среда представлена на рисунке 4. Где синим обозначен агент, красным – стены, а зеленым – цель.

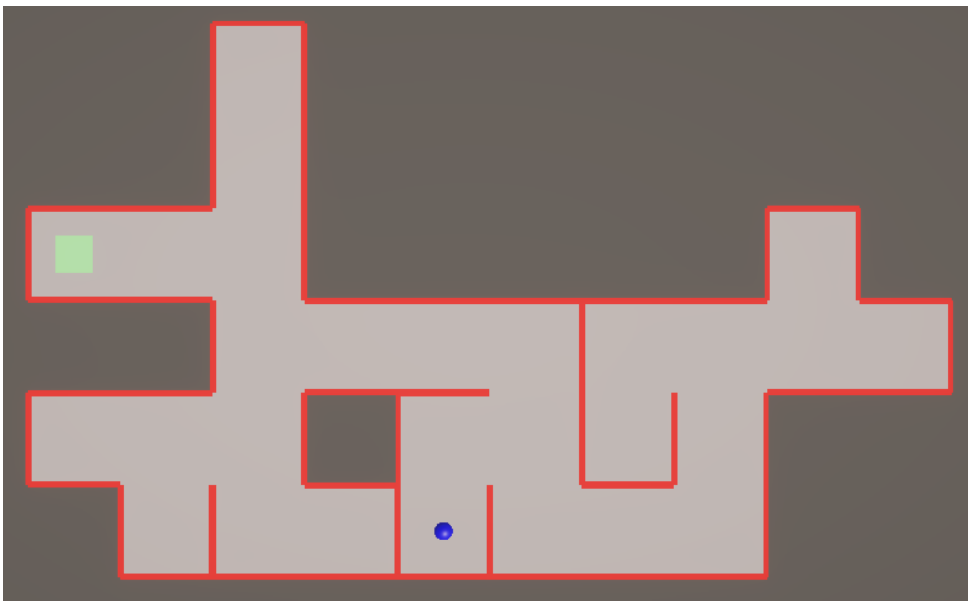

Рисунок 4 – Визуальное представление среды

1.6. НАБЛЮДЕНИЯ ВО ВТОРОЙ СРЕДЕ

Состояние среды однозначно описывается координатами агента, цели и параметрами каждой комнаты (сведения о соседних комнатах).

В качестве предоставляемых средой наблюдений взять состояние, как есть, не получится. В таком случае модель будет обучаться проходить конкретный лабиринт эпизода. А так как в начале эпизода генерируется случайный лабиринт, смысла у такого обучения нет. По этой причине рассматривать данную среду как описываемую МППР нельзя. Тогда в качестве наблюдений необходимо взять контекстную информацию.

Тогда предоставляемые наблюдения выглядят следующим образом:

- тип наблюдаемого объекта (стена, проход или цель) для каждого из четырех направлений,
- количество посещений объекта для каждого из четырех направлений. В случае стены – большое число,
- количество перемещений из текущей комнаты в каждое из четырех направлений.

Также хорошей практикой является нормирование наблюдений для более стабильного обучения. В данном случае удобно будет взять обратные величины для количественных наблюдений.

В итоге имеем:

- тип наблюдаемого объекта (стена, проход или цель) для каждого из четырех направлений,
- число обратное количеству посещений объекта для каждого из четырех направлений. В случае стены – 0,
- число обратное количеству перемещений из текущей комнаты в каждое из четырех направлений.

Так как наблюдение описывает состояние среды контекстно, среда описывается ЧНМППР.

1.7. ФОРМИРОВАНИЕ НАГРАДЫ ВО ВТОРОЙ СРЕДЕ

Как и в случае предыдущей среды, выберем принцип плотной награды, так как оптимальная стратегия агента в среде очевидна. К тому же, при использовании разреженного принципа награды присутствует большой шанс расхождения, так как траектории действий для успешного завершения эпизода относительно длинные.

Начисление награды на каждом шаге будет происходить следующим образом:

- при переходе в соседнюю комнату: $-(s_1 + s_2)$,
- при попытке пройти в стену: -10,
- при успешном нахождении цели: 100,

где s_1 – количество посещений соседней комнаты в выбранном направлении, s_2 – количество перемещений в данном направлении из текущей комнаты.

Выполним масштабирование награды, также в пределах 100:

- при переходе в соседнюю комнату: $-10\frac{s_1+s_2}{2r}$,
- при попытке пройти в стену: $-\frac{100}{2r}$,
- при успешном нахождении цели: 100,

где r – количество комнат лабиринта.

1.8. ПРОСТРАНСТВО ДЕЙСТВИЙ АГЕНТА ВО ВТОРОЙ СРЕДЕ

Для полноценного взаимодействия со средой агенту необходимо делать шаги в соседние комнаты. Для этой цели подойдет дискретное пространство действий, так как свободное перемещение от агента не требуется.

Выходными данными модели будет являться число, обозначающее действие:

- 0 – шаг влево,
- 1 – шаг вправо,
- 2 – шаг вверх,
- 3 – шаг вниз.

2. МЯГКИЙ АКТЕР-КРИТИК

2.1. ОПИСАНИЕ МЕТОДА

В качестве алгоритма для обучения был выбран мягкий актер-критик[1], так как он является одним из передовых алгоритмов в обучении с подкреплением. Мягкий актер-критик был представлен в 2018 году как алгоритм, призванный решить проблемы безмодельных алгоритмов.

Одной из таких проблем является низкая эффективность использования данных у алгоритмов с жадной политикой обновления. Такие алгоритмы требуют новые данные для каждого градиентного спуска, что приводит к быстрому росту вычислительной сложности вместе с ростом сложности задачи.

Алгоритмы, не использующие жадную политику обновлений, используют предшествующий опыт агента и не требуют новых данных для каждого градиентного спуска. Таким образом эффективность использования данных у таких алгоритмов лучше. Однако при использовании их в многомерных задачах, алгоритмы становятся крайне нестабильными и чувствительными к гиперпараметам.

Мягкий актер-критик является алгоритмом с не жадной политикой обновления, поэтому эффективно использует данные. При этом используется фреймворк максимальной энтропии вместо простой максимизации награды, что решает проблему нестабильности и сильной зависимости от гиперпараметров.

2.2. ПОСТРОЕНИЕ ФУНКЦИЙ ДЛЯ ГРАДИЕНТНОГО СПУСКА

Учитывая фреймворк актер-критик, в алгоритме будут использоваться параметризированная Q-функция (критик) $Q_\theta(s_t, a_t)$ и политика (актер) $\pi_\Phi(a_t|s_t)$. Параметрами данных сетей являются θ и Φ соответственно. Стоит заметить, что отдельной сети для функции состояния нет, так как она может быть вычислена

с помощью функции 1. Политика реализует континуальное пространство действий, соответственно действия агента могут быть смоделированы как экземпляры нормального распределения с мат. ожиданием и дисперсией, данными моделью.

$$V(s_t) = \mathbb{E}_{\boldsymbol{a}_t \sim \pi}[Q(s_t, a_t) - log\pi(a_t|s_t)]. \tag{1}$$

Q-функция обучается минимизировать остаток Бэллмана

$$J_Q(\theta) = \mathbb{E}_{(s_t,a_t)\sim\mathbb{D}}\left[\frac{1}{2}(Q_\theta(s_t, a_t) - \hat{Q}(s_t, a_t))^2\right], \tag{2}$$

где $\mathbb{D}$ – множество предыдущих наблюдений;

$$\hat{Q}(s_t, a_t) = r(s_t, a_t) + \gamma\mathbb{E}_{s_{t+1}\sim\rho}[V_{\bar{\theta}}(s_{t+1})],$$

где $r(s_t, a_t)$ – награда за действие a_t при наблюдении s_t; $V_{\bar{\theta}}(s_{t+1})$ вычисляется с помощью формулы 1. Данная функция оптимизируется следующими стохастическими градиентами

$$\begin{aligned}\hat{\nabla}_\theta J_Q(\theta) = \nabla_\theta Q_\theta(s_t, a_t)(Q_\theta(s_t, a_t) - (r(s_t, a_t) + \\ \gamma(Q_{\bar{\theta}}(s_{t+1}, a_{t+1}) - \alpha log(\pi_\Phi(a_{t+1}|s_{t+1}))))).\end{aligned} \tag{3}$$

Такой шаг обновления функции использует целевую функцию Q с параметрами $\bar{\theta}$, которая является экспоненциальным скользящим средним Q-функции. Это позволяет стабилизировать обучение.

Политика π обучается минимизируя расстояние Кульбака-Лейблера

$$J_\pi(\Phi) = \mathbb{E}_{s_t\sim\mathbb{D}}[\mathbb{E}_{a_t\sim\pi_\Phi}[\alpha log(\pi_\Phi(a_t|s_t)) - Q_\theta(s_t)]]. \tag{4}$$

Минимизация происходит с помощью нейронной сети, соответственно функция должна быть дифференцируема. Но a_t берется случайным образом из нормального распределения и эта операция не позволяет использовать метод обратного распространения ошибки. Для решения этой проблемы используется репараметризация, принцип работы которой заключается в следующем: взять экземпляр x из распределения с мат. ожиданием μ и среднеквадратичным отклонение σ эквивалентно взятию экземпляра e из распределения $\mathbb{N}(0, 1)$ и вы-

полнением $x = \sigma e + \mu$. При таком вычислении a_t можно использовать метод обратного распространения ошибки с учетом параметров μ и σ. Обозначим репараметризированный a_t как ϵ_t. Таким образом, политика оптимизируется с помощью следующих стохастических градиентов

$$\begin{aligned} \hat{\nabla}_\Phi J_\pi(\Phi) = \nabla_\Phi \alpha log(\pi_\Phi(a_t|s_t)) + \\ (\nabla_{a_t} \alpha log(\pi_\Phi(a_t|s_t)) - \nabla_{a_t} Q(a_t, s_t)) \nabla_\Phi \epsilon_t. \end{aligned} \tag{5}$$

2.3. АВТООБНОВЛЕНИЕ ПАРАМЕТРА ТЕМПЕРАТУРЫ

Параметр α является важным параметром, определяющим насколько свободно агент исследует среду. Но задавать его как фиксированный параметр не является оптимальным выбором, так как подбор α нетривиален и необходим для каждой отдельной задачи[2]. Более того, даже в рамках одной задачи параметр должен варьироваться. В состояниях среды, где оптимальное действие еще не известно, он должен позволять модели достаточное исследование среды. А в состояниях, где оптимальное действие известно, α должен приближать модель к детерминированной.

Параметр α аппроксимируется с помощью двойного градиентного спуска. Двойной градиентный спуск чередует оптимизацию лагранжиана, с α в качестве лагранжевой переменной, и градиентный шаг по двойным переменным[2, с.6]. Таким образом, градиент для α вычисляется из функции 6.

$$J(\alpha) = \mathbb{E}_{a_t \sim \pi_t}[-\alpha log \pi_t(a_t|s_t) - \alpha \hat{\mathbb{H}}]. \tag{6}$$

2.4. ГРАДИЕНТНЫЙ ШАГ

На практике в модель добавляют вторую Q-функцию и целевую функцию для нее. При вычислении $\hat{Q}(s_t, a_t)$ и оптимизации политики используется Q-функция с минимальным значением на шаге. Это позволяет избежать переоценки функции состояния $V(s_t)$.

В итоге, с учетом формул 3, 5 и 6, градиентный шаг алгоритма выглядит следующим образом (уравнения 7).

$$\begin{aligned}
&\theta_i = \theta_i - \lambda_Q \hat{\nabla}_{\theta_i} J_Q(\theta_i) \text{ где } i \in \overline{1,2}, \\
&\Phi = \Phi - \lambda_\pi \hat{\nabla}_{\Phi} J_\pi(\Phi), \\
&\alpha = \alpha - \lambda \hat{\nabla}_{\alpha} J(\alpha), \\
&\bar{\theta}_i = \tau\theta_i + (1-\tau)\bar{\theta}_i \text{ где } i \in \overline{1,2}.
\end{aligned} \tag{7}$$

Где $\bar{\theta}_i$ – параметры целевой i-ой Q-функции, τ – смягчающий коэффициент, λ – темп обучения.

3. ОБУЧЕНИЕ АГЕНТА

3.1. ПЕРВАЯ СРЕДА

Для обучения агента были выбраны гиперпараметры, представленные в таблице 1.

Обучение проводилось на 10000 эпизодах. Из графика зависимости средней награды (за последние 100 эпизодов) от шага обучения (рисунок 5) можно судить о том, что политика модели близка к оптимальной, так как в последней четверти графика средняя награда стабильна и близка к 100. При том, что среда спроектирована таким образом, чтобы награда за эпизод была ограничена, $-100 \leq r \leq 100$.

Протестируем среду на обобщенность путем смещения абсолютных координат начального положения объектов. Результаты тестирования представлены в таблице 2. Можно сделать вывод, что модель инвариантна к смещениям координат. А так как состояние среды в полной мере описывается только координатами объектов, можно сказать, что она достаточно обобщена.

Также, при обучении моделей в среде, отсутствует проблема переобучения, так как стартовые координаты агента и врагов случайны. Поэтому в начале каждого эпизода система находится в уникальном состоянии, что предотвращает переобучение.

Таблица 1 – Гиперпараметры для обучения во второй среде

Гиперпараметр	**Значение**
Размерность внутренних слоев нейросети	256
Темп обучения критика	3e-4
Темп обучения α	3e-4
Темп обучения актера	1e-4
Размер выборки	1000
γ	0.99
τ	0.005

Таблица 2 – Тестирование среды на чувствительность к смещению

Смещение (x, y)	**Количество успешных эпизодов из 50**
$(100, 0)$	46
$(100, 100)$	46
$(-100, 0)$	46
$(-100, -100)$	48
$(0, -100)$	45

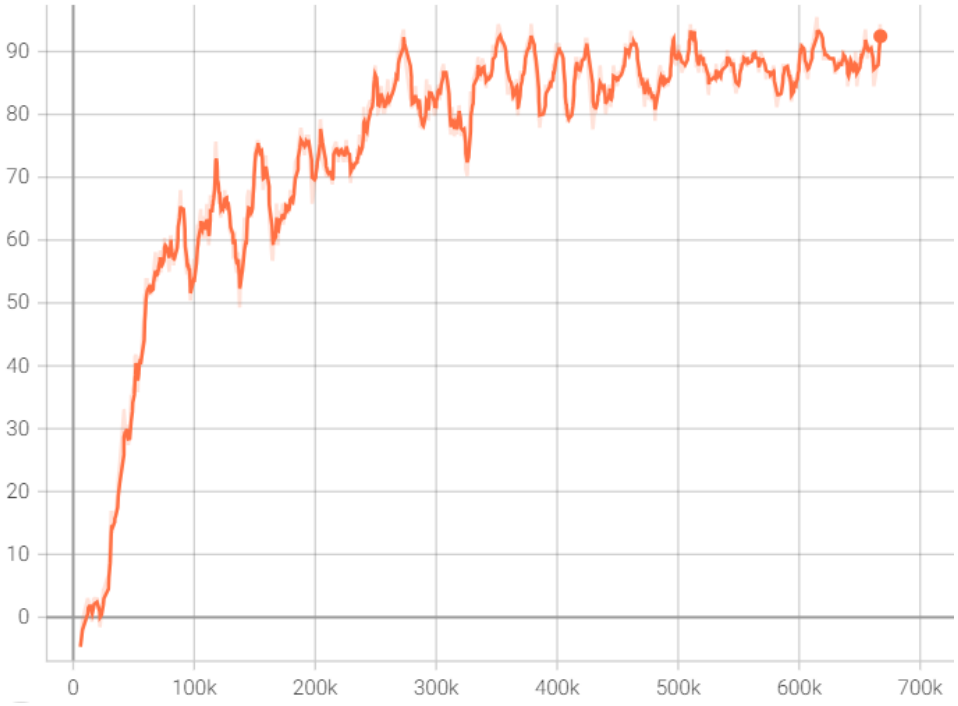

Рисунок 5 – График средней награды за время обучения

3.2. ВТОРАЯ СРЕДА

Для обучения агента были выбраны гиперпараметры, представленные в таблице 3.

Обучение проводилось на 10000 эпизодах. Как и в случае предыдущей среды, из графика зависимости средней награды (рисунок 7) можно судить о том, что политика модели близка к оптимальной.

Однако даже несмотря на оптимальную политику, при тестировании агентаможно заметить, что агент не всегда совершает оптимальные действия, что свойственно для нейросетевых моделей. При этом можно заметить, что задача легко решается алгоритмическим путем. Сравним работу простого алгоритма 1 и обученной модели.

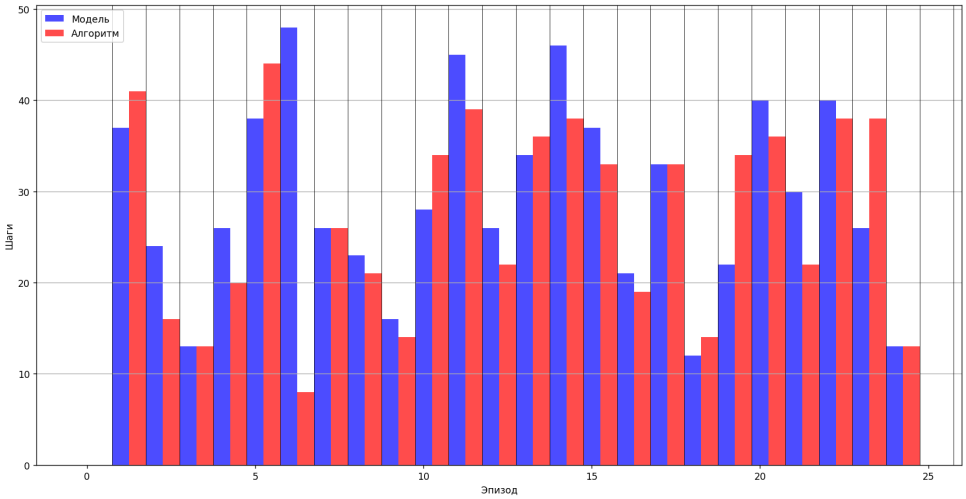

Рисунок 6 – График средней награды за время обучения

Из диаграммы 6 видно, что модель показала лучший результат меньше чем в половине случаев (11 из 25). Это можно объяснить тем, что при данных наблюдениях среды задача легко поддается алгоритмизации. Поэтому производительность модели, даже при идеальной работе, не превзойдет алгоритм, а лишь сравняется с ним. У модели банально отсутствуют данные, которое она могла бы эффективно использовать для более оптимальной работы.

Algorithm 1 Алгоритм решения лабиринта

while цель не достигнута **do**
  if цель находится в соседней комнате **then**
    шаг в комнату с целью
  else
    отбросить направления ведущие в стену
    выбрать направления в котором минимальное количество шагов
    выбрать направление с комнатой с минимальным кол-вом посещений
    сделать шаг в выбраном направлении
  end if
end while

Таблица 3 – Гиперпараметры для обучения во второй средей

Гиперпараметр	**Значение**
Размерность внутренних слоев нейросети	64
Темп обучения критика	3e-4
Темп обучения α	3e-4
Темп обучения актера	1e-4
Размер выборки	1000
γ	0.99
τ	0.005

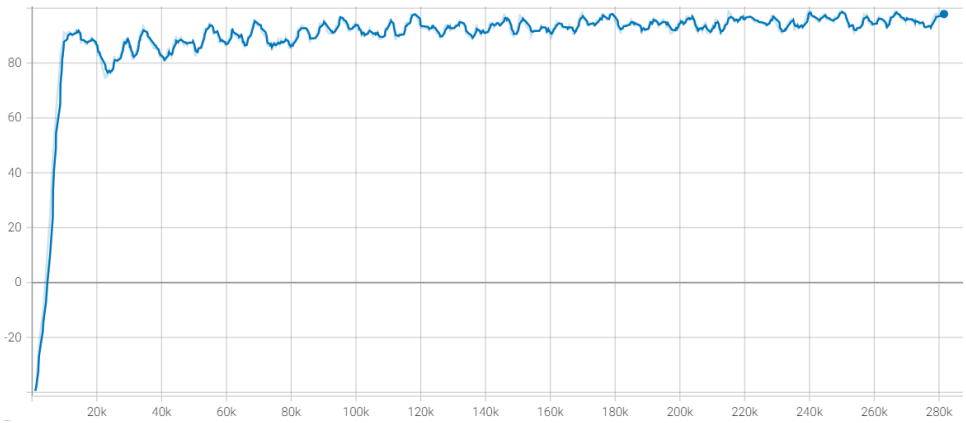

Рисунок 7 – График средней награды за время обучения

ЗАКЛЮЧЕНИЕ

В результате выполнения практики были спроектированы две среды, описываемые МППР и ЧНМППР соответственно. В ходе проектирования сред были выбраны наблюдения, такие, чтобы максимально обобщить состояние среды. Также были выбраны принципы оценки агента и сформированы награды за шаг обучения, после чего награды были масштабирована для лучшей производительности.

Как итог, в первой среде была успешно обучена модель. Дальнейшие тесты показали, что добавление смещения к состоянию среды не влияет на производительность ранее обученной модели, что подтверждает обобщенность среды.

Во второй среде также была успешно обучена модель. Однако в силу ограниченности наблюдений и легкой алгоритмизации задачи производительность модели не была оптимальной. Стоит заметить, что большинство алгоритмов обучения с подкреплением подразумевают, что среда обучения описывается МППР. Поэтому среды ЧНМППР перед началом обучения требуют приведения к более близкому к МППР виду, что не является тривиальной задачей.

СПИСОК ЛИТЕРАТУРЫ

[1] Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor [Electronic resource] / Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, Sergey Levine – Mode of access: https://arxiv.org/pdf/1801.01290 - Title from screen.

[2] Soft Actor-Critic Algorithms and Applications [Electronic resource] / Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, Sergey Levine – Mode of access: https://arxiv.org/pdf/1812.05905v2 - Title from screen.

[3] Revisiting Sparse Rewards for Goal-Reaching Reinforcement Learning [Electronic resource] / Gautham Vasan, Yan Wang, Fahim Shahriar, James Bergstra, Martin Jagersand, A. Rupam Mahmood – Mode of access: https://arxiv.org/pdf/2407.00324 - Title from screen.

[4] Unity ML-Agents [Электронный ресурс]. URL: https://github.com/Unity-Technologies/ml-agents/tree/develop (дата обращения 16.12.2024).

ПРИЛОЖЕНИЕ

Проект Unity, включающий в себя обе среды, а также код метода мягкого актера-критика находятся в следующем GitHub репозитории: https://github.com/Vefery/Practice/.