Московский государственный университет имени М.В. Ломоносова
Факультет Вычислительной Математики и Кибернетики

Магистерская программа
«Логические и комбинаторные методы анализа данных»

Магистерская диссертация

«Стратегии исследования окружений в обучении с подкреплением с непрерывными пространствами состояний»

Работу выполнил:
Гурьянов Алексей Константинович

Научный руководитель:
профессор, д.ф.-м.н.
Дьяконов Александр Геннадьевич

Москва
2017

Содержание

Аннотация

Для решения задач на стыке машинного обучения и оптимального управления была создана область обучения с подкреплением, фокусирующаяся на обучении поведения агента в определенных окружениях для достижения определенного результата. В работе агентов выделяют важную проблему выбора между эксплуатацией текущих знаний агента для получения наибольшей награды и исследованием окружения с целью получения знаний о структуре окружения. В данной работе ставилась задача изучения стратегий исследования окружений. Были предложены модификации существующих стратегий исследования и новая стратегий исследования окружений в обучении с подкреплением. Было обнаружено, что не существует стратегии исследования, показывающей наилучшее качество на всех окружениях. Предложенный метод показал наилучшее качество на одном из рассматриваемых окружений.

1 Введение

С развитием современных технологий человечество ставит себе на службу системы с все более и более возрастающей сложностью. При разработке таких систем возникает проблема оптимального их контроля, которую зачастую невозможно решить реализацией какого-то определенного алгоритма. Для решения подобных задач на стыке областей машинного обучения и оптимального управления была создана область обучения с подкреплением, фокусирующаяся на обучении поведения агента в определенных окружениях для достижения определенного результата. Задачи, в которых эти методы нашли применения, варьируются от задач управления роботами и контроля индустриальных процессов до управления портфелями ценных бумаг и участием в компьютерных играх. В последнее время популярными стали алгоритмы обучения с подкреплением, использующие нейронные сети над состояниями окружений. Такая архитектура позволяет решать задачи с непрерывным пространством состояний.

Задача обучения с подкреплением предполагает наличие агента, получающего определенную информацию о состоянии окружения и способного совершать действия, оказывающие влияние на окружение. Цель обучения с подкреплением состоит в обучении агента способу получения максимального значения скалярного выражения числовой награды, которая выдается агенту при совершении определенных действий в определенных окружениях. Одним из базовых понятий в области обучения с подкреплением является Марковский процесс принятия решений. Окружение, которое можно описать через Марковский процесс принятия решений, однозначно стохастически определяет изменение состояния через предыдущее состояние и предпринятое агентом действие.

В работе агентов выделяют важную проблему выбора между эксплуатацией текущих знаний агента для получения наибольшей награды и исследованием окружения с целью получения знаний о структуре окружения. Эти два варианта выбора действий подразумевают под собой одну и ту же итоговую цель, один вариант максимизирует полученную награду в ближайшем будущем, а другой – в долгосрочном периоде. Выбор случайных действий с целью исследования дает очень плохой результат с точки зрения скорости схождения к оптимальной стратегии. Существуют стратегии исследования, гарантированно приводящие агента к оптимальной стратегии действий, что показано в имеющихся исследованиях на Марковских процессах принятия решений с маленькой размерностью пространства состояний, но на задачах с большой или бесконечной размерностью пространства состояний такие методы становятся неприменимы из-за больших вычислительных затрат, что приводит к необходимости применять достаточно простые стратегии исследования, и к необходимости исследования более эффек-

тивных механизмов исследования окружений. Одна из трудностей в исследовании стратегий исследования состоит в том, что в то время как оптимальное с эксплуатационной точки зрения действие естественным образом определяется как действие, максимизирующее ожидаемую награду, не существует принятого определения для оптимального действий с точки зрения исследования. Еще большие трудности в формализацию понятия исследования вносит тот факт, что во многих современных методах обучения с подкреплением агент не хранит информацию об окружении напрямую.

Задачами данной работы являются:

- Изучение существующих методов исследования окружений.
- Выделение наилучших методов исследования окружений с точки зрения максимальной достигнутой награды и скорости обучения относительно затраченного в симуляции времени.
- Построение нового метода исследования окружений и экспериментальное сравнение с существующими методами.

Для построения нового метода предлагается использование комбинации нескольких методов исследования окружений. Данная работа состоит из следующих разделов: постановка задачи и введение основных терминов; обзор существующих методов обучения с подкреплением и стратегий исследования на них, описание предложенного нового метода исследования окружений в алгоритме Q−обучения, описание программной реализации, результаты работы нового метода; а также сравнение нового метода с существующими на примере различных задач обучения с подкреплением.

2 Постановка задачи

Основной целью данной работы является нахождение алгоритма исследования окружений, позволяющий получить наибольшую награду или наибольшую скорость достижения этой награды при условии равенства ее итогового размера.

В этой задаче можно выделить следующие подзадачи:

- Выделить перспективные алгоритмы исследования окружений.
- Провести исследование характеристик выделенных методов.
- Предложить новый метод исследования окружений, превосходящий существующие.

3 Задача обучения с подкреплением

3.1 Постановка задачи

Под задачей обучения с подкреплением понимается задача обучения из взаимодействия для достижения какой-то цели. Обучающаяся и принимающая решения единица называется агентом. Объект, с которым агент взаимодействует, называется окружением. Они постоянно взаимодействуют, агент выбирает действия и окружение отвечает на эти действия и предоставляет новую ситуацию агенту. Окружающая среда также порождает награду, специальное численное значение, которое агент пытается максимизировать за время своей работы. Полное описание окружающей среды определяет одну из задач области обучения с подкреплением.

Более точно, агент и окружение взаимодействуют на каждом шаге последовательности дискретных временных шагов $t = 0, 1, 2, 3, \ldots$. На каждом шаге агент принимает какое-то представление окружения в качестве состояния $S_t \in S$, где S является множеством всех возможных состояний, и на основании состояния выбирает действие $A_t \in A(S_t)$, где $A(S_t)$ описывает множество действий доступных агенту в состоянии S_t. На следующем временном шаге, в том числе и под воздействием принятого действия, агент принимает численную награду $R_{t+1} \in R$, а также новое состояние, S_{t+1}. На каждом временном шаге агент сопоставляет состоянию набор вероятностей выбора каждого доступного действия. Это сопоставление называется стратегией агента, и описывается π_t, где $\pi_t(a|s)$ описывает вероятность того, что действие $A_t = a$ будет выбрано в состоянии $S_t = s$. Методы обучения с подкреплением определяют способ, которым агент меняет свою стратегию в результате полученного опыта. Целью агента, грубо говоря, является максимизация полного значения полученной награды.[1]

Подобная схема абстрактна и гибка и может быть применена ко многим различным проблемам многими способами. В общем случае, действия могут определяться любым решением, обстоятельствам применения которого мы хотим обучиться, а состояния могут содержать любую информацию, полезную в принятии этих решений.

Общее правило, которого следует придерживаться, говорит, что все, что не может быть изменено агентом произвольным образом, считается находящимся вне этого агента, и, таким образом, является частью окружения. Схема не предполагает, что все в окружении известно агенту. Например, агент часто много знает о способе вычисления его наград как функции состояния и действия. Но мы всегда предполагаем вычисление награды как внешнее к агенту, потому что оно определяет задачу, поставленную перед агентом, и таким образом у агента не должно быть возможности менять эту функцию произвольным образом. В некоторых случаях агент может знать все о том, как устрое-

но его окружение, и все равно иметь некоторые трудности при решении задачи обучения с подкреплением. Эта ситуация аналогична тому как мы можем знать все о пазле кубик-рубик, но все равно можем не уметь его решить. Граница агента и окружения представляет границу абсолютного контроля агента, но не границу знаний агента.

Граница агента и окружения может быть расположена в различных местах для решения различных задач. Например, в сложном роботе несколько разных агентов могут работать единовременно, каждый со своей собственной границей. Один агент может принимать высокоуровневые решения, на основе которых будут формироваться состояния для более низкоуровневых агентов. На практике, граница агента и окружения определена после того как определены множества состояний, действий и наград, и таким образом определена конкретная задача управления.

В задаче обучения с подкреплением, цель агента формализуется в понятиях специального сигнала награды, который предается от окружения агенту. На каждом временном шаге награда представляется одним числом $R_t \in R$. Неформально говоря, целью агента является максимизация полного размера полученной награды. Это означает максимизацию не только немедленной награды, но и суммарной награды за долгий промежуток времени.

Использование сигнала награды для формализации понятия цели является одной из отличительных характеристик задачи обучения с подкреплением. На практике подобный метод показал свою гибкость и широкую применимость. Например, для того, чтобы обучить робота ходить, исследователи предоставляли награду на каждом временном шаге пропорциональную достигнутой роботом скорости движения вперед. Для обучения робота побегу из лабиринта, награда часто задается равной нулю, до момента действия, приводящего к успешному побегу, после которого она задается равной единице.

Награды не должны включать в себя поощрения о том, как агенту следует достигать поставленной цели. В случае, например, шахмат это значит, что мы не должны включать в награду подцели, такие как удержание центра доски или взятие вражеских фигур, так как агент может найти способ удовлетворения подцелей без достижения главной цели, например пытаясь взять как можно больше вражеских фигур, даже ценой проигранной партии.

Пусть последовательность наград, получаемых после временного шага t, записываются как $R_{t+1}, R_{t+2} \ldots$. Тогда в общем случае в задаче обучения с подкреплением необходимо максимизировать ожидаемую награду, G_t, которая определена как некоторая функция от последовательности наград.

Самый простой способ заключается в сумме всех последующих наград:

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_T$$

где T задает последний временной шаг. Этот подход имеет смысл только при существовании в задаче понятия о финальном временном шаге, что приводит к разбиению взаимодействия агент и окружения на подпоследовательности, которые называются эпизодами, такими как одна партия игры, одно путешествие в лабиринте или какое-либо другое повторяющееся взаимодействие. Каждый эпизод заканчивается особым терминальным состоянием, за которым следует переход к стандартному начальному состоянию или к выбору начального состояния из распределения начальных состояний. Задачи с подобными эпизодами называются эпизодическими задачами. В эпизодических задачах иногда возникает потребность в разделении нетерминальных состояний,обозначающихся S от набора терминальных состояний, обозначающихся S^+.

Но во многих случаях взаимодействие агента и окружения не разбивается на легко опознаваемые эпизоды, а продолжается постоянно без предела. Одним из таких случаев является задача продолжительной задачи управления, или задача управления роботом с большим сроком эксплуатации. Такие задачи называются продолжительными задачами. Вышеописанная формула вызывает проблемы в подобных задачах, так как последний временной шаг может не существовать, а ожидаемая награда, которую мы стремимся максимизировать, может достигать бесконечности. Для решения таких проблем вводится подход обесценивания. В соответствии с этим подходом, агент пытается выбрать действия таким образом, чтобы сумма обесценивающихся получаемых наград была наибольшей. То есть выбирается такое действие A_t, которое максимизирует ожидаемую обесцененную награду:

$$G_t = R_{t+1} + \lambda R_{t+2} + \lambda^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \lambda^k R_{t+k+1}$$

где λ задает параметр обесценивания, $0 \leq \lambda \leq 1$.

Параметр обесценивания задает ценность наград в будущем: награда размером в единицу, получаемая через k временных шагов, стоит только λ^{k-1} по сравнению с исходной наградой размером в единицу, в случае если бы мы получали ее немедленно. Если $\lambda < 1$, то бесконечная сумма имеет конечное значение при условии ограниченности последовательность наград R_k. Если $\lambda = 0$, то агент учитывает только немедленные награды, и действие выбирается с целью максимизации R_{t+1}. Если каждое из действий агента влияет только на непосредственную награду, то такой агент может одновременно максимизировать и ожидаемую обесцененную награду для $\lambda \neq 0$. Но в большинстве случаев

маленький параметр обесценивания ограничивает доступ агента к будущим наградам, а параметр обесценивания, близкий к единице, сильнее учитывает будущие награды.

3.2 Марковские процессы

Процесс обладает свойством Маркова, если все последующие изменения в процессе возможно описать только по последнему этапу процесса. Формулируя формально в контексте задач обучения с подкреплением мы получаем, что процесс удовлетворяет свойству Маркова, если для любых возможных значений прошедших событий $S_0, A_0, R_1, \ldots, S_{t-1}, A_{t-1}, R_t, S_t, A_t$ полное распределение вероятностей состояний, которое выражается по формуле

$$Pr\{R_{t+1} = r, S_{t+1} = s' | S_0, A_0, R_1, \ldots . S_{t-1}, A_{t-1}, R_t, S_t, A_t\} = Pr\{R_{t+1} = r, S_{t+1} = s' | S_t, A_t\}$$

Если окружение обладает свойством Маркова, тогда история одного последнего шага может быть использована для расчета следующего состояния и ожидаемой награды для текущих состояния и действия.

Задача обучения с подкреплением, которая удовлетворяет свойству Маркова называется Марковским процессом принятия решения (в дальнейшем МППР). Если пространство состояний и действий конечны, то задача называется конечным Марковским процессом принятия решений (конечным МППР).

Конкретный конечный МППР определен его наборами состояний и действий и одношаговой динамикой окружения. При условии любого состояния s и действия a, вероятность каждого возможного следующего состояния, s', равна

$$p(s'|s, a) = Pr\{S_{t+1} = s' | S_t = s, A_t = a\}$$

Аналогично, при условии любого текущего состояния и действия s и a, вместе с каждым следующим состоянием, s', ожидаемая функция награды равна

$$r(s, a, s') = E[R_{t+1} | S_t = s, A_t = a . S_{t+1} = s']$$

Эти величины, $p(s'|s, a)$ и $r(s, a, s')$ полностью задают наиболее важные аспекты динамики конечного МППР, утеряна только информация о распределении наград вокруг ожидаемой величины.

3.3 Функция ценности

Большинство алгоритмов обучения с подкреплением включает оценивание функций ценности − функции от состояния(или от пары состояния и действия), которая оценивает, насколько ценно для агента пребывать в заданном состоянии (или насколько ценно применить данное действие в данном состоянии). Понятие о том насколько ценно какое-то действие или состояние выражается в понятии будущей ожидаемой награды. Понятно, что награда, которую агент может рассчитывать получить в будущем зависит от принимаемых в дальнейшем действий. Соответственно функции ценности определены для отдельных стратегий принятия решений.

Вспомним, что стратегия π - это сопоставление состояния $s \in S$ и действия $a \in A(s)$ вероятности $\pi(a|s)$ принятия действия a в состоянии s. Неформально, ценность состояния s при стратегии π, равное $v_\pi(s)$, вычисляется как ожидаемая награда при начале из состояния s и выполнения политики π в будущем. Для Марковского процесса принятия решений мы можем записать $v_\pi(s)$ как

$$v_\pi(s) = E_\pi[G_t|S_t = s] = E_\pi\left[\sum_{k=0}^{\infty} \lambda^k r_{t+k+1} \middle| S_t = s\right]$$

Где $E_\pi\Big[\ \cdot\ \Big]$ показывает ожидаемое значение награды при действии агента в соответствии со стратегией π на каждом временном шаге. Заметим, что значение ценности терминального состояния всегда равно нулю. Функция v_π называется функцией ценности состояния для стратегии π.

Ценность действия a в состоянии s при стратегии π, обозначаемая $q_\pi(s, a)$, определяется как ожидаемое значение награды начиная с состояния s, принимая действие a и в дальнейшем придерживаясь стратегии π

$$\begin{aligned} q_\pi(s,a) &= E_\pi[G_t|S_t = s, A_t = a] \\ &= E_\pi\left[\sum_{k=0}^{\infty} \lambda^k R_{t+k+1} \middle| S_t = s, A_t = a\right] \end{aligned}$$

q_π называется функцией ценности действия для стратегии π.

Функции ценности v_π, q_π могут быть оценены из опыта. Например, если агент следует стратегии π и для каждого состояния поддерживает среднее действительных значений наград, которые следовали за данных состоянием, то это среднее будет сходится к среднему этого состояния, по мере того как число пребываний в этом состоянии стремится к бесконечности.

Фундаментальное свойство функций ценности, которое используется в в методах обучения с подкреплением и динамического программирования, заключается в том, что оно удовлетворяет определенному рекурсивному отношению. Для каждой стратегии π и каждого состояния s, выполняется соотношение

$$\begin{aligned} v_\pi(s) &= E_\pi[G_t|S_t = s] \\ &= \sum_a \pi(a|s) \sum_{s'} p(s'|s,a)\Big[r(s,a,s') + \lambda v_\pi(s')\Big] \end{aligned}$$

Это соотношение называется уравнением Беллмана для v_π. Оно выражает отношение между значениями ценности состояния и значением ценности последующих состояний. Функция ценности состояния v_π выражает единственное решение уравнения Беллмана.

Решение задачи обучения с подкреплением означает нахождение стратегии, под действием которой агент получит много наград в долгосрочном периоде. Для конечных МППР мы можем точно определить оптимальную стратегию. Для этого введем частичный порядок на пространстве стратегий на основании функции ценности. Стратегия π является больше либо равной стратегии π' если ожидаемая награда для стратегии π больше чем ожидаемая награда для стратегии π' для всех состояний. В такой системе частичного порядка будет хотя бы одна стратегия, которая больше либо равна всех остальных стратегий. Она называется оптимальная стратегия. Все оптимальные стратегии обозначаются через π_*. У таких стратегий одинаковые функции ценности состояния, которые называются оптимальными функциями ценности состояния:

$$v_*(s) = \max_\pi v_\pi(s), \forall s \in S$$

У всех оптимальных стратегий одна и та же оптимальная функция ценности действия:

$$q_*(s,a) = \max_\pi q_\pi(s,a), \forall s \in S, a \in A(s)$$

Для каждой пары состояния и действия эта функция выдает ожидаемую награду при взятии действия a в состоянии s, а затем придерживаясь оптимальной стратегии. Таким образом возможно выразить оптимальную функцию ценности действия через оптимальную функцию действия состояния.

$$q_*(s,a) = E[R_{t+1} + \lambda v_*(S_{t+1})|S_t = s, A_t = a]$$

3.4 Фундаментальные методы решения

3.4.1 Динамическое программирование

Основные методы динамического программирования включают в себя итерирование стратегий Ховарда и итерирование ценности Беллмана. Алгоритм итерирования стратегий Ховарда во время одной итерации сначала вычисляет функцию награды для данной стратегии, а потом улучшают эту стратегию с помощью максимизации функции награды [2]

Основная идея алгоритма динамического программирования Беллмана заключается в использовании функций награды для структурирования поиска хороших стратегий. Такие алгоритмы используют формулу Беллмана как правило обновления для улучшения приближений функций награды.[1]

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s'} p(s'|s,a)\Big[r(s,a,s') + \lambda v_\pi(s')\Big], \forall s \in S$$

Если схема функционирования окружения известны, то уравнения Беллмана сводится к $|S|$ линейным уравнениям с $|S|$ неизвестными $(v_\pi(S), s \in S)$.Прямое решение подобной системы приведет к ответу, но будет требовать больших затрат вычислительных ресурсов. Часто для нахождения функции ценности используют итерационные методы нахождения линейной системы уравнений. Исходное приближение v_0 выбирается произвольным образом, а каждое последующее приближение находится через уравнение Беллмана, интерпретируемого как правило обновления:

$$\begin{aligned} v_{k+1} &= E_\pi[R_{t+1} + \lambda v_k(S_{t+1})|S_t = s] \\ &= \sum_a \pi(a|s) \sum_{s'} p(s'|s,a)\Big[r(s,a,s') + \lambda v_l(s')\Big] \end{aligned}$$

Такой алгоритм называется итерационным оцениванием стратегий.

Для вычисления каждого последующего приближения из предыдущего итерационное оценивание стратегий применяет ту же операцию к каждому состоянию s: оно заменяет старые значения s новым приобретенным значением. Алгоритм заканчивает свою работу, когда изменение оценки для каждого состояния не становится достаточно маленьким, что показывает близость результата алгоритма к предельному значению процесса.

После вычисления функции награды для каждого состояния при фиксированной стратегии, алгоритмы динамического программирования используют ее для улучшения

текущей стратегии. Для обоснования выбора улучшенной стратегии используется теорема об улучшении стратегии, которая гласит, что если для двух стратегий π и π' для любого состояния $s \in S$ верно, что $q_\pi(s, \pi'(s)) \geq v_\pi(s))$, то стратегия π' даст большую ожидаемую награду на всех состояниях $s \in S$: $v_\pi \geq v_\pi(s)$.

В соответствии с теоремой улучшения стратегии мы можем создать такую стратегию, которая выбирает наилучшее действие в каждом состоянии в соответствии с данной функцией награды:

$$\begin{aligned} \pi'(s) &= \text{argmax}_a q_\pi(s, a) \\ &= \text{argmax}_a E[R_{t+1} + \lambda v_\pi(S_{t+1}) | S_t = s, A_t = a] \\ &= \text{argmax}_a \sum_{s'} p(s'|s, a) \Big[r(s, a, s') + \lambda v_\pi(s') \Big] \end{aligned}$$

Такая стратегия соответствует условию теоремы улучшения стратегии, поэтому она будет не хуже, чем исходная стратегия π. Причем если функция награды новой стратегии не улучшает исходную стратегию, из правила обновления стратегии будет следовать, что

$$\begin{aligned} v_{\pi'}(s) &= max_a E[R_{t+1} + \lambda v_{\pi'}(S_{t+1}) | S_t = s, A_t = a] \\ &= max_a \sum_{s'} p(s'|s, a) \Big[r(s, a, s') + \lambda v_{\pi'}(s') \Big] \end{aligned}$$

Эта формула является записью уравнения Беллмана, из чего будет следовать, что обе стратегии π и π' являются оптимальными. Таким образом процесс улучшения стратегии выдает строго лучшую стратегию, за исключением тех случаев, когда текущая стратегия оптимальная.

Процесс построения новой стратегии, которая улучшает исходную, с помощью жадного выбора действия на основе функции награды исходной стратегии называется процессом улучшения стратегии.

После построения улучшенной стратегии π' по исходной стратегии π и ее функции награды v_π, новую стратегию π' можно таким же образом оценить и улучшить. Таким образом последовательное применение оценивания и улучшения стратегий выдает последовательность монотонно улучшающихся стратегий и функций награды. В случае конечного МППР количество стратегий конечно, из чего будет следовать, что такой процесс в конце концов сойдется к оптимальной стратегии и оптимальной функции награды за конечное количество шагов. Такой способ нахождения оптимальной стратегии называется итерированием стратегий.

Большой недостаток алгоритмов динамического программирования состоит в том, что их применение включает в себя операции, использующие все множество возможных состояний. Если пространство этих состояний очень велико, то даже один проход по всему этому пространству может быть слишком вычислительно дорог.

Асинхронные алгоритмы динамического программирования - это алгоритмы динамического программирования, которые не требуют систематического прохода по всему множеству состояний. Эти алгоритмы сохраняют значения состояний в произвольном порядке, и некоторые состояния могут быть сохранены несколько раз, в то время как другие не будут сохранены ни разу, но для корректного схождения каждое состояние должно быть рассчитано хотя бы раз.

Игнорирование полных проходов по состояниям не влечет за собой гарантированного уменьшения вычислительных затрат, но это приводит к возможности улучшать политику агента без необходимости дожидаться конца прохода.

3.4.2 Методы Монте-Карло

Методы Монте-Карло (МК) не предполагают полного знания об окружающей среде, в отличии от динамического программирования. Методы Монте-Карло требуют наличие опытных данных, под которым понимается набор последовательностей состояний, действий и наград, полученных в результате онлайн или смоделированного взаимодействия с окружающей средой. Хотя для смоделированных опытных данных и требуется модель, данной модели требуется только генерировать выборки перемещений, а не полное распределение всех возможных перемещений, как того требует динамическое программирование. [1]

Методы Монте-Карло решают задачу обучения с подкреплением, основываясь на средней ценности выборки. Чтобы удостовериться, что ценность будет определенной, методы Монте-Карло определяются только для эпизодических задач. Это предполагает, что опытные данные разделены на эпизоды, и что все эпизоды в итоге заканчиваются, вне зависимости от того, какие действия выбраны. И только после завершения эпизода происходит оценивание значений и стратегии изменяются. Таким образом, методы Монте-Карло являются инкрементными на уровне эпизодов, а не на уровне шагов.

МК-метод всех посещений вычисляет $v_\pi(s)$ как среднее значение ценностей, которые были получены в результате посещения всех состояний s в некотором наборе эпизодов. МК-метод первого посещения усредняет только те значения, которые были получены в результате первых посещений состояния s.

Как метод первых посещений, так и метод всех посещений сходятся к $v_\pi(s)$ при стремлении числа посещений (или первых посещений) s стремящихся к бесконечности.

Важный факт о МК-методах состоит в том, что оценки для каждого состояния являются независимыми, в отличии от методов динамического программирования. В особенности, следует заметить, что вычисление оценки отдельного состояния не зависит от числа состояний. Это делает МК-методы особенно привлекательными в тех случаях, когда требуется оценить только некоторое подмножество состояний.

Если модель недоступна, то тогда полезно получить оценки действий, а не состояний. При наличии модели вполне достаточно иметь оценки состояний, чтобы определить стратегию. Для этого необходимо просмотреть на один шаг вперед и выбрать, какое действие приводит к наилучшей комбинации вознаграждения и следующего состояния. При отсутствии модели, однако, одних оценок состояния недостаточно. Необходимо в явном виде оценить значение каждого действия для того, чтобы в дальнейшем полученные оценки использовать для построения стратегии.

Задача оценки стратегии состоит в вычислении функции $q_\pi(s, a)$ – ожидаемой награды при начале в состоянии s, выполнении действия a, и следуя стратегии π. В данном случае МК-методы такие же, как и для оценки состояний. МК-метод всех посещений оценивает пару состояние-действие как среднее всех наград, которые были получены при посещении данного состояния и выборе данного действия. МК-метод первого посещения усредняет награды, которые были получены в результате первого посещения данного состояния и выполнения данного действия в каждом эпизоде.

Единственная трудность состоит в том, что количество важных непосещенных пар состояние-действие может оказаться велико. Если π – детерминированная стратегия, то следуя ей можно наблюдать награды для каждого из действий в каждом состоянии. Так как отсутствуют значения награды для усреднения, то оценки Монте-Карло для других действий не улучшатся с накоплением опыта. Это серьезная проблема, так как целью обучения является выбор действий, которые доступны в каждом состоянии. Чтобы сравнить альтернативы, необходимо вычислить значения всех действий в каждом состоянии, а не только наиболее предпочтительного в данный момент.

Данная задача является основной задачей обучения с подкреплением. Для оценки стратегии, чтобы выбрать подходящее действие, необходимо убедиться, что исследование осуществляется непрерывно. Достичь этого можно определив первый шаг в каждом эпизоде как пару состояние-действие, а также убедившись, что такая пара имеет ненулевую вероятность выбора в начале действия. Это гарантирует, что в пределе при числе эпизодов стремящихся к бесконечности, все пары состояние-действие будет посещены бесконечное число раз. Данное предположение называется предположением об изучающих стартах.

Предположение об изучающих стартах может быть полезным, но в общем случае

на него нельзя полагаться, особенно при обучении, основанном на непосредственном взаимодействии с окружающей средой. В данном случае начальные условия скорее всего не будут полезны. Наиболее распространенный альтернативный подход для того, чтобы убедиться в том, что все пары состояние-действие встречаются — это рассматривать только стохастические стратегии с ненулевой вероятностью выбора всех действий.

Общая идея формирования управления состоит в действиях по схеме, которая опирается на понятие обобщенной итерации по стратегиям (ОИС). В концепции ОИС используется как приближенная стратегия, так и приближенная функция ценности. Коррекция функции ценности происходит таким образом, чтобы наилучшим образом отвечать текущей стратегии, а стратегия постоянно уточняется в зависимости от текущей функции ценности.

МК-версия классической итерации по стратегиям состоит в попеременном выполнении полных шагов оценки стратегии и улучшении стратегии, начиная с произвольной стратегии π_0, и заканчивая оптимальной стратегией и оптимальной функцией ценности.

Улучшение стратегии происходит за счет создания жадной стратегии по отношению к функции ценности. Так как имеется функция ценности действия, то не требуется модели для того, чтобы построить жадную стратегию. Для любой функции оценки действия q, соответствующая ей жадная стратегия — это такая, что для каждого состояния $s \in S$, можно определенным образом выбрать действие с максимальной ценностью действия:

$$\pi(s) = \operatorname*{argmax}_a q(s, a).$$

Улучшение стратегии может быть осуществлено за счет построения каждой стратегии π_{k+1} как жадной стратегии по отношению к q_{π_k}. Методы Монте-Карло могут быть использованы при наличии лишь некоторого набора эпизодов, без использования знаний о модели и динамике окружающей среды.

Как правило в методах Монте-Карло чередуются операции оценивания и улучшения от эпизода к эпизоду. После завершения эпизода происходит оценивание стратегии, используя награду, которая была получена в эпизоде, после чего стратегия улучшается. Алгоритм, соответствующий данной схеме получил название ИС-алгоритм Монте-Карло с изучающими стартами.

В ИС-алгоритмах Монте-Карло все награды для каждой пары состояние-действие накапливаются и усредняются, независимо от того, какая стратегия имела место в момент получения данной награды.

Для того, чтобы избежать предположения об изучающих стартах, нужно обеспечить неограниченно частый выбор любого из возможных действий. Это можно сделать, заставив агента выбирать их. Для этой цели существует два подхода: с интегрированной

оценкой ценности стратегий и методам с разделенной оценкой ценности стратегий. Методы с интегрированной оценкой ценности стратегий (ИМК-методы) пытаются оценивать или улучшать стратегию непосредственно используя эту стратегию при принятии решений.

В ИМК-методах управления стратегия, как правило, является гибкой. Существует множество вариаций ИМК-методов. Основная идея ИМК-метода такая же как метода ОИС. Как и в ИС-методе здесь используются методы первого посещения для оценки функции действия текущей стратегии. Однако так как без предположения об изучающих стартах, для того чтобы улучшить стратегию, ее недостаточно сделать жадной по отношению к функции ценности, так как это будет являться препятствием для дальнейшего исследования нежадных действий. Однако так как ОИС-метод не требует постоянно использовать жадную стратегию, а требует постоянно двигаться по направлению к жадной стратегии, что и используется в данном случае. Данный метод использует ε-жадные стратегии. То есть большую часть времени данные стратегии выбирают действия с максимальными расчетными ценностями, а с вероятностью ε выбирают произвольные действия. Данные стратегии являются примером ε-гибких стратегий, которые определяются как стратегии, для которых выполняется неравенство

$$\pi(s,a) \geq \frac{\varepsilon}{|\mathcal{A}(s)|}.$$

В ИМК-методе осуществляется продвижение лишь к некоторой ε-жадной стратегии.

Во многих случаях возможно также получить функцию ценности для некоторой стратегии, располагая только опытными данными полученными "вне"данной стратегии. Однако, для того, чтобы использовать эпизоды стратегии π' для оценки стратегии π необходимо, чтобы каждое действие, которое осуществляется в стратегии π осуществлялось хотя бы иногда и в стратегии π'. Для этого в эпизодах, которые были получены при использовании стратегии π', для i-го первого посещения состояния s, и полной последовательности состояний и действий, которые следуют за этим посещением необходимо рассчитать вероятности $p_(s)$ и $p'_i(s)$. Это вероятности того, что данная полная последовательность будет иметь место при стратегиях π и π' при начальном состоянии s. $G_i(s)$ — выгода состояния s. Искомая оценка функции

$$V(s) = \frac{\sum_{i=1}^{n_s} \frac{p_i(s)}{p'_i(s)} G_i(s)}{\sum_{i=1}^{n_s} \frac{p_i(s)}{p'_i(s)}}.$$

В методах Монте-Карло с разделенной оценкой ценности стратегии (РМК-методы) выделяются стратегии поведения и стратегия оценивания. Стратегия, которая используется для формирования поведения называется стратегией поведения. Стратегия, которая оценивается и улучшается называется стратегией оценивания. Преимущество такого

разделения заключается в том, что стратегия оценивания может быть детерминированной, в то время как стратегия поведения может испытывать все возможные действия. Функция ценности в РМК-методах оценивается как функция оценивания одной стратегии, при использовании другой стратегии, как было описано ранее. Эта методика требует, чтобы стратегия поведения имела ненулевую вероятность выбора тех действий, которые могут быть использованы стратегией оценивания.

Обобщая вышесказанное, следует сказать, что методы Монте-Карло имеют 3 вида преимуществ по сравнению с методами динамического программирования. Во-первых, оптимальное поведение можно построить в результате непосредственного взаимодействия с окружающей средой, при этом не требуется знания модели динамики окружающей среды. Во-вторых, данные методы можно использовать в сочетании с методами моделирования, позволяющими получить смоделированные совокупности эпизодов. В-третьих, методы Монте-Карло могут легко и эффективно фокусироваться на небольшом подмножестве состояний.

3.4.3 Методы временных различий, Q−learning

Методы временных различий (TD-методы − Temporal Difference) совмещают в себе идеи методов Монте-Карло и динамического программирования. Как и методы Монте-Карло, TD-методы могут обучаться без модели динамики окружения, а непосредственно из опыта. Как и методы динамического программирования, TD-методы обновляют расчетные оценки, частично основываясь на других оценках, при этом не ожидая финального результата. [1]

И TD-методы, и методы Монте-Карло используют опыт, чтобы решить задачу предсказания. Из некоторого данного опыта следования стратегии π, оба метода обновляют их оценки V функции V_p для нетерминальных состояний S_t, которые присутствуют в данном опыте. В случае посещения нетерминального состояния s_t в момент времени t, оба метода корректируют свои оценки, основываясь на том, что случилось после посещения данного состояния. Метод всех посещений Монте-Карло, который подходит для всех нестационарных состояний :

$$V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)],$$

где G_t − фактическая выгода, полученная после момента времени t, α − постоянная длина шага. Данный метод называется МК-метод с постоянной α. Однако, в то время как методы Монте-Карло должны ждать окончания эпизода для определения приращения к $V(s_t)$, в случае с TD-методами необходимо дождаться следующего временного

шага. Простейший TD-метод, известный как TD(0), имеет следующий вид:

$$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{(t+1)} - V(S_t)].$$

Так как при коррекции TD-метод частично основывается на существующих оценках, то будем называть этот метод самонастраивающимся, как и метод ДП. TD-метод сочетает в себе выборки метода МК с самонастройкой методов ДП.

Одним из преимуществ TD-методов над методами ДП является то, что первые не требуют знания модели окружающей среды. Преимуществу TD-методов над методами МК является то, что первые являются интерактивными, полностью инкрементными методами. Если в МК-методах, необходимо дождаться завершения эпизода, чтобы узнать награду, то в TD-методах необходимо дождаться лишь следующего временного шага. Кроме того, МК-методы должны игнорировать эпизоды или снижать значимость эпизодов, в которых предпринимаются экспериментальные действия, что может сильно замедлить обучение. TD-методы не являются чувствительными к проблемам такого рода, так как обучаются на основе каждого перехода, вне зависимости от осуществляемых в дальнейшем действий. В настоящее время не доказано, что какой-либо из методов МК или TD сходится быстрее другого. Однако, на практике было показано, что TD-методы сходятся быстрее методов Монте-Карло с постоянной α.

TD-метод управления с интегрированной оценкой ценности стратегий состоит в следующем. Необходимо оценить функцию $q_\pi(s, a)$ для текущей стратегии π и для всех состояний s и действий a.

$$Q(s_t, a_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)]$$

Данная коррекция происходит после каждого перехода из нетерминального состояния s_t. Если состояние s_{t+1} является терминальным, то значение $Q(s_{t+1}, a_{t+1})$ полагается равным нулю. Это правило использует каждый элемент из пятерки событий $(s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1})$, из которых состоит переход от одной пары состояние-действие к другой. Это пятерка дала название SARSA данному алгоритму. Метод оценки ценности стратегии управления состоит в оценке функции q_π для стратегии π, в кто время как стратегия π делается более жадной по отношению к q_π.

Другой метод управления с разделенной оценкой ценности стратегий известен как Q-обучение. Его простейшая форма, одношаговое Q-обучение определяется следующим образом:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[R_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$$

В этом случае искомая функция ценности действия Q, непосредственно приближает q*, оптимальную функцию ценности действия, независимо от применяющейся стратегии.

Стратегия по-прежнему определяет какие пары состояние-действие корректируются и посещаются, однако для обеспечения сходимости необходимо лишь, чтобы все пары продолжали корректироваться.

4 Базовый алгоритм обучения с подкреплением

4.1 Q-сеть для задач обучения с подкреплениемі

Хорошие результаты на играх Atari показал алгоритм глубокого Q-обучения [3]. Разработанная система использовала сверточную нейронную сеть над набором последних кадров игры как аппроксиматор функции награды и для проведения шагов обновления алгоритма Q-обучения.

Дополнительным нововведением в этом подходе является техника повтора опыта, согласно которой совершенные агентом переходы, задающиеся четверкой $e_t = (s_t, a_t, r_t, s_{t+1})$, сохранялись в базу данных. Для обучения нейронной сети из полученной базы выделялась подвыборка, которая затем использовалась в шаге обновления Q-обучения. Алгоритм сохраняет фиксированное число последних переходов. Подобный подход к обучению что увеличивает скорость сходимости и стабильность обучения.

4.2 Двойная Q-сеть.

Одним из недостатков стандартного алгоритма Q-обучения является задокументированая и теоретически обоснованная тенденция к значительному завышению оценок функции награды. Причиной подобного поведения является то, что Q-обучение, при наличии нескольких альтернатив действий, выбирает действие с наибольшей оценкой его функции награды, что приводит при наличие неизбежного из-за стохастической природы алгоритма шума, к тому, что алгоритм будет стремиться выбирать действия с наибольшим размером завышения оценки. Подобное систематическое завышение приводит к изменению оптимальной стратегии поведения агента, к которой в асимптотике сходится алгоритм.

Для решения данной проблемы было предложено использовать две нейронные сети параллельно [4] . Одна из них обучается обычным алгоритмом Q-обучения на выбранном обучающем наборе, вторая же используется для оценки функции награды от выбранного действия, и обновляется значением первой нейронной сети каждые несколько десятков шагов. Такое решение значительно уменьшило размер переоценки функции награды и привело к улучшению достигнутого результата на тестовом наборе окружений.

4.3 Дуэлирующая двойная Q-сеть.

Основная идея, лежащая в основе дуэлирующей двойной Q-сети [5] , заключается в том, что для большинства состояний в большинстве исследуемых окружений функция награды действия в состоянии $Q(s, a)$ можно разделить на функция выгоды действия и

функция награды состояния

$$Q_\pi(s,a) = V_\pi(s) + A_\pi(s,a)$$

$$E_{a\in\pi}A(s,a) = 0$$

каждая из которых обладает своим интерпретируемым смыслом и изменяется по-разному в зависимости от обрабатываемых опытов. В связи с этим разделение вычисления функции награды на две вышеописанные части приводит к более точному изменению оцениваемой функции награды при использовании тех же обучающих примеров.

Другим важным усовершенствованием, использованным в архитектуре с дуэлирующей двойной Q-сетью, является приоритезированный повтор опыта [6] , который вместо полностью случайного выбора обучающей выборки из накопленного банка опыта пытается выбрать те моменты, которые наиболее важны для обучения агента. Центральным понятием такого подхода является критерий, оценивающий важность для обучения каждого из совершенных агентом переходов. Одна из самых простых эвристик, позволяющая оценить подобную характеристику - это ошибка в оценке функции награды итогового состояния до и после совершения действия. Вероятность выбора каждого перехода в набор обучения равна

$$P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha}$$

где $p_i > 0$ - оценка важности перехода i. Параметр α определяет насколько сильно используется приоритизация, и при α равном нулю система выбирает различные переходы с равной вероятностью. p_i возможно вычислять пропорциональной приоритизацией, где $p_i = |\delta_i| + \epsilon$ или приоритизацией, основанной на ранге, где $p_i = \frac{1}{rank(i)}$ - обратное число к индексу опыта под номером i после сортировки этих опытов в соответствии с ошибкой δ_i.

Введение приоритезированного использования опыта изменяет математическое ожидание оцениваемых величин функции награды из-за изменения вероятностей использования различных данных. Для исправления этой ошибки оценивания необходимо внести дополнительные штрафующие веса на обновляющее правило в Q-обучении :

$$w_i = \Big(\frac{1}{N} * \frac{1}{P(i)}\Big)^\beta$$

где параметр β показывает, насколько сильно необходима поправка на веса в данный момент обучения. Так как точность схождения важна в большей степени в финальных этапах обучения, уменьшение этого параметра на первых этапах обучения позволяет алгоритму сильнее обучаться на начальном опыте.

5 Программная структура системы обучения с подкреплением

В рамках проделанной работы была реализована система обучения с подкреплением[1] над окружениями из фреймворка OpenAI Gym[7] на языке Python2.7 с использованием библиотек Theano[8] и Lasagne[9]. Данная система реализует метод дуэлирующей двойной Q−сети с стандартным повтором опыта, и позволяет интегрировать любое окружение из фреймворка OpenAI Gym.

6 Стратегии исследования окружений

Базовые стратегии исследования окружений включают в себя ϵ-жадную стратегию и стратегию исследования по Больцману [10] .

6.1 $\mathcal{E}$-жадная стратегия исследования

$\mathcal{E}$-жадная стратегия исследования совершает на каждом шаге алгоритма случайное действие с вероятностью ϵ, и совершает оптимальное действие согласно с текущим значением функции награды с вероятностью $1 - \epsilon$. Типичная стратегия обучения состоит в инициализации значения параметра ϵ большим значением и постепенным понижением этого параметра до маленького значения по мере обучения для уменьшения доли случайного поведения.

$$\pi_\epsilon(s) = \begin{cases} random, & \text{с вероятностью } p = \epsilon \\ \text{argmax}_a Q(s,a), & \text{с вероятностью } p = 1 - \epsilon \end{cases}$$

6.2 Стратегия исследования по Больцману

Стратегия исследования по Больцману выбирает случайное действие с весом, пропорциональным текущему значению функции награды для этого действия. Такая стратегия позволяет приоритизировать действия, которые потенциально приведут к большей награде в будущем.

$$P_b(\pi_\epsilon(s) = a^*) = \frac{e^{\frac{Q(s,a^*)}{T}}}{\sum_a e^{\frac{Q(s,a)}{T}}}$$

T в формуле исследования по Больцману - это параметр температуры, который задает силу сглаживания распределения вероятностей между действиями. При стремлении этого параметра к бесконечности алгоритм начинает выбирать случайные действия, при

[1] `https://github.com/Goorman/Reinforcement_Learning_Dissertation`

движении параметра температуры к нулю алгоритм начинает выбирать оптимальное действие согласно текущей функции награды

$$\begin{cases} P_b(\pi_\epsilon(s) = a^*) = \frac{1}{|A|}, & \text{при } T \to \infty, \\ \pi_\epsilon(s) = \text{argmax}_a Q(s,a), & \text{при } T \to 0. \end{cases}$$

6.3 Приоритезированное исследование на основе модели, предсказывающей динамику окружения

Одним из подходов исследования окружений является изменение полученной агентом награды для учета пользы, которая получена из-за проведенного исследования окружения. Один из таких подходов использует кодированное представление состояния s - $\sigma(s)$ [11] . Составив модель предсказывания кодированного представления нового состояния по кодированному представлению старого состояния и совершенному действию $M : \sigma(S) \times A \to \sigma(S)$, можно получить способ вычисления ошибки этой модели.

$$e(s_t, a_t) = ||\sigma(s_{t+1} - M(\sigma(s_t), a_t))||$$

$$\tilde{e}_T = \frac{e_T}{\max_{t<T} e_t}$$

Эту ошибку можно использовать для изменения награды, полученной после совершения действия, подавая агенту дополнительный стимул совершать те действия, результат которых модель M предсказывает плохо.

$$R_{pred}(s,a) = R(s,a) + \beta * \left(\frac{\tilde{e}_t}{t^C}\right)$$

Способ кодирования $\sigma(s)$, использующий значения слоев нейронной сети, оказывается неэффективным. Для уменьшения размерности пространства состояний, можно применять автокодировщики - нейронные сети, уменьшающие размерность входа и оптимизирующие ошибку при возврате исходного значение по сжатому представлению.

В данной работе предложена модификация данного алгоритма, заключающаяся в том, что автокодировщик делит свою первую половину слоев нейронной сети с Q-сетью. Подобный подход позволит внести в первые слои Q-сети дополнительную функцию извлечения закодированного представления состояния, что должно улучшить способность правильно предсказать функцию ценности. Модель предсказывания реализована как нейронная сеть от конкатенированных значений центрального слоя автокодировщика и совершенного действия в one-hot кодировке.

6.4 Приоритезированное исследование на основе количества посещений состояния

Другим способом поощрения исследования является способ, добавляющий награду на основе того, насколько часто агент бывал в текущем состоянии.[12] Для того, чтобы поддерживать счет количеству посещений в пространстве состояний, которое может быть непрерывно, к центральному слою добавляется функция активации сигмоида: $f_s(x) = \frac{1}{1+e^{-x}}$, выдающая значения от 0 до 1, к оптимизируемой функции ошибки добавляется регуляризирующий член на центральный слой автокодировщика: $Loss(h) = 0.25 - (0.5 - h)^2$, а после центрального слоя добавляется слой гауссового шума с фиксированной дисперсией. Подобные изменения приводят к тому, что центральный слой автокодировщика выдает значения, близкие либо к 0, либо к 1, и при этом функция потерь нейронной сети не потеряла свойство дифференцируемости. Это позволяет получить для каждого состояния его дискретное представление $\phi(s)$ через округление вектора значений центрального слоя автокодировщика до ближайшего целого. На каждом этапе обучения алгоритм поддерживает количество посещений каждого значения закодированного представления состояний $n(\phi(s))$. На основе этой статистики вычисляется дополнительная награда:

$$R_{count}(s,a) = R(s,a) + \beta * \left(\frac{1}{(n(\phi(s)))^C}\right)$$

В данной работе предложена модификация данного алгоритма, заключающаяся в том, что автокодировщик делит свою первую половину слоев нейронной сети с Q-сетью.

6.5 Байесовское Q-обучение

Один из возможных подходов к исследованию заключается в выражении ожидаемой награды $Q(s,a)$ в состоянии s и действии a через нормальное распределение с гиперпараметрами $\nu_{s,a}, \tau_{s,a}$, которые задают среднее значения и точность распределения [13]. Гиперпараметры этого выражения выражаются через априорное гамма-нормальное распределение - сопряженное распределение к нормальному распределению.

$$p(\nu,\tau) \sim NG(\nu_0, \lambda, \alpha, \beta)$$

$$p(\nu,\tau) \propto \tau^{\frac{1}{2}} e^{-\frac{1}{2}\lambda\tau(\nu-\nu_0)^2} \tau^{\alpha-1} e^{\beta\tau}$$

Из таких предположений будет следовать, что апостериорное распределение гиперпараметров нормального распределения после учета данных о награде будет так же принадлежать гамма-нормальному распределению. Байесовское Q-обучение заключается в приближенном подсчете гиперпараметров этого распределения в каждой паре

состояния-действия вместо приближенного расчета значения функции награды. При получении нового элемента выборки M правила обновления гиперпараметров задаются следующим образом:

$$\nu_0^* = \nu_0 + \frac{1}{\lambda + 1} * (M - \nu_0)$$

$$\beta^* = \beta + \frac{h}{2(h+1)} * (M - \nu_0)^2$$

$$\lambda^* = \lambda + 1$$

$$\alpha^* = \alpha + 0.5$$

Причем математическое ожидание и дисперсия $Q(s, a)$ могут быть вычислены на основе гиперпараметров следующим образом:

$$\mathrm{E}[Q(s, a)] = \nu_0$$

$$\mathrm{Var}[Q(s, a)] = \frac{b}{\lambda(\alpha - 1)}$$

Исследование окружения, которое подразумевает под собой выбор действия на основе текущего состояния, при таком подходе можно проводить несколькими способами.

- Жадный выбор

При подходе жадного выбора алгоритм выбирает такое действие, которое максимизирует значения математического ожидания распределения. Такой агент не будет исследовать окружения и будет пытаться выбрать оптимальное действие на каждом шаге.

- E-жадное исследование на основе вычисленной дисперсии

При таком подходе алгоритм с вероятностью ϵ принимает действие, максимизирующее функцию награды, а с вероятностью $1 - \epsilon$ принимает действие, $\mathrm{Var}[Q(s, a)]$ которого максимальна.

- Сэмплинг Q-функции

В подходе сэмплинга Q-функции действие a в состоянии s выбирается с вероятностью, равной вероятности того, что случайная величина математического ожидания функции награды $\nu_{s,a}$ будет максимальной среди всех доступных действий.

$$P(a = \mathrm{argmax}_{a*}\nu_{s,a*} = P(\forall \neq a, \nu_{s,a} > \nu_{s,a*} =$$

$$\int_{-\infty}^{+\infty} P(\nu_{s,a} = q_a) \prod_{a* \neq a} P(\nu_{s,a*} < q_a) dq_a$$

На практике точное вычисление этих вероятностей по вышеприведенным формулам можно заменить семплированием каждой случайной величины с распределением для каждого действия и выбором того действия, сэмпл которого оказался максимальным. Такая процедура выдает действия с такой же вероятностью, которую выдал бы прямолинейный подсчет вероятности того, что ожидание функции награды действия будет максимально.

- Близорукий выбор на основе ценности информации

Подход близорукого выбора заключается в присваивании полученной информации о работе системы ценности, получение которой поручается агенту в дополнение получению стандартной награды. Полученная информация может быть полезна в случае, когда полученный опыт превратил действие, прежде считавшиеся субоптимальным, в оптимальное, или в случае когда ранее считающееся оптимальным действие превратилось в субоптимальное. В первом случае, если a_1 - лучшее действие на данном этапе, то есть $E[\nu_{s,a_1} \geq E[\nu_{s,a_1}]$, и новый опыт показывает, что $\nu^*_{s,a} > E[\nu_{s,a_1}]$, то агент получает $E[u_{s,a_2}] - \nu^*_{s,a_1}$. Во втором же случае, если a_1 и a_2 - соответственно первое и второе лучшие действия, и новый опыт показывает, что $u_{s,a_1} < E[\nu_{s,a_2}]$, то агент, совершая действие a_2 вместо a_1, будет получать $E[\nu_{s,a_2}] - \nu^*_{s,a_1}$. Полную функцию выгоды от нового значения параметр $\nu^*_{s,a}$ можно выразить как

$$Gain_{s,a}(\nu^*_{s,a}) = \begin{cases} E[\nu_{s,a_2}] - \nu^*_{s,a} & \text{если } a = a_1, \nu^*_{s,a} < E[\nu_{s,a_2} \\ \nu^*_{s,a} - E[\nu_{s,a_1}] & \text{если } if a \neq a_1, \nu^*_{s,a} > E[\nu_{s,a_1} \\ 0 & otherwise \end{cases}$$

Полную ожидаемую выгоду можно выразить как

$$VPI(s,a) = \int_{-\infty}^{\infty} \mathrm{Gain}_{s,a}(x) P(\nu_{s,a} = x) dx$$

Ценность, полученная этим исследования получена ценой уменьшения ожидаемой награды предпринятого действия. Это значит, что действие можно выбирать с целью максимизации $VPI(s,a) - (max_{a^*} E[Q(s,a^*)] - E[Q(s,a)])$, что аналогично выбору действия, максимизирующего $E[Q(s,a)] + VPI(s,a)$

В данной работе предлагается избежать вычисления гиперпараметров гамма-нормального распределения λ и α с помощью нейронной сети. Это необходимо в связи с тем, что нейронная сеть не гарантирует того, что в процессе обучения эти параметры будут для всех состояний окружений иметь адекватные значения, а также из-за того, что постоянное увеличение параметров λ и α приведет к преждевременной сходимости процесса обучения. λ и α считаются гиперпараметрами алгоритма исследования и фиксируются на

все время обучения алгоритма. В таких условиях нейронная сеть должна поддерживать только гиперпараметры ν_0 и β, на основании которых необходимо было бы вычислять математическое ожидание и дисперсию ожидаемой награды. В данной работе предлагается вычислять математическое ожидание и дисперсию ожидаемой награды напрямую, используя следующие правила обновления:

$$E(Q(s,a))_{new} = E(Q(s,a)) + \frac{1}{\lambda+1} * (M - E(Q(s,a)))$$

$$Var(Q(s,a))_{new} = \frac{\lambda*(\alpha-1)}{(\lambda+1)*(\alpha-0.5)} * \left(Var(Q(s,a)) + \frac{(M-E(Q(s,a)))^2}{2(\lambda+1)(\alpha-1)}\right)$$

Для выбора действия используется метод E-жадного исследования на основе вычисленной дисперсии. Во всех рассматриваемых окружениях в качестве параметров λ и α взяты значения 4 и 2 соответственно.

6.6 Комбинированный метод на основе разделения оценки награды за эксплуатацию и исследование

Большинство рассмотренных методов исследования окружений использует изменение получаемых агентом наград как способ поощрить за посещение ценных с точки зрения исследования состояний. Побочным эффектом такого подхода является тот факт, что вычисляемая алгоритмом Q-learning функция ожидаемой награды в процессе обучения перестает представлять приближение оптимальной функции награды. Такое изменение процесса обучения может негативно сказаться на скорости процесса обучения. Для преодоления этой проблемы предлагается разделить вычисление $Q(s,a)$ как непосредственно функции награды окружения и функции $E(s,a)$, которая выражает ожидаемую функцию награды за исследование при совершении в состоянии s действия a. В такой формулировке стратегия исследования окружения будет задаваться способом вычисления $E(s,a)$ и функцией $M(\overline{Q(s,a),\forall a}, \overline{E(s,a),\forall a}, \delta) \to a_{best}$, которая на основе параметра $\delta \in (0,1)$ и значений функций $Q(s,a)$ и $E(s,a)$ во всех доступных действиях для состояния s вычислит действие, которое необходимо предпринять в данном состоянии. Все рассмотренные в данной работе способы исследования окружений представляются в такой форме:

- $\mathcal{E}$-жадная стратегия исследования

$$E(s,a) = 1$$

$$M(Q,E,\delta) \sim \begin{cases} random, & \text{с вероятностью } p = \delta \\ \text{argmax}_a Q_a, & \text{с вероятностью } p = 1-\delta \end{cases}$$

- Стратегия исследования по Больцману

$$E(s,a) = 1$$

$$M(Q,E,\delta) \sim P(M(Q,E,\delta) = a) = \frac{e^{\frac{Q_a}{\ln(\frac{1}{\delta})}}}{\sum_a e^{\frac{Q_a}{\ln(\frac{1}{\delta})}}}$$

- Приоритезированное исследование на основе модели, предсказывающей динамику окружения

$E(s,a)$ вычисляется с помощью правила обновления Беллмана над дополнительной наградой метода $R_{pred}(s,a) - R(s,a)$, после чего проводится либо полностью жадная стратегия поведений агента, либо одна из базовых стратегий над функцией награды $Q^*(s,a) = Q(s,a) + E(s,a)$

- Приоритезированное исследование на основе количества посещений состояния

$E(s,a)$ вычисляется с помощью правила обновления Беллмана над дополнительной наградой метода $R_{count}(s,a) - R(s,a)$, после чего проводится либо полностью жадная стратегия поведений агента, либо одна из базовых стратегий над функцией награды $Q^*(s,a) = Q(s,a) + E(s,a)$

- Байесовское Q-обучение

$E(s,a)$ вычисляется с помощью правила обновления Беллмана над дополнительной наградой метода $R_{bayes}(s,a) - R(s,a)$, после чего проводится либо полностью жадная стратегия поведений агента, либо одна из базовых стратегий над функцией награды $Q^*(s,a) = Q(s,a) + E(s,a)$

В системе обучения с подкреплением реализован метод, комбинирующий Байесовское Q-обучение и приоритизацию исследования на основе количества посещений, $E(s,a)$ вычисляется с помощью правила обновления Беллмана над дополнительной наградой метода $R_{bayes}(s,a) - R(s,a) + R_{count}(s,a) - R(s,a)$.

7 Эсперимантальное исследование

7.1 Рассматриваемые окружения

- Cart Pole

Окружение CartPole (Рис. 1), цель игры состоит в поддержании вертикального положения шеста на тележке. Пространство состояний состоит из 4 вещественных переменных, описывающих положение тележки и столба. Окружение предоставляет два действия, позволяющие толкнуть тележку направо или налево. Каждый кадр игры дает положительную награду, если столб отклоняется от вертикального положения больше чем на заданное значение, то эпизод заканчивается с большой отрицательной наградой. Эпизод заканчивается либо когда шест на тележке отклоняется от вертикального положения больше, чем на 15 градусов, либо после прошествия 500 кадров.

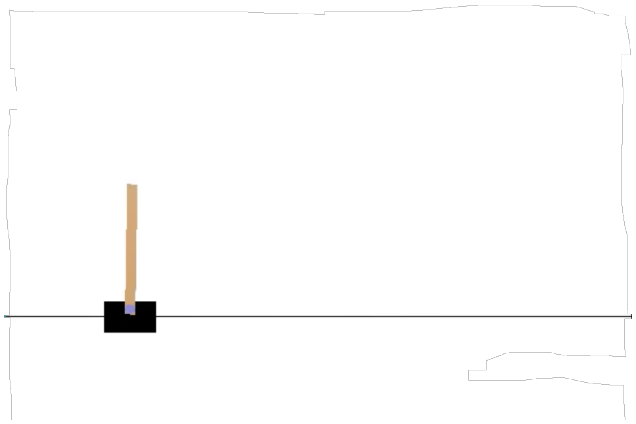

Рис. 1: Кадр из игры CartPole.

- Lunar Landing

Окружение LunarLanding (Рис. 2), цель игры состоит в посадке лунного модуля на выделенную площадку с минимальными повреждениями и затратами топлива. Пространство состояний состоит из 8 вещественных переменных, задающих физическую конфигурацию текущего состояния системы, доступны четыре действий, позволяющие включить двигатели в одно из трех направлений, либо не делать ничего. Работа двигателей тратит топливо, что дает отрицательную награду. Каждый кадр игры дает награду, обратно пропорциональную расстоянию до посадочной площадки по горизонтали. Если при посадке на поверхность модуль не имел достаточно маленькой скорости, то агент получает большую отрицательную награду. Игра заканчивается либо при аварийной

посадке модуля (приземление с достаточно большой скоростью), либо при удачной посадке модуля (приземление с достаточно маленькой скоростью), либо после прошествия 200 кадров.

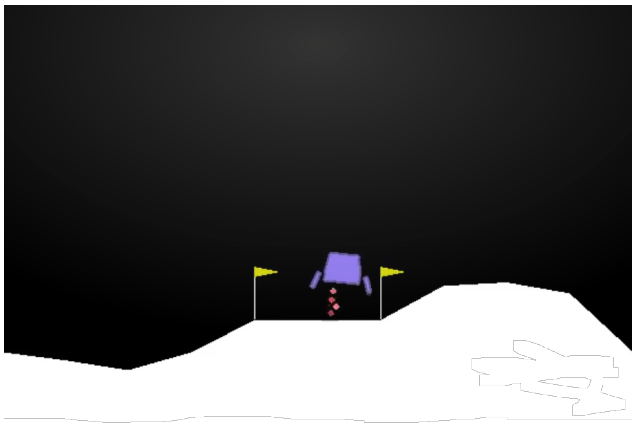

Рис. 2: Кадр из игры LunarLanding.

7.2 Методика проведения исследований

- Построение кривой обучения алгоритмов

Для исследования реализованных алгоритмов каждый из реализованных алгоритмов применяется на наборе окружений, обучение прекращается после достижения I_{maxep} эпизодов. Каждому эпизоду обучения под номером $i \in [1, I_{maxep}]$ сопоставляется полученная в этом эпизоде награда $R(i)$. Функция $R(i)$ отображенная на графике задает кривую обучения алгоритма на заданном окружении. Эта кривая будет использоваться для анализа построенных алгоритмов.

- Усреднение результатов по нескольким экспериментам и сглаживание кривых обучения

Каждая конкретная кривая обучения подвержена влиянию случайных эффектов. Эти случайные эффекты включают в себя выбор начальных весов нейронной сети, случайные действия алгоритмов исследований окружений, выбор уровня в каждом конкретном эпизоде окружения. Наличие этих случайных факторов означает, что одна кривая обучения алгоритма на окружении не даст полезной информации о свойствах алгоритма. В этой работе каждый алгоритм исследования запускается $N_{exp} = 10$ раз, после чего берется среднее от всех полученных кривых обучения, образуя $R_{mean}(i) = \frac{\sum_{k=1}^{N_{exp}} R_k(i)}{N_{exp}}$. Далее кривая обучения сглаживается методом скользящего среднего с размером окна windowsize: $R_{alg}(i) = \frac{\sum_{k=1}^{\text{windowsize}} R_{mean}(i+1-k)}{\text{windowsize}}$, $i \in [\text{windowsize}, I_{maxep}]$. Полученная функция $R_{alg}(i)$ используется для анализа алгоритмов исследования.

- Метрики качества

В данной работе для каждого алгоритма на каждом окружении вычисляются две метрики качества: MaxR(1000), где

$$\text{MaxR}(K) = \max_{i \leq K} R_{alg}(i)$$

- показывает максимально достигнутую алгоритмом награду в окружении до заданного количества эпизодов в обучении, и MeanR(1000), где

$$\text{MeanR}(K) = \text{mean}_{i \leq K} R_{alg}(i)$$

- среднее значения награды до заданного количества эпизодов в обучении, показывающее нормированную площадь под кривой обучения и являющуюся аналогом скорости обучения алгоритма.

7.3 Результаты исследований

Обозначим реализованные методы следующим образом:

- greedy - $\mathcal{E}$-жадная стратегия исследования.
- boltzman - Стратегия исследования по Больцману.
- count - Приоритезированное исследование на основе модели, предсказывающей динамику окружения.
- inc - Приоритезированное исследование на основе количества посещений состояния.
- bayes - Байесовское Q-обучение.
- comb - Комбинированный метод на основе разделения оценки награды за эксплуатацию и исследование.

По результатам экспериментов над окружением CartPole, показанных на Рис. 3, Рис. 4, Таблице 1, можно увидеть, что максимальной награды за 1000 эпизодов добился комбинированный алгоритм, а наилучшей скорости обучения, представляемой средней наградой за 1000 эпизодов, добился алгоритм Байеса. Все алгоритмы показали в целом хорошие результаты в решении поставленной задачи обучения с подкреплением.

По результатам экспериментов над окружением LunarLanding, показанных на Рис. 5, Рис. 6, Таблице 2, можно увидеть, что максимальной награды за 1000 эпизодов добился

алгоритм, основанный на приоритизации исследования на основе количества посещений состояний, а наилучшей скорости обучения добился алгоритм, основанный на приоритизации исследования с помощью предсказывающей модели. Остальные алгоритмы показали плохие результаты на этом окружении. Причина этой неудачи заключается скорее всего в структуре выдачи награды в окружении LunarLanding, которая создает локальный оптимум в пространстве стратегий действия агента - так как награда, поощряющая близость лунного модуля к центру посадочной площадки по горизонтали выдается каждый шаг, а награда за правильную посадку дается только в конце сложной последовательности действий, агенты вместо плавного балансирования своего положения при спуске предпочитают резкий переворот. Алгоритмы «count» и «inc» основаны на изменении исходных наград, поступающих от окружения агенту, и эта особенность алгоритма позволяет избежать локального оптимума в пространстве стратегий.

На основании проведенных экспериментов можно сделать вывод, что невозможно выделить алгоритм исследования окружений, проявляющий себя лучше всех остальных на всех окружениях. В связи с этим возникает потребность исследования более широкого спектра окружений с целью выделения подмножеств окружений, для каждого из которых будет определен алгоритм исследования, проявляющий себя наилучшим образом. В данной работе такое исследование было невозможно произвести из-за нехватки вычислительных ресурсов.

Стоит отметить, что новые методы, которые были предложены в данной работе, показывают себя как наилучшие на одном из окружений, из чего следует сделать вывод, что существует ряд окружений на которых эти методы являются наилучшими. В будущем следует выделить характеристики таких окружений.

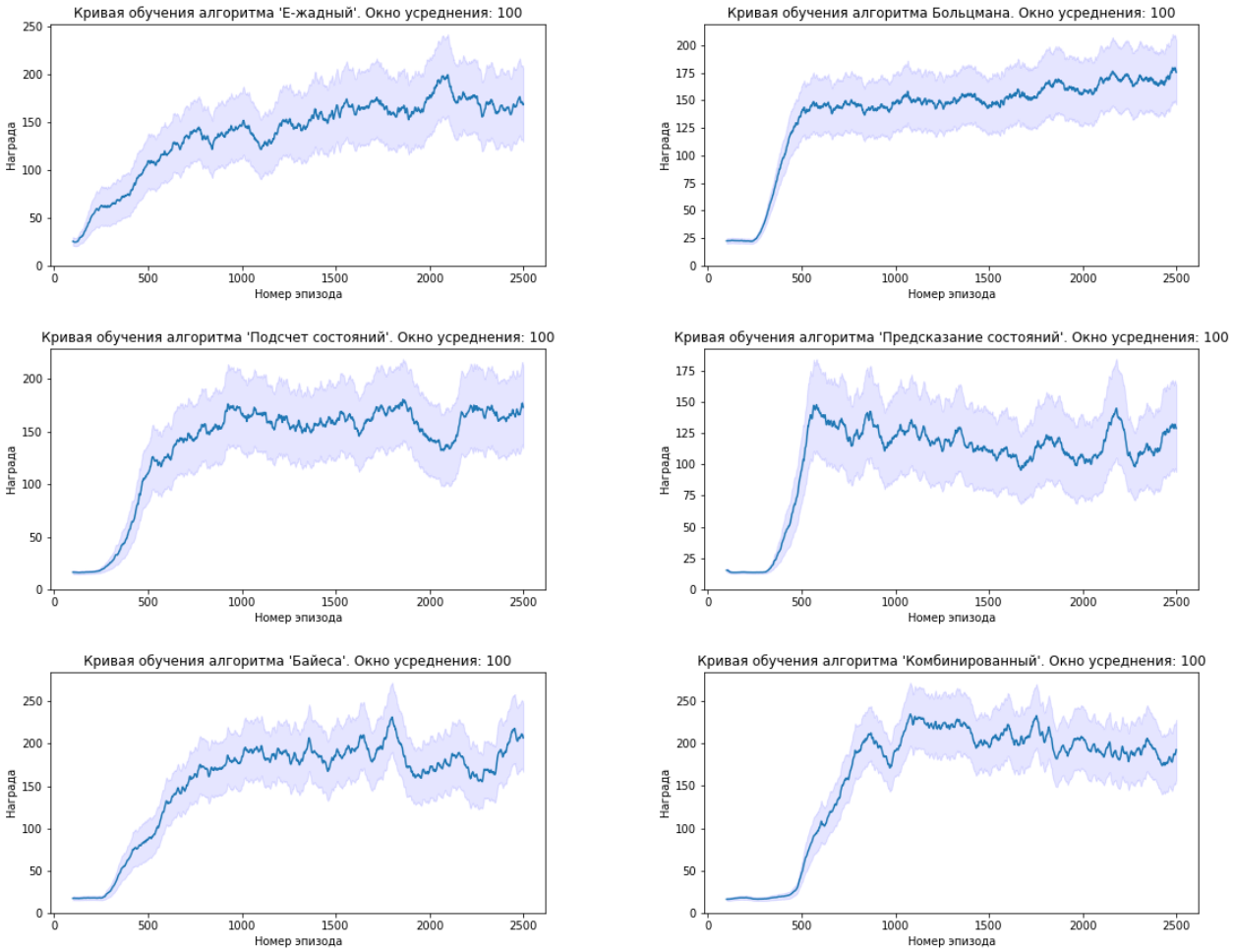

Рис. 3: Усредненные и сглаженные кривые обучения алгоритмов на окружении CartPole.

Алгоритм исследования	MaxR(1000)	MeanR(1000)
greedy	151.81	112.00
boltzman	158.05	120.36
count	176.10	114.97
inc	147.56	96.40
bayes	196.45	**121.15**
comb	**234.59**	113.66

Таблица 1: Результаты экспериментальных исследований на окружении CartPole

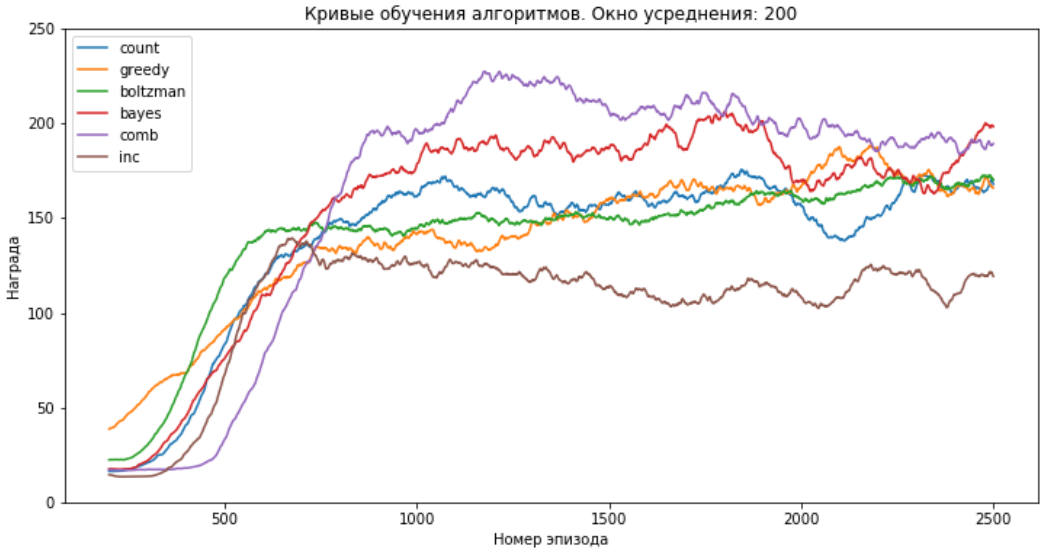

Рис. 4: Усредненные и сглаженные кривые обучения алгоритмов на окружении CartPole.

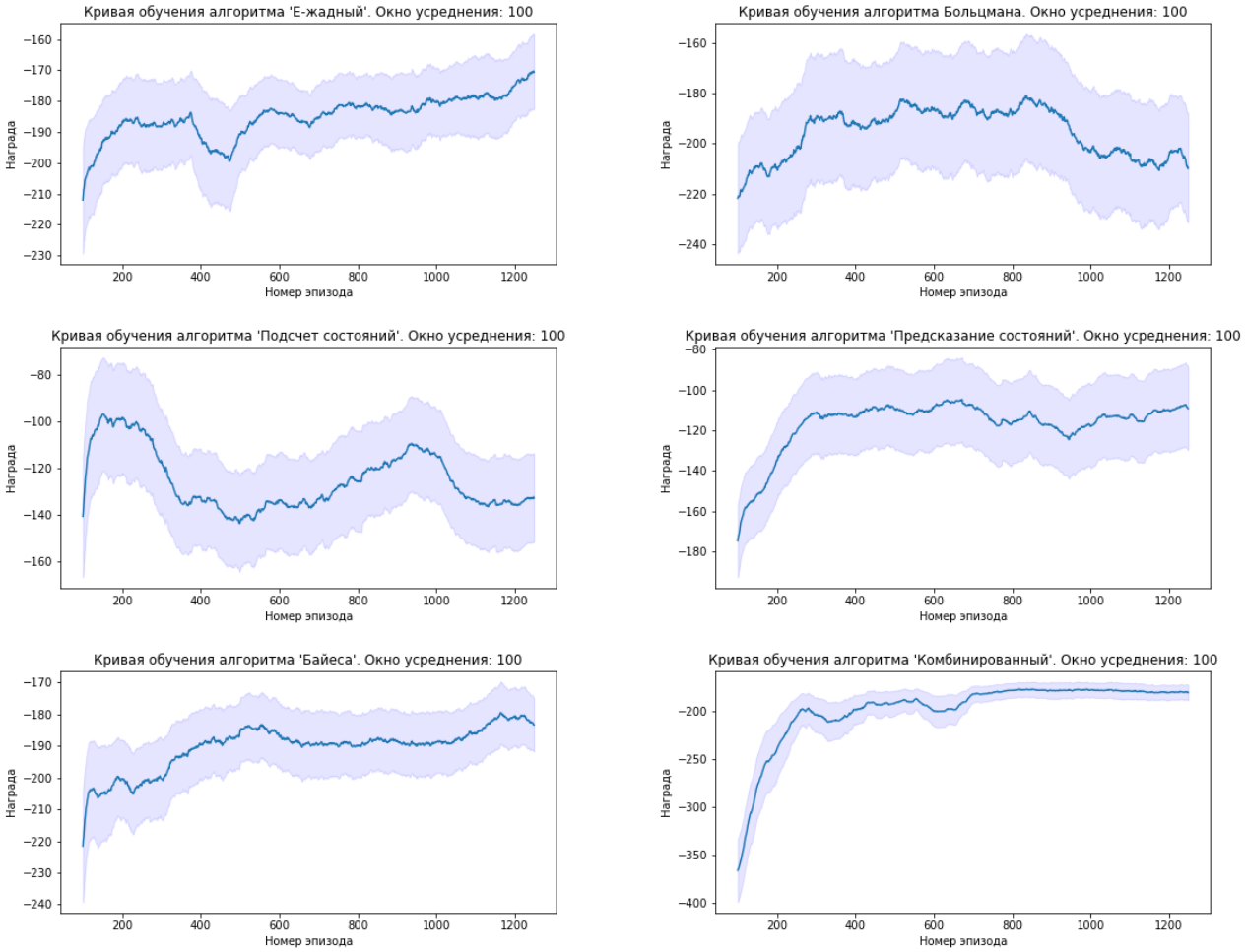

Рис. 5: Усредненные и сглаженные кривые обучения алгоритмов на окружении LunarLanding.

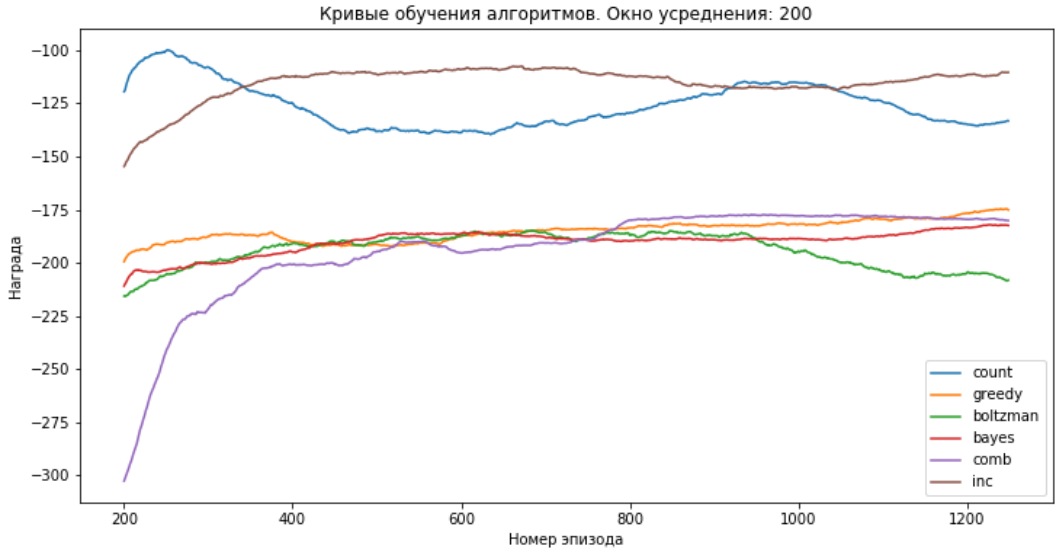

Рис. 6: Усредненные и сглаженные кривые обучения алгоритмов на окружении LunarLanding.

Алгоритм исследования	MaxR(1000)	MeanR(1000)
greedy	-178.06	-185.32
boltzman	-181.08	-191.77
count	**-98.18**	-126.07
inc	-104.72	**-113.66**
bayes	-183.17	-190.27
comb	-176.69	-189.99

Таблица 2: Результаты экспериментальных исследований на окружении LunarLanding.

8 Заключение

В ходе данной работы было проделано следующее:

- Реализована система обучения с подкреплением на основе фреймворка OpenAI Gym.
- Предложены модификации к существующим методам(«count», «inc.», «bayes») и новый метод исследования окружений («comb»)
- Реализованы 6 методов исследования окружений в составе реализованной системы обучения с подкреплением.
- Проведено экспериментальное исследование реализованных методов на двух окружениях OpenAI Gym.

На основе результатов проделанной работы были сделаны следующие выводы:

- Среди реализованных алгоритмов исследования окружений нельзя выделить алгоритм, наилучшим образом проявляющий себя на всех окружениях.
- Предложенный метод позволяет получить наилучшее качество по одной из метрик на одном из рассматриваемых окружений.
- По другой метрике реализованная модификация одного из существующих методов получила наилучшее качество.
- Следует провести анализ характеристик окружений, на которых предложенный алгоритм исследования и реализованные модификации к существующим методам показывают наилучшее качество по введенным метрикам.

Список литературы

[1] Sutton R. S., Barto A. G. Reinforcement learning: An introduction. – Cambridge : MIT press, 1998. – Т. 1. – №. 1.

[2] Wiering M., Van Otterlo M. Reinforcement learning //Adaptation, Learning, and Optimization. – 2012. – Т. 12.

[3] Mnih V. et al. Playing atari with deep reinforcement learning //arXiv preprint arXiv:1312.5602. – 2013.

[4] Van Hasselt H., Guez A., Silver D. Deep reinforcement learning with double Q-learning //CoRR, abs/1509.06461. – 2015.

[5] Wang Z., de Freitas N., Lanctot M. Dueling network architectures for deep reinforcement learning //arXiv preprint arXiv:1511.06581. – 2015.

[6] Schaul T. et al. Prioritized experience replay //arXiv preprint arXiv:1511.05952. – 2015.

[7] Brockman G. et al. OpenAI gym //arXiv preprint arXiv:1606.01540. – 2016.

[8] Team T. T. D. et al. Theano: A Python framework for fast computation of mathematical expressions //arXiv preprint arXiv:1605.02688. – 2016.

[9] Dieleman S. et al. Lasagne: First release //Zenodo: Geneva, Switzerland. – 2015.

[10] McFarlane R. A Survey of Exploration Strategies in Reinforcement Learning.

[11] Stadie B. C., Levine S., Abbeel P. Incentivizing exploration in reinforcement learning with deep predictive models //arXiv preprint arXiv:1507.00814. – 2015.

[12] Tang H. et al. # Exploration: A Study of Count-Based Exploration for Deep Reinforcement Learning //arXiv preprint arXiv:1611.04717. – 2016.

[13] Dearden R., Friedman N., Russell S. Bayesian Q-learning //AAAI/IAAI. – 1998. – С. 761-768.

[14] Nouri A., Littman M. L. Multi-resolution exploration in continuous spaces //Advances in neural information processing systems. – 2009. – С. 1209-1216.