Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

“САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ”

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

РАЗРАБОТКА ИГРОВЫХ СТРАТЕГИЙ С ПРИМЕНЕНИЕМ МЕТОДОВ МАШИННОГО ОБУЧЕНИЯ НА ПРИМЕРЕ HEARTHSTONE

Автор _____ *Качальский И.В.*

Направление подготовки (специальность)
01.03.02 Прикладная математика и информатика

Квалификация *бакалавр*

Руководитель _____ *кандидат технических наук Ульянцев В.И.*

К защите допустить

Зав. кафедрой _____ *Васильев В.Н.*

Санкт-Петербург, 2017 г.

Студент *Качальский И.В.* Группа *М3436* Кафедра *КТ* Факультет *ИТиП*

Направленность (профиль), специализация

Математические модели и алгоритмы в разработке программного обеспечения

Квалификационная работа выполнена с оценкой _____

Дата защиты «____» июня 2017 г.

Секретарь ГЭК *Павлова О.Н.* принято «____» июня 2017 г.

Листов хранения _____

Демонстрационных материалов/Чертежей хранения _____

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

АННОТАЦИЯ
ПО ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

Студента Качальский И. В.
(Фамилия, И., О.)

Факультет Информационных технологий и программирования

Кафедра Компьютерных технологий **Группа** М3436

Направление (специальность) Прикладная математика и информатика

Квалификация (степень) Бакалавр прикладной математики и информатики

Наименование темы: Разработка игровых стратегий с применением методов машинного обучения на примере Hearthstone

Руководитель Ульянцев В. И., к т.н., доцент кафедры КТ, университет ИТМО.
(Фамилия, И., О., ученое звание, степень)

КРАТКОЕ СОДЕРЖАНИЕ ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ И ОСНОВНЫЕ ВЫВОДЫ

объем 41 стр., **графический материал** 0 стр., **библиография** 18 наим.

- Направление и задача исследований

Задачей исследование является разработка модели, способной улучшить результат предыдущих работ, а также расширить область, в которой агент способен оперировать путём применения различных техник машинного обучения.

- Проектная или исследовательская часть (с указанием основных методов исследований, расчетов и результатов)

В рамках проделанной работы была простроена модель, позволяющая играть в Hearthstone.
Разработана архитектура, показывающая применение данной модели к задаче игры в Hearthstone.

- Экономическая часть (какие использованы методики, экономическая эффективность результатов)

Не рассматривалась.

- Характеристика вопросов экологии, техники безопасности и др.

Не рассматривалась.

- Является ли работа продолжением курсовых проектов (работ), есть ли публикации

Работа не является продолжением курсовых проектов. Данная работа не имеет публикаций.

Практическая ценность работы. Рекомендации по внедрению

Результаты, полученные в данной работе, могут быть использованы во многих областях, где возникает потребность в разработке искусственного интеллекта, например, в разработке коллекционных карточных игр.

Выпускник_____
(подпись)

Руководитель_____
(подпись)

“_____” _____ 2017 г.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

С ранних игр-лабиринтов (как Pac-Man, Namco, 1980) и платформеров (как Donkey Kong, Nintendo, 1981), сегодняшние видео игры прошли длинный путь и имеют богатую историю.

В начале большинство игр были просты и требовали от игрока способности быстро реагировать, чтобы победить.

Позднее, с появлением лучших способов хранить и распространять информацию, таких как CD и DVD диски, видео игры начали становиться всё более и более сложными, имея богатый 3D мир и сложные взаимодействия между игроком и средой. Видео игры стали более реалистичными с точки зрения визуальной составляющей, и сложность, которую они несли в себе, потребовала от игрока постоянного внимания ко многим составляющим игры.

Многие игры имеют режим, где игрок может сразиться с искусственным интеллектом (AI). В 80-ых, за Пакманом (Pac-Man) уже гонялись небольшие приведения, которые управлялись набором правил. Создание AI для них не было сложной задачей, поскольку среда и цель были довольно примитивны. Но с развитием игр создание хорошего AI становится всё более сложной задачей.

До сих пор в некоторых играх AI управляется наборами правил, которые, в зависимости от ситуации, выполняют заданную последовательность действий. Такой способ не подходит для игр с более сложной средой, поскольку игра может содержать бесконечное количество ситуаций, и написать скрипты для каждой за разумное время невозможно.

К счастью, с приходом эры машинного обучения создание AI перешло на новый уровень. На замену AI, стратегии которых описаны разработчиком, пришли агенты, стратегии которых основаны на действиях множества игроков.

В данной работе рассмотрена разработка AI для коллекционной карточной онлайн-игры Hearthstone [1]. Игра была выпущена в 2014 году и уже успела привлечь 70 миллионов [2] человек своей простотой в обучении и глубиной стратегий. Игровой процесс сводится к дуэли двух игроков, которые оперируют своими колодами, состоящими из карт. Победителем в дуэли

становится игрок, уничтоживший героя противника (более подробное описание игрового процесса будет приведено в первой главе). Hearthstone, имея очень простые правила, содержит огромную глубину и не может быть описан одной лишь стратегией, именно поэтому он был выбран для этой работы.

ГЛАВА 1. ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Предметная область

В данной главе кратко описаны основные правила и механики игры, которые используются в данной работе. Вначале будет дано описание игрового процесса и текста карт. После этого будут описаны результаты существующих работ, их преимущества и недостатки, также будут приведены теоретическое описание методов машинного обучения, используемых в работе. В конце главы будет приведена формальная постановка задачи.

1.1.1. Игровой процесс

Hearthstone – это пошаговая стратегия для двух игроков, разработанная Blizzard Entertainment.

Рисунок 1 иллюстрирует как выглядит *стол* – состояние игры. Стол имеет две стороны, по одной для каждого игрока. Каждый игрок представлен *героем*, у каждого героя есть уникальная *сила героя* и *здоровье* (в начале игры оно равно 30). Цель игры – понизить здоровье противника до нуля. Чтобы это сделать, игроки разыгрывают карты из *руки*. Карты могут быть *существами* или *заклинаниями*. Содержание руки противника скрыто, можно лишь узнать количество карт. Каждый игрок имеет *колоду* – набор из 30 карт, которые игрок выбирает сам до начала игры. В колоде одинаковые карты могут встречаться не более, чем два раза. В начале игры колоды перемешиваются, и, хотя игрок и знает содержание своей колоды, он не знает какая карта находится сверху колоды.

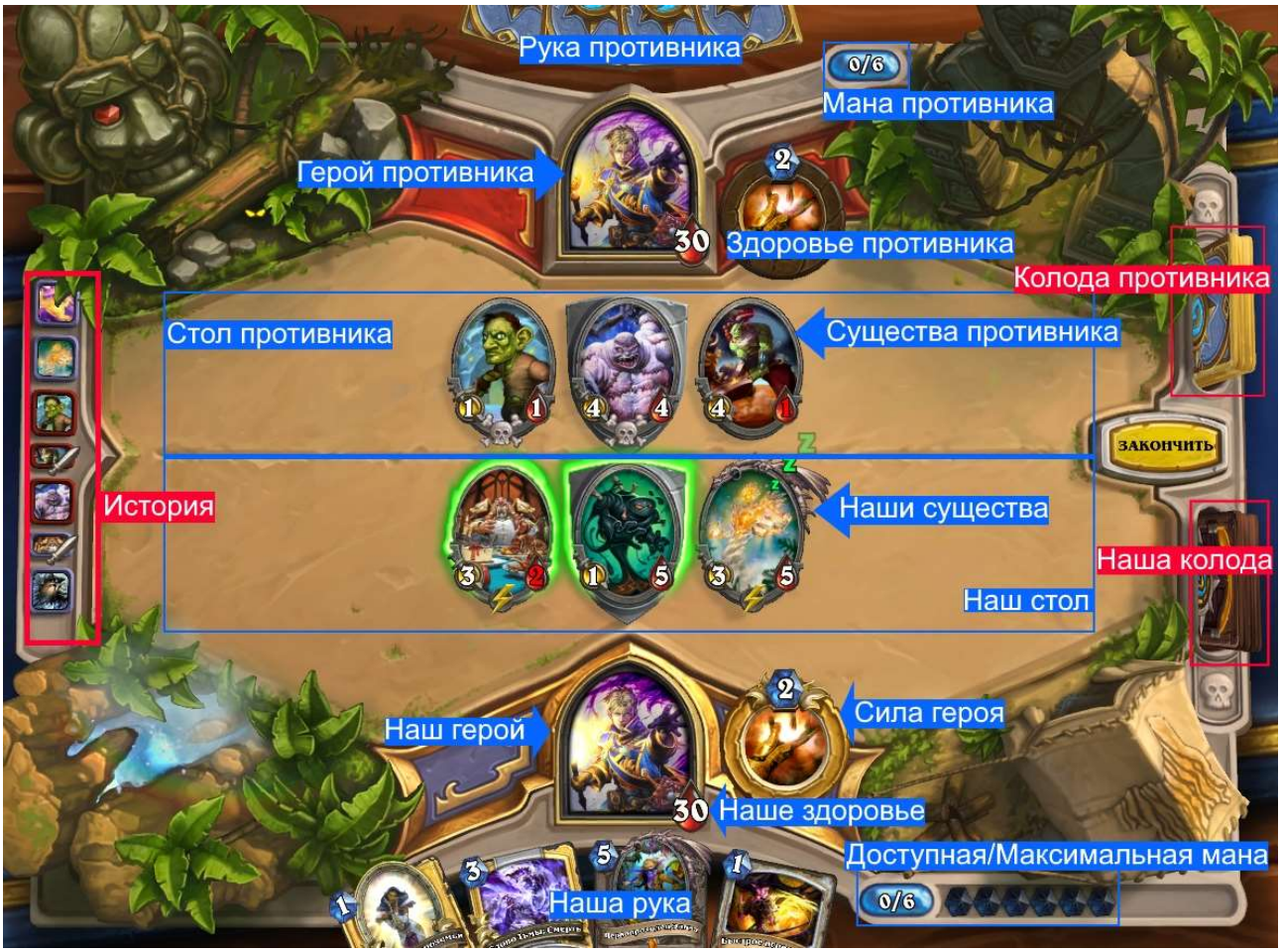

Рисунок 1. Пример игрового процесса Hearthstone с обозначениями его характеристик

Hearthstone основан на пошаговой системе передачи ходов между игроками в течение матча, то есть игроки никогда не действуют одновременно. В начале своего хода игрок тянет карту из колоды, если это возможно (колода не пуста). В течение своего хода игрок может разыграть любое количество карт из своей руки. Если у него есть существа на столе, то он также может их использовать для атаки героя или существа противника. Игрок может использовать силу героя каждый ход, но только один раз.

Разыгрываемые карты расходуют *ману* – ресурс, привязанный к игроку, восполняемый полностью в начале каждого хода. Изначально каждый игрок имеет один кристалл маны (единица маны), в каждом последующем ходе игрок имеет на один кристалл больше, пока не достигнет максимума в 10.

Розыгрыш карты приводит к раскрытию её противнику. Если это карта заклинания, то её текст выполняется, а карта сбрасывается. Если это карта существа, то игрок выбирает позицию на столе, куда его поставить, и исполняется *боевой клич* – специальный текст у некоторых существ. После попадания на стол, существа имеют две основные характеристики: показатели здоровья и атаки, которые обозначаются целым, не отрицательным числом, и некоторые другие свойства, которые будут описаны позже. Здоровье показывает, сколько урона нужно нанести данному существу, до того, как оно умрёт (показатель здоровья станет равен нулю). Атака указывает на то, сколько урона нанесёт данное существо, если атакует или будет атаковано другим существом или героем. На каждом ходу существо может атаковать лишь один раз, если иное не указано в тексте карты.

1.1.2. Текст карт

Как правило карты содержат текст. Текст – это описание различных характеристик карт: что она делает, когда разыграна, реакция на определённое действие и др.

Рисунок 2. Карта заклинания «Слово силы: Щит» и существа «Жрец Когтя из «Кабала». Изображения карт взяты из [1]

Текст карты может быть абсолютно любым – от многословного описания действия заклинания до пустого поля у некоторых существ. На рисунке 2 изображены карты заклинания, которое стоит 1 ману (слово «единицы» будет опускаться далее, как общепринятое сокращение), и существа, которое стоит 3 маны, имеет 3 атаки, 4 здоровья и *боевой клич*.

Текст заклинания «Слово силы: Щит» несёт в себе информацию о возможной цели заклинания – «Выбранное существо…», что означает, что только существо можно выбрать в качестве цели заклинания (нельзя выбрать героя); «…получает +2 к здоровью. Вы берёте карту.» – означает, что выбранное существо получит +2 к показателю здоровья и игрок, разыгравший данное заклинание, возьмёт карту из колоды.

Текст карты может также содержать *ключевые слова* – сокращения, как правило, выделенные полужирным шрифтом. Далее мы приведём описание некоторых из них.

Боевой клич – текст, который исполняется, когда существо попадает на стол. В случае карты «Жрец Когтя из «Кабала» (рисунок 1.2), текст несёт в себе информацию о возможной цели действия боевого клича и его эффекте. Текст «… вашего существа…» означает, что целью могут быть только существа, которые находятся на половине стола игрока (союзные существа). Слова «… увеличивает здоровье… существа на 3» означают, что выбранное нами союзное существо получить +3 к показателю здоровья. *Рывок* – свойство, позволяющее нарушить правила Hearthstone. Обычно существа, разыгранные на этом ходу, не могут сразу атаковать, однако наличие слова «рывок» в тексте позволяет нарушить это правило. *Провокация* – свойство существа; если игрок хочет атаковать существом, он вынужден выбрать в качестве цели одно из существ противника с такой способностью. *Божественный щит* – свойство существа, которое блокирует наносимый урон один раз, после чего существо теряет данное свойство.

1.1.3. Нейронные сети

Машинное обучение [3] – поле информатики, в которое позволяет обучаться компьютерам, не будучи специально запрограммированными.

Одной из моделей машинного обучения является нейронная сеть [4] – система, основанная на соединении перцептронов в слои.

Перцептрон – это математическая модель, основанная на представлении о работе нейронов мозга, позволяющая устанавливать ассоциации между данными на входе и выходе. На рисунке 3 приведено схематическое изображение перцептрона.

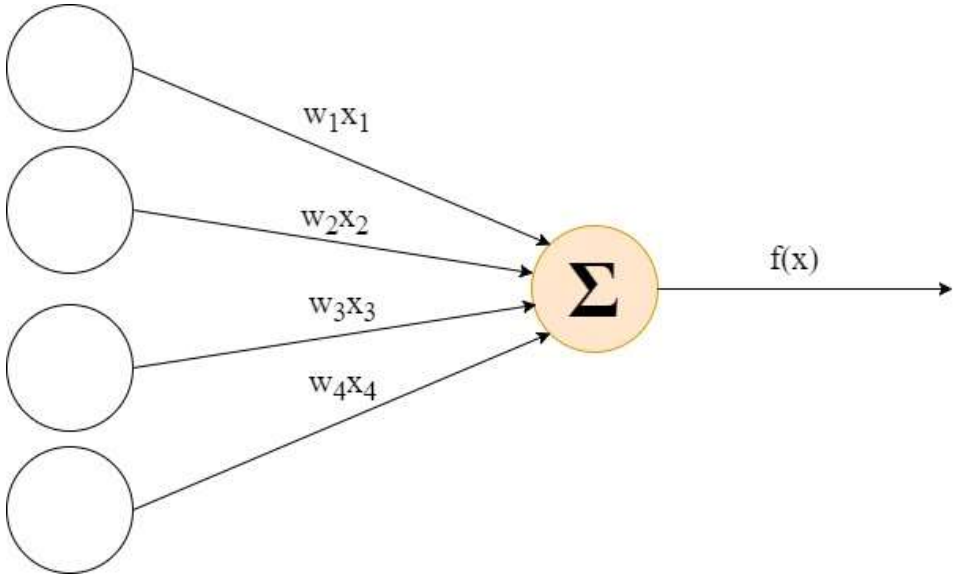

Рисунок 3. Схематическое изображение модели перцептрона

В вершину (отмечена жёлтым) поступают сигналы от других перцептронов $x_1, x_2, \dots, x_n$, далее эти сигналы преобразуются и суммируются в соответствии с весами $w_1, w_2, \dots, w_n : x = \sum_{i:=1}^{n} w_i \cdot x_i$. Затем к полученному сигналу применяется f – функция активации, и сигнал передаётся дальше. Полностью связанная (fully connected) нейронная сеть представляет собой структуру из нескольких слоёв перцептронов, которые связаны между собой всеми возможными способами. Перцептроны внутри слоёв не имеют соединений. Слои различают по функции активации, которые используются в перцептронах слоя, их количеству и вешний это слой или внутренний. В данной работе будут использоваться нейронные сети прямого распространения.

1.1.4. Q-обучение

Обучение с подкреплением [5] – это область машинного обучения, в которой модель обучается взаимодействовать со средой путём воздействия на неё а также способ максимизировать итоговую награду за свои действия в этой среде.

Q-обучение [6] – это техника обучения с подкреплением, позволяющая найти оптимальную стратегию выбора действий для любой конечной Марковской цепи [7]. Q-обучение пытается узнать ценность нахождения в текущем состоянии и ценность выбора конкретного действия из этого состояния. В самом простом варианте Q-обучение – это таблица значений для каждого состояния (строки) и действия (столбца). В каждой ячейке таблицы, мы ищем насколько выгодно данное действие не только в данный момент, но и после нескольких переходов. Чтобы обновлять значения в такой таблице используют уравнение Беллмана:

$$Q(s,a) = r + \gamma \left(\max_{a' \in A'_s} Q(s',a') \right)$$

Оно говорит о том, что долгосрочная награда $Q(s,a)$ за выбранное действие a из состояния s равна сумме мгновенной награды за выполнение действия – r и ожидаемой награды за лучшее действие из состояния, полученного путём выполнения действия a в состояние s – s'. Обновляя таблицу таким способом, мы постепенно получим точные значения функции Q.

Если же Марковскую цепь не задать таблицей, например, среда имеет почти бесконечное количество состояний, используют нейронные сети как способ аппроксимации функции Q(s, a).

1.1.5. Обзор существующих работ

В области коллекционных карточных игр существует довольно мало работ, использующих преимущества машинного обучения.

Одна из первых работа была опубликована в августе 2014 года. Эли Бёрстэйн [8] описал как с помощью статистического обучения получить преимущество, предсказывая следующий ход противника. К сожалению, результаты его работы не применимы в нашем случае, так как за 4 года игра сильно изменилась, и сейчас, колода противника состоит практически из случайных карт.

Следующая большая работа была написана в 2015 году. Дэвид Таралла [9] создал модель, которая способна выигрывать 93,45% игр против случайного агента (действия выбираются с равной вероятностью) и 10,96% против сценарного агента (все действия выбираются с помощью набора правил). В этой работе использовали обучение классификатора с учителем, основанное на сильно случайных деревьях [10], что позволило автору научить Нору (имя агента) побеждать случайного агента.

Основным недостатком подхода этой работы является отсутствия учёта взаимодействия между картами и действиями, так как Нора сравнивает текущие возможные действия между собой и выбирает лучшее. Цепочки таких действий построены жадно, что не позволяет ей побеждать сценарного агента, так же не позволяет учесть взаимодействия между картами, которых довольно много в Hearthstone.

В июне 2016 Герберт Тео, Ютонг Вэнг и Джирен Джу опубликовали работу [11], в которой при помощи дерева действий и алгоритма поиска A* [12], обучения с учителем и обучения с подкреплением, смогли достигнуть 75% побед над эвристическим агентом для обучения с учителем и 50% побед для обучения с подкреплением при тестировании на 100 играх. Для каждого хода они строили дерево возможных действий, листьями в котором являлось действие «закончить ход», затем вычисляли ценность каждого конечного состояния и выполняли лучшую цепочку действий. Ценность состояния определялась двумя методиками: нейронной сетью, которая была обучена за счёт разметки цепочки истории состояний от конца к началу с понижающимся значением и Q-обучением.

Результаты этой работы лучше предыдущей работы, но не достоверны. После исследования инженерной реализации описанных в работе моделей, а также тестирования их на большом наборе игр (по исследованиям [9], точные результаты получаются после 10 тысяч игр), было обнаружено, что лучший результат, который был достигнут в работе [11] является 66,87% при игре против эвристического агента на тренировочной колоде и 67,35% при замере на случайных колодах. Эти результаты были взяты как базовые в нашей работе.

1.2. Постановка задачи

В данной работе будут использованы различные подходы машинного обучения для разработки агента, способного превзойти результаты предыдущих работ. Предполагается, что разделение игрового процесса на две части – стол и рука, где модель руки зависит от модели стола позволит применить Q-обучение к модели стола. Кроме того, будет предложен способ построения полного дерева действий и его оценки, который позволит не разделять игру на подмодели, что даст возможность применять данный способ на более широком спектре задач.

Выводы по главе 1

В главе были рассмотрены подходы, к созданию агента для игры в Hearthstone, также были описаны основные механики самой игры, некоторые модели машинного обучения.

Далее в работе будут рассмотрены модели, сведённые к описанным задачам машинного обучения.

ГЛАВА 2. ТЕОРЕТИЧЕСКОЕ ОПИСАНИЕ АГЕНТОВ

В данной главе будет описан способ применения агента к игре Hearthstone, приведён набор характеристик, которые будут передаваться использованным методам машинного обучения, также приведено описание агентов, основанных на Q-обучении и приведено описание агентов, основанных на методах обучения с учителем.

2.1. Применение агентов для игры в Hearthstone

Игра в Hearthstone имеет структуру, основанную на ходе. Свой ход игрок начинает и некоторого состояния игры s_0, затем, выполняя действия, игрок переходит в состояния $s_1, s_2, \dots, s_n$, где состояние s_n, получено выполнением действия, связанным с передачей хода противнику. Данное состояние s_n становиться состоянием s_0 для противника. В данной работе агент будет применяться для генерации выполняемой последовательности действий внутри хода игрока. Схема применения агента изображена на рисунке 4.

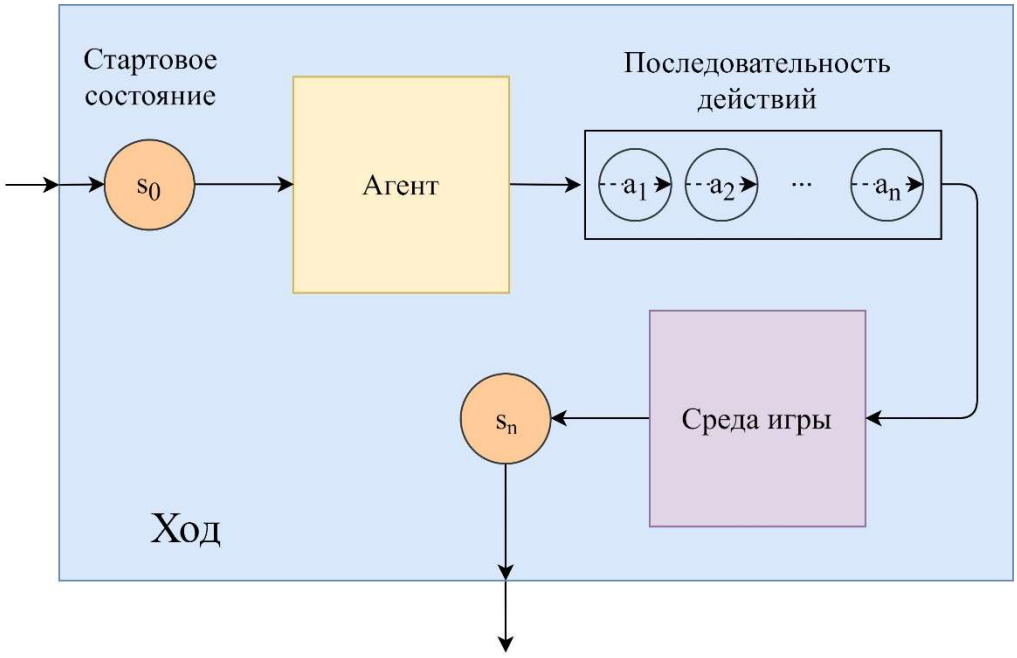

Рисунок 4. Схема интеграции агента в игру Hearthstone

На данной схеме агент в начале хода получает состояние игры s_0 и выдаёт последовательность действий, которая выполняется в среде игры для генерации состояния s_n.

Агент состоит из двух частей: выделения характеристик и набора моделей, которые генерируют последовательность действий. Схематическое изображение агента приведено на рисунке 5.

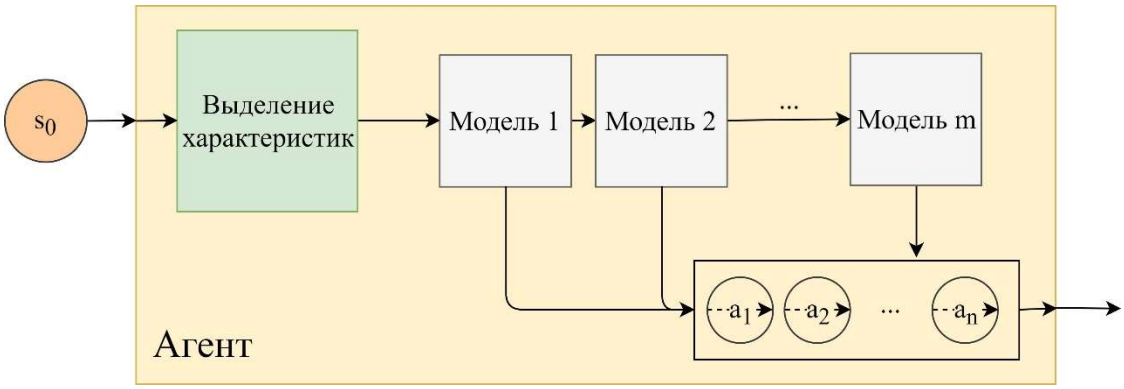

Рисунок 5. Схема устройства агента для игры в Hearthstone

2.2. Характеристики среды

Чтобы свести нашу задачу к задаче машинного обучения, нужно выделить характеристики среды, которые бы наилучшим способом задавали состояние игры, но в то же время не были слишком сложны. Выбранные характеристики описаны в таблице 1.

Таблица 1. Характеристики, описывающие состояние игры

Номер характеристики	Описание характеристики	Область значений
0	Здоровье нашего героя	$\mathbb{Z}$
1	Здоровье героя противника	$\mathbb{Z}$
[2,85]	14 шестимерных вектора для характеристик существ	(Таблица 2)
[86,93]	2 четырёхмерных вектора для характеристик игрока	(Таблица 3)

94	Номер хода	$\mathbb{Z}$
95	Суммарная доступная атака существ на этом ходу	$\mathbb{Z}$

Таблица 2. Характеристики, описывающие существо

Номер характеристики	Описание характеристики	Область значений
0	Здоровье существа	$\mathbb{Z}$
1	Атака существа	$\mathbb{Z}$
2	Может атаковать на этом ходу	[0,1]
3	Имеет провокацию	[0,1]
4	Имеет божественный щит	[0,1]
5	Есть на столе	[0,1]

Таблица 3. Характеристики, описывающие состояние игроков

Номер характеристики	Описание характеристики	Область значений
0	Суммарный урон существ	$\mathbb{Z}$
1	Количество существ	$\mathbb{Z}$
2	Количество карт в руке	$\mathbb{Z}$
3	Количество карт в колоде	$\mathbb{Z}$

2.3. Агенты, основанные на Q-обучении

Далее будут описаны две модели, которые разделяют общий подход к принятию решения на каждом ходу в Hearthstone. Так как действий в Hearthstone, которые может совершить игрок, очень много, а Q-обучение работает на ограниченном наборе действий, далее будет рассмотрен способ сократить количество действий, чтобы обеспечить возможность сведения нашей задачи к задаче Q-обучения.

2.3.1. Базовая теория

В Hearthstone действия, которые игрок выполняет, можно разделить на две части: действия, связанные с рукой (разыграть карту, использовать силу героя), и действия, связанные со столом (выбрать существо для атаки и цель для атаки). Таким образом, игровой процесс можно разделить на две части: стол и рука. Подмодель руки и множество действий, которое она содержит, может влиять на модель стола и на состояние модели руки. Модель стола и множество действий, содержащиеся в ней, может влиять только на себя, но не на модель руки (за исключением нескольких карт, которые не рассматриваются в данной работе). Таким образом, если научить агента принимать решения, связанные со столом, он сможет принимать решения, связанные с рукой, опираясь на изменения в модели стола.

2.3.2. Модель с 57 действиями

Первая модель, которую мы рассмотрим, будет модель с 57 действиями. В любой момент времени на столе у игрока может быть от 0 до 7 существ, то же самое верно и для противника. Совершение атаки в Hearthstone – это двухфазное действие. Сначала игрок выбирает атакующее существо на своём столе, затем атакуемого персонажа, который будет атакован атакующим существом. Персонаж включает в себя существо на столе противника или героя противника. Таким образом, максимальное количество вариантов атаки равно $7 \cdot 8 = 56$, 57-ым действием будет возможность для агента не выполнять действие, в нашем случае это нажатие кнопки «Закончить ход», то есть передать ход противнику.

Для аппроксимации функции $Q(s, a)$ будем использовать многослойную нейронную сеть, основанную на перцептронах. Чтобы обучить эту модель будет использован алгоритм, опубликованный в работе [16], он приведён на листинге 1.

Листинг 1. Алгоритм обучения нейронной сети для функции Q(s, a)

Инициализировать память записей D размера N
Инициализировать нейронную сеть для Qс случайными весами
for Эпизод $= 1, M$ **do**
  while Игра не закончилась **do**
    С вероятностью ϵ выбрать случайное действие $a_i \in A_s$
    Иначе выбрать действие $a_i = \max_{a_i \in A_s} Q^*(s_i, a_i)$
    Выполнить действие a_i в симуляции, получить награду r и следующее состояние s'
    Перейти в состояние s'
    Сохранить в D переход (s, a_i, r, s')
    Получить случайный набор переходов T из D
    for $(s, a_i, r, s') \in T$ **do**
      $y =$ оценки действий для s
      $$y_i = \begin{cases} r_i, \text{если } s' - \text{терминальное состояние} \\ r_i + \gamma \max_{a' \in A_{s'}} Q(s', a'), \text{иначе} \end{cases}$$
      Выполнить обучение нейронной сети на s и обновлённой y
    end for
  end while
end for

Обучаемый агент будет играть против эвристического агента, использованного в работе [11] в рамках модели стола. То есть оба агента не будут иметь возможности разыграть карту из руки или использовать силу героя. Так как существа появляются в модели стола только за счёт действий модели руки, будем генерировать эпизоды для обучения путём игры двух случайных агентов между собой. Таким образом, мы получим наиболее полный набор состояний стола, что улучшит наши результаты.

После того как будет обучена модель стола, будет реализовано дерево действий руки, вершины в котором – это состояния, полученные путём переходов – действий.

Агент, основанный на описанных моделях руки и стола, будет играть в Hearthstone следующим образом: в начале каждого хода будем строить дерево действий руки, сравнив листы в этом дереве с помощью модели стола, путём вычисления среднего значения награды за каждое действие модели стола, будем выполнять лучшую последовательность действий дерева. Затем управление передаётся модели стола, которая может использовать существа для атаки или передать ход противнику. Схема устройства агента приведена на рисунке 6.

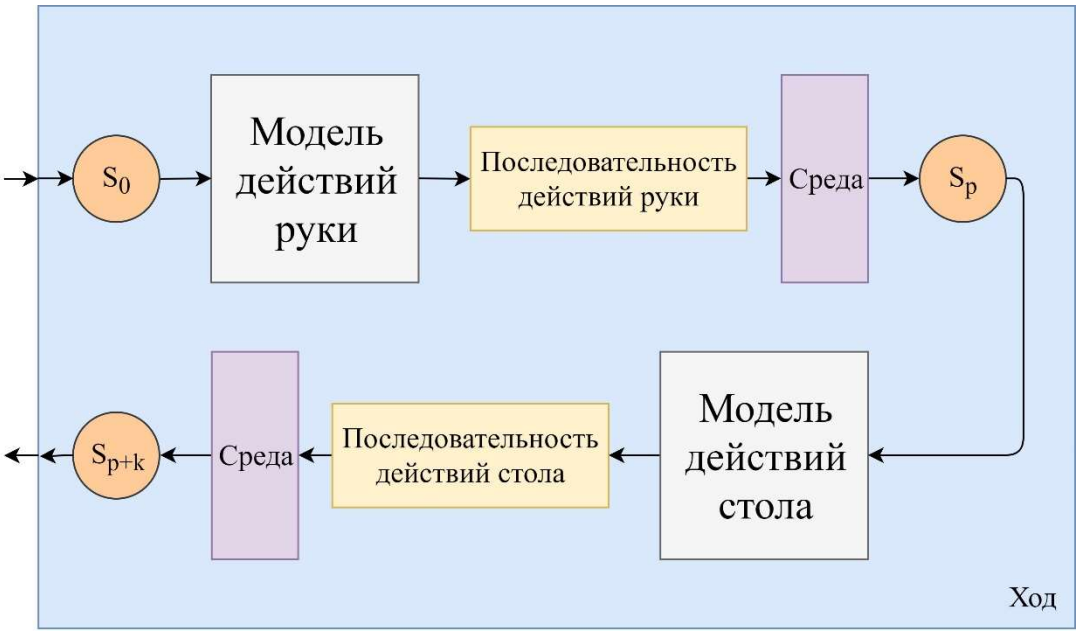

Рисунок 6. Схема устройства агента, основанного на Q-обучении

2.3.3. Модель с 8 действиями

Вторая модель, основанная на Q-обучении, практически идентична предыдущей, за исключением нескольких различий в модели стола. Вместо 56 действий, выражающих все возможные комбинации действия атаки, модель Q-обучения будет использована для 7 действий, определяющих выбор атакующего существа, и действия «Закончить ход». Выбор атакуемого существа будет производиться с помощью обучения с учителем MLP (multilayer perceptron) на действиях эвристического агента. Генерация набора данных будет произведена путём записей атакуемых существ в игре между двумя эвристическими агентами.

Таким образом, описанный агент в начале каждого хода будет строить дерево действий, выбирать и исполнять лучшую последовательность действий на основе модели стола, затем управление будет передаваться модели стола. Модель стола будет выбирать между выполнением атаки определённым существом и передачей хода противнику. Если модель выбрала атакующее существо, управление передаётся модели выбора атакуемого существа. Такой подход позволит ускорить обучение нейронной сети в основе Q-обучения, а

также избежать проблем с выбором одного и того же типа действия, которые могут возникнуть у модели с 57 действиями.

2.4. Агенты, основанные на дереве действий

Второй вид моделей, которые рассмотрены в этой работе, не делают разделение на типы действий, что позволяет не делить игру на подмодели, которые ограничивают нас в свободе изменения порядка действий.

Агенты, основанные на обучении с учителем, имеют две части: построение дерева действий и оценка листов в этом дереве. С помощью дерева действий мы получаем набор конечных состояний – листов, которые сравниваются между собой с помощью модели оценки листов (метрики состояния игры). Последовательность действий, ассоциированная с лучшим листом становиться результатом работы агента. Схема этого вида агентов приведена на рисунке 7. Далее будет приведено теоретическое описание этих моделей.

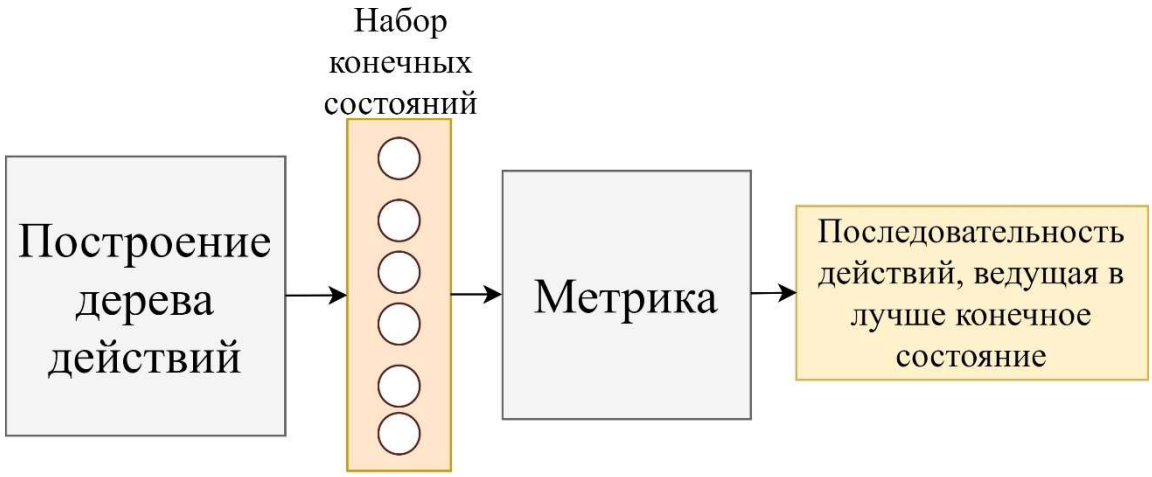

Рисунок 7. Схема устройства агента, основанного на дереве действий

2.4.1. Построение дерева действий

Дерево действий – это дерево, в котором вершины задаются состояниями, а переходы – действиями. В листы такого дерева ведут переходы, ассоциированные с передачей хода противнику. Цель такого дерева – определить лучшую последовательность действий на основе сравнения конечных состояний

в цепочке действий. При простом подходе построение дерева действий сводится алгоритму, представленному на листинге 2.

Листинг 2. Наивный алгоритм построения дерева действий

```
function ПОСТРОЕНИЕ ДЕРЕВА(s, a)
    s' = совершить a в s
    A' = дествия возможные в s'
    for a' ∈ A' do
        if a' = конец хода then
            Return s'
        else
            Добавить к ответу Построение дерева(s', a')
        end if
    end for
    Return ответ
end function
```

Такой способ построит полное дерево возможностей хода, но ведёт к большому размеру дерева, который может доходить до миллиардов вершин, а обработка занимать часы. Хотя такой способ даёт возможность проверить все возможные ходы, что позволяет совершить самое корректное действие, в Hearthstone существует ограничение по времени, отведённому на ход игроку. На данный момент это ограничение равно 75 секундам. Поэтому наивный алгоритм построения дерева действий не применим в условиях нашей задачи.

В Hearthstone некоторые действия могут быть выполнены в любом порядке и иметь одинаковый конечный результат, другие же сильно меняют среду и доступные действия. На рисунке 8 приведены две карты, которые могут быть разыграны в любом порядке между собой. Как видно из текста карты «Слово Тьмы: Смерть», её целью не может быть «Страж Луносвета», поэтому порядок разыгрывания этих двух карт из руки может быть изменён.

С другой стороны, карты, приведённые на рисунке 9 нельзя поменять местами в последовательности действий без изменения исхода.

Рисунок 8. Карты «Слово Тьмы: Смерть» и «Страж Луновсета», которые можно разыграть в любом порядке. Изображения карт взяты из [1]

Рисунок 9. Карты «Зелье огня Скверны» и «Демон Бездны», которые нельзя разыграть в любом порядке без изменения конечного результата. Изображения карт взяты из [1]

Из-за действия заклинания «Зелье огня Скверны», если разыграть «Демона Бездны» до использования заклинания, то существо погибнет, если наоборот, то существо не будет затронуто действием заклинания, а значит выживет. Поэтому важно уметь различать последовательности, ведущие к

одинаковому промежуточному результату, так как таковых в дереве действий хода много.

Чтобы сократить размер дерева и не посещать исследованные состояния в начале построения дерева, пройдём до самого конца одну из последовательностей, сохранив её в памяти. При совершении перехода, будем сравнивать новое состояние с состояниями в сохранённой цепочке. Если новое состояние идентично одному из последовательности или доминируется состоянием из последовательности, то мы не будем продолжать рассматривать текущую ветвь, в противном случае продолжаем рассматривать последовательность. Если конечное состояние последовательности лучше сохранённой (в соответствии с метрикой оценки листа дерева), заменим последовательность в памяти на новую и продолжим.

2.4.2. Оценка листов дерева действий

Оценивать состояние игры можно разными способами. Можно, опираясь на экспертное мнение, реализовать функцию, которая по характеристикам будет возвращать число – некую оценку «хорошести» состояния. В данной работе мы рассмотрим другой способ, который позволяет не опираться на экспертное мнение. Чтобы дать оценку листам дерева состояний, будет реализовано два подхода: обучение с учителем на полной цепочке состояний игры и обучение на цепочках состояний автомата, основанного на уровне угрозы.

Суть обучения на цепочках состояний похожа отдалённо на Q-обучение. Во время игры мы сохраняем последовательность состояний, в которых мы побывали. По окончанию игры цепочка размечается в соответствии с функцией $f(i,n) = \gamma^{n-i} \cdot reward$, где γ – это коэффициент потери, $reward$ – это награда за победу или поражение, i – это номер в состояния в цепочке, n – это длинна цепочки. Каждое состояние в цепочке получает определённую оценку. Таким образом, состояния наиболее близкие к моменту победы получат лучшую оценку

по модулю, нежели состояния, через которые мы проходим в начале игры. Пример разметки цепочки приведён на рисунке 10.

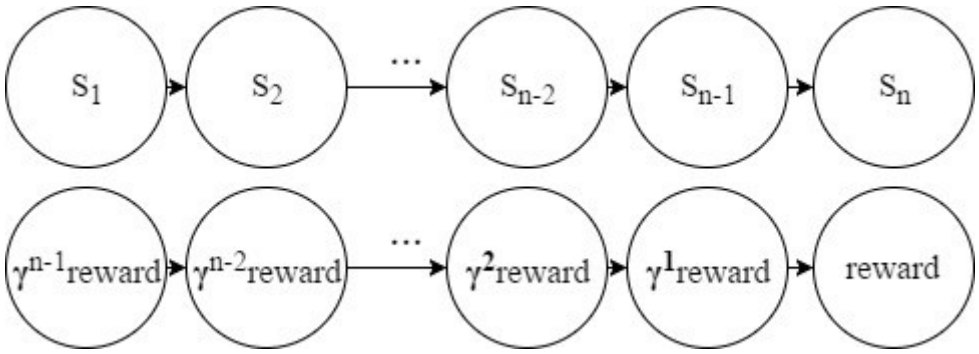

Рисунок 10. Пример разметки цепочки состояний

Чтобы сгенерировать набор пар состояние-оценка, используем двух эвристических агентов, которые будут играть между собой, добавляя состояния в цепочку в конце своего хода. После игры промаркируем состояния в цепочках и добавим их в учебный набор данных. На данном наборе данных обучим нейронную сеть.

Другой способ оценки состояния, который будет рассмотрен в данной работе – это модель, основанная на уровне угрозы. Будем различать три уровня угрозы: зелёный – низкий уровень угрозы, жёлтый – средний уровень угрозы и красный – высокий уровень угрозы. Чтобы различать уровни угрозы в состояниях, введём параметр *эффективное здоровье* (Effective HP). Эффективное здоровье – это разность между здоровьем героя и суммой значений атаки существ на столе противника. Таким образом, игру можно задать автоматом, в котором есть переходы между зелёной и жёлтой вершинами, жёлтой и красной, у всех трёх цветных вершин есть переход в состояние победы, у красной вершины есть переход в состояние поражения. Изображение данного автомата приведено на рисунке 11.

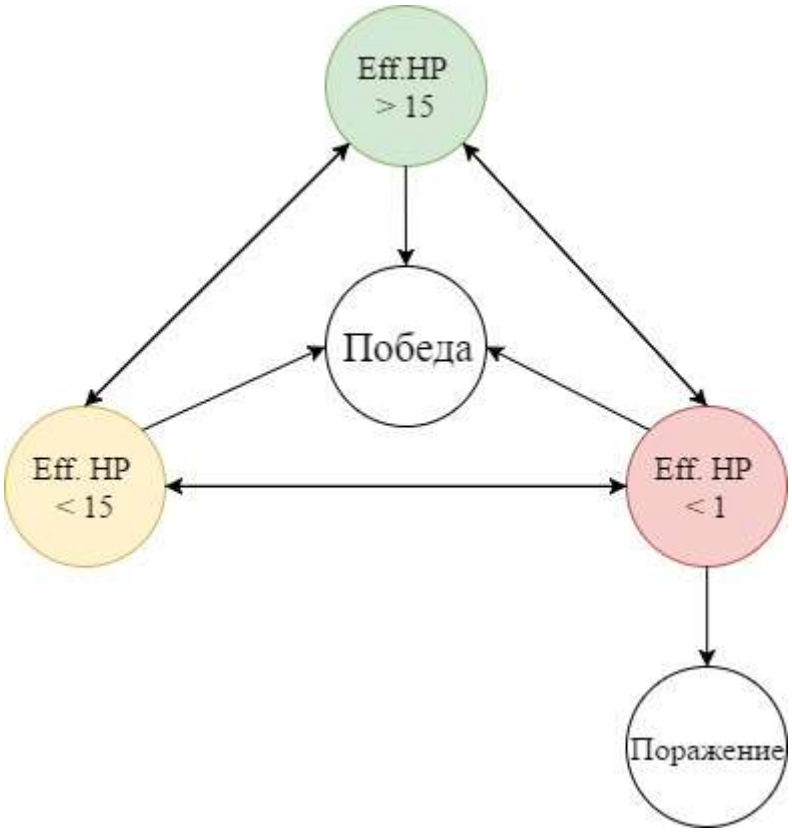

Рисунок 11. Схема автомата, основанного на уровне угрозы

При переходе от жёлтого состояния к другому, будем выделять подцепочку жёлтого цвета и размечать в соответствии с формулой, описанной выше. Тоже самое верно и для других цветов. Пример разметки жёлтой цепочки приведён на рисунке 12. Цепочка начинается с первого состояния после перехода в другой цвет и заканчивается первым состоянием нового цвета или терминальной вершиной (состояние победы или поражения).

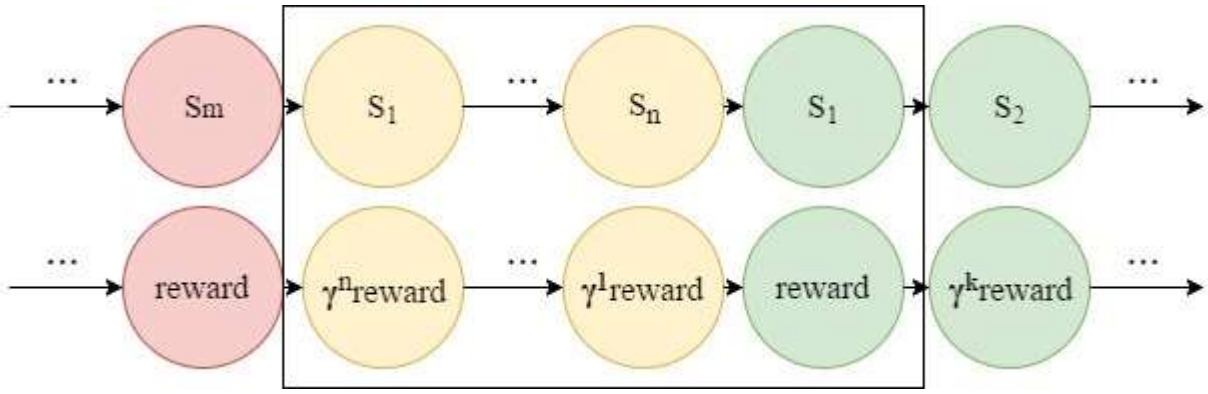

Рисунок 12. Пример разметки жёлтой цепочки.

Чтобы сгенерировать учебные наборы, также будем использовать двух эвристических агентов, но сохранять состояния будем после каждого действия. Таким образом, получим три набора данных для каждого состояния автомата. На этих наборах данных обучим три нейронных сети. Этот подход, по сравнению с предыдущим, должен исправить проблему стартовых состояний, в которых при использовании предыдущего метода будут сильные и разнящиеся значения.

Выводы по главе 2

В данной главе было приведено описание моделей на основе разделения игры Hearthstone на две подмодели с применением Q-обучения к одной из них, модели, основанные на дереве действий и обучение с учителем, а также способы их обучения, реализация которых будет описана в главе 3.

ГЛАВА 3. РЕАЛИЗАЦИЯ АГЕНТОВ

В данной главе будут кратко описаны существующие симуляторы Hearthstone. Также представлена архитектура реализованных агентов, описаны результаты их работы. В конце будет проведён сравнительный анализ результатов работы агентов.

3.1. Симуляторы Hearthstone

Для реализации агентов, описанных в главе 2, требуется среда, с которой агенты смогут взаимодействовать. Использовать в этих целях Hearthstone, разработанный Blizzard Entertainment [17], не представляется возможным по нескольким причинам. Потребность узнавать состояние игры по визуальной составляющей (картинке на экране), что само по себе довольно сложная задача из-за большого количества эффектов и разнообразия изображений, длительность анимации – совершение действия влечёт за собой довольно длинную анимацию (примерно 0.1 с.), что не позволяет играть и обучаться быстро. Поэтому далее рассмотрены несколько существующих симуляторов Hearthstone, которые позволяют легко узнавать характеристики среды и выполнять сотни игр в короткий период времени.

Fireplace[1] – симулятор, написанный на языке python, содержит в себе все базовые механики, подходит для симуляции большого количества игр, но на момент написания этой работы не обновлялся долгое время, и не содержал карт трёх последних дополнений Hearthstone (обновления, содержащие большие количества новых карт и механик).

Hearthbreaker [2] – проект был создан специально для применения машинного обучения, имеет пользовательский интерфейс, но также не имеет двух последних дополнений. Стоит отметить, что этот симулятор использовался в работе “Will our new robot overlords play Hearthstone with us?” [11], поэтому хотя

[1] https://github.com/jleclanche/fireplace/
[2] https://github.com/danielyule/hearthbreaker

мы и не будем пользоваться Hearthbreaker, нам понадобится реализовать копию агента, который был использован для проведения симуляций с целью получения результата.

Metastone [1] – самый полноценный (реализованы все существующие карты) на данный день симулятор Hearthstone, написанный на Java, содержащий в себе имитацию визуальной составляющей Hearthstone. В данном проекте также предусмотрено место для реализации агентов и имеется набор AI. Внутри этого проекта и будут реализованы агенты, описанные в главе 2.

3.2. Подготовка

Для реализации агентов, описанных в главе 2, была реализована модель машинного обучения – нейронная сеть с возможностью создания различных конфигураций слоёв и функций активации. В качестве способа обучения в реализации был использован метод обратного распространения ошибки (backpropagation) с функцией потери MSE. Также в исходный проект был добавлен интерфейс, позволяющий выделить из объекта, описывающего состояние среды, характеристики из таблицы 1. Чтобы при обучении избежать создания неправильных зависимостей на основе номера ячейки в векторе характеристик, интерфейс случайно расставляет существа по позициям.

3.3. Агенты, основанные на Q-обучении

В данной главе будет описано исследование агентов, основанных на Q-обучении.

3.3.1. Модель с 57 действиями

В качестве мгновенной награды агент получал ноль во всех состояниях, кроме терминальных. Для обучения были выбраны следующие гиперпараметры:

[1] https://github.com/demilich1/metastone

фактор дисконтирования был взят $\gamma = 0{,}9$, значение мгновенной награды в терминальной вершине было взято за 100 для победы и -100 для поражения. Для аппроксимации Q-функции были испробованы несколько конфигураций нейронной сети. Самой успешной среди них оказалась нейронная сеть, имеющая 3 скрытых слоя по 64 нейрона каждый, с функциями активации сигмоида [18], и линейной функцией активации на последнем слое.

Обучение, описанное в главе 2, было проведено на 1500 играх случайных агентов, что позволило обучиться на более чем 2 миллионах переходов. При замере в игре против Trading agent [1] (далее эвристический агент), модель выиграла 52,34% игр у противника (этот и все дальнейшие замеры произведены на 10 тысячах игр), но при рассмотрении значений Q-функции для состояний, выяснилось, что действия, связанные с атакой героя противника, всегда получали на порядок большую награду, даже если агент начинал проигрывать игру. Это было связано со способом выдачи мгновенных наград – награды отличные от нуля агент получал только в конце игры, что существенно понижает скорость обучения.

Данную проблему получилось разрешить путём изменения способа выдачи мгновенной награды. Вместо награды в конце игры будем выдавать мгновенную награду за каждое действие. Чтобы выдавать мгновенную награду будем вычислять значение r по следующей формуле: $r = \rho(s') - \rho(s)$, где функция $\rho(s)$ – это функция оценки состояния, которая реализована в эвристическом агенте. В качестве фактора дисконтирования γ возьмём 0,8. После обучения модели с улучшениями, описанными выше, веса действий, связанных с атакой героя, упали, и модель стала пытаться улучшить проигрышную ситуацию, что дало 61,26% выигранных игр.

[1] Trading agent – копия агента из Hearthbreaker

3.3.2. Модель с 8 действиями

Модель с 8 действиями лучше предыдущей, так как не может выбирать цели атаки, соответственно не может постоянно атаковать героя противника. Способ выдачи мгновенных наград, основанный на эвристическом агенте, и гиперпараметры были взяты из предыдущей главы. За конфигурацию нейронной сети была взята конфигурация модели с 57 действиями.

Для модели, выбирающей цель атаки, была обучена нейронная сеть с одним внутренним слоем с 256 нейронами, выходным слоем размера 8. В качестве функций активации была использована сигмоида. Для обучения был собран набор данных размера 16 тысяч атак путём игры двух эвристических агентов между собой. Перед началом обучения данный набор был разделён на 6 частей: 5 учебных и 1 тестовая. Нейронная сеть была обучена пять раз на учебных частях, исключая по очереди каждую из них. Были замерены результаты на исключённых частях, среднее значение 5 замеров отличалось от замера на тестовой части примерно на 2%. Данный результат означает, что переобучение нейронной сети не было достигнуто.

Обученная нейронная сеть была использована в обучении модели с 8 действиями на 1000 игр случайных агентов, 1,5 миллионах переходов. После замера обученной модели были получены следующие результаты: 65,34% игр модель с 8 действиями выиграла у эвристического агента и 98,58% у случайного агента.

3.4. Агенты, основанные на дереве действий

Далее будет приведено описание процесса реализации агентов на основе дерева действий и генерации учебных наборов.

3.4.1. Реализация обхода дерева действий

Для реализации обхода дерева действий была разработана структура, позволяющая хранить ветви дерева и их ценность. Также был реализован

алгоритм проверки доминирования состояния. Перед проверкой на доминирование существа в обоих состояниях упорядочиваются в порядке убывания сначала атаки, затем здоровья, затем суммарного показателя свойств, вычисляемого по формуле $f(m) = 4 * m.taunt + 2 * m.divineShield + m.canAttack$, где m – это данное существо, а taunt, divineShield, canAttack – это показатели наличия определённых свойств у данного существа, которые могут принимать значения 0 или 1. Далее все характеристики сравниваются. Если все характеристики первого состояния больше или равны характеристикам второго и хотя бы одна из них строго больше, то первое состояние доминирует второе. Для проверки равенства был использован другой способ, который позволяет заметно сократить время проверки состояний на равенство. В начале каждого хода агент создаёт хеш-таблицу пар числового представления состояния и итоговой оценки за посещение данного состояния. Числовое представление состояния – это значение хэш-функции от строкового представления состояния. В качестве хэш-функции была использована функция, предоставляемая средой Java. Строковое представление состояния – это характеристики, описанные в главе 2, которые переведены в систему счисления с основанием 64 (за счёт использования заглавных и прописных букв английского алфавита), что позволяет отводить на каждую характеристику один символ, и объединённые в строку. Характеристики существ задаются тремя символами: атака, здоровье и значение функции свойств существа, описанной ранее.

3.4.2. Обучение на полных цепочках

Учебный набор для обучения на полных цепочках был сгенерирован путём игры между собой двух эвристических агентов. Всего было сыграно 1000 игр, что позволило образовать 2000 цепочек, примерно 20 тысяч состояний. Полученные цепочки были промаркированы в соответствии с результатом игры. Базовое значение награды было выбрано -100 и 100 за поражение и победу соответственно. Цепочки были промаркированы в соответствии со значениями

функции, описанной в главе 2, собраны в единый набор и перемешаны. Далее проводилось обучение нейронной сети с двумя внутренними слоями размера 128 и функцией активации сигмоида. Расхождение средней ошибки и ошибки на тестовом наборе было менее 3%.

Для проверки итоговых результатов работы агента были произведены замеры на случайном и эвристическом агентах. Полученные проценты выигранных игр агентом приведены в таблице 4.

Таблица 4. Результаты замеров агента, обученного на полных цепочках

	Эвристический агент	Случайный агент
Учебная колода	72,35%	99,99%
Случайные колоды	78,27%	100,0%

Учебная колода приведена в приложении, случайные колоды – это колоды, взятые с сайта Hearthpwn[1], всего было взято 10 случайных колод.

Как видно из результатов, модель, основанная на построении полного дерева хода, заметно улучшила результаты моделей, основанных на Q-обучении. Улучшение результатов нашего агента на случайных колодах, говорит о том, что агент смог пользоваться более сложными картами, которые не присутствовали в учебной колоде и лучше принимал решения за счёт отсутствия жадности в обходе дерева действий.

3.4.3. Обучение на цепочках автомата

Для генерации учебного набора были также использованы эвристические агенты, но состояния добавлялись в цепочку после каждого действия агента, что позволило лучше проследить закономерности в среде. Цепочки были собраны на тысяче игр между эвристическими агентами. Полученные подцепочки размечались с помощью функции из главы 2, значение reward равно -100 при

[1] http://www.hearthpwn.com/decks?filter-is-forge=1&filter-deck-tag=1

переходе в худшее состояние автомата (из зелёного в жёлтое, из жёлтого в красное, из красного в состояние поражения) и 100 при переходе в лучшее состояние (переход в состояние победы, из красного в жёлтое, из жёлтого в зелёное). Таким образом были получены три набора данных размерами 48, 38, 9 тысяч состояний для зелёного, жёлтого и красного состояния автомата соответственно. Из-за малого размера красного набора, были специально сгенерированы цепочки, начинающиеся в красном состоянии, путём случайного задания стартового состояния игры, удовлетворяющего условию $Eff.HP < 1$. Таким образом красный набор данных был пополнен до 40 тысяч состояний. Далее было обучено три нейронных сети с одинаковой структурой: два внутренних слоя размера 128. С функцией активации сигмоида, по методике, описанной в параграфе 3.3.2. Результаты работы обученного агента приведены в таблице 5.

Таблица 5. Результаты замеров агента, обученного на цветных цепочках

	Эвристический агент	Случайный агент
Учебная колода	82,24%	100,0%
Случайные колоды	84,35%	100,0%

Как видно из результатов, метод использования автомата, основанного на уровне угрозы, дал прирост в проценте побед над эвристическим агентом по сравнению с предыдущим методом. Связано это с тем, что предыдущий метод не даёт однозначных оценок на начальных состояниях игры, что зачастую приводит к поражению.

3.5. Сравнение с аналогами

Для сравнения нашей модели с агентом, описанным в работе [11], был реализован алгоритм поиска A*, и использована обученная модель, которую авторы работы предоставили. Замеры были произведены для всех моделей, описанных в нашей работе на 10 тысячах игр, на учебной и случайных колодах. Результаты приведены в таблице 6.

Таблица 6. Результаты замеров обученных моделей против агента из работы [11]

	Учебная колода	Случайные колоды
Q-модель (57 действий)	35,25%	32,21%
Q-модель (8 действий)	45,48%	44,79%
Дерево действий (полные цепочки)	56,97%	59,55%
Дерево действий (цветной автомат)	67,23%	72,90%

Как видно, Q-модели не смогли превзойти модель, описанную в работе [11]. Модели, основанные на дереве поиска, заметно улучшили предыдущий результат, особенно модель, основанная на автомате уровня угрозы, так как он позволил решить проблему с принятием решений на ранних ходах.

3.6. Тестирование асессорами

Для окончательной проверки качества модели, основанной на дереве поиска и цветном автомате, были проведены 100 матчей против 5 асессоров. Каждый игрок закончил 10 партий на учебной колоде и 10 партий на случайной. Результаты этого исследования приведены в таблице 7.

Таблица 7. Усреднённые результаты замеров игры агента против людей

	Учебная колода	Случайные колоды
Дерево действий (цветной автомат)	50,0%	42,0%

Данные результаты показывают, что наша модель сравнима с реальным игроком на механически не очень сложной колоде (учебная колода), но проиграла больше половины игр на случайных колодах. Из отзывов игроков, стало ясно, что основной причиной проигрыша нашего агента на случайных колодах стали специальные карты, свойство которых уникально и имеет затяжной эффект на модель стола. Такие карты не могут быть описаны

предложенным списком характеристик. Примеры таких карт приведён на рисунке 13.

Рисунок 13 – Карты «Вестник рока» и «Бесстрашный репортёр» с уникальными свойствами. Изображения карт взяты из [1].

Эффект «Вестника рока» срабатывает в начале следующего хода и зачастую агент использовал его, уничтожая свой стол. «Бесстрашный репортёр» – пример карты, сущность которой задаётся нашими характеристиками, но имеет большой уровень опасности с точки зрения стратегии, так как если оставить эту карту на столе на долгое время она «вырастет» и станет большой угрозой.

Выводы по главе 3

В данной главе были представлены результаты реализации моделей, описанных в главе 2. Также были описаны проблемы, которые не были предвидены в теоретическом описании агентов. Приведены результаты испытаний против различных оппонентов.

ЗАКЛЮЧЕНИЕ

В данной работе рассмотрен процесс построения агента для ККИ Hearthstone.

В работе были достигнуты следующие результаты:

1. Был предложен метод построения агента для игры.
2. Показаны результаты описанной модели в игре против учебного агента и людей.
3. Проведены замеры, показавшие улучшение результата по сравнению с предыдущими работами.

Для достижения этих результатов были использованы два метода машинного обучения: обучение с учителем и обучение с подкреплением. На основе этих методов были реализованы четыре модели, из которых две позволили улучшить результаты предыдущей работы. Также был предложен способ описания состояния игры в Hearthstone, метод разделения игры на две подмодели, в которых одна зависит от другой, что позволяет решить сначала более простую подзадачу, а потом задачу, включающую в себя предыдущую. Описан способ обхода дерева действий игры Hearthstone, сократить вычисления и уложиться во временные условия игры.

Таким образом, все задачи исследования выполнены, а цель достигнута.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Hearthstone [Электронный ресурс]. – Режим доступа: http://eu.battle.net/hearthstone/ru. – Заглавие с экрана. – (Дата обращения 09.06.2017).
2. 70 Million Players! [Электронный ресурс]. – Режим доступа: http://us.battle.net/hearthstone/en/blog/20720847/70-million-players-5-1-2017. – Заглавие с экрана. – (Дата обращения 09.06.2017).
3. Machine learning [Электронный ресурс]. – Режим доступа: https://en.wikipedia.org/wiki/Machine_learning. – Заглавие с экрана. – (Дата обращения 09.06.2017).
4. Using neural nets to recognize handwritten digits [Электронный ресурс]. – Режим доступа: http://neuralnetworksanddeeplearning.com/chap1.html. – Заглавие с экрана. – (Дата обращения 09.06.2017).
5. Саттон Р. С. Обучение с подкреплением / Р. С. Саттон, Э. Г. Барто.– М.: Изд-во Бином, 2014.– 400 с.– С. 12-40
6. Саттон Р. С. Обучение с подкреплением / Р. С. Саттон, Э. Г. Барто.– М.: Изд-во Бином, 2014.– 400 с.– С. 116-198
7. Markov chain [Электронный ресурс]. – Режим доступа: https://en.wikipedia.org/wiki/Markov_chain. – Заглавие с экрана. – (Дата обращения 09.06.2017).
8. I am a legend: hacking Hearthstone using statistical learning methods [Электронный ресурс]. – Режим доступа: https://cdn.elie.net/publications/i-am-a-legend-hacking-hearthstone-using-statistical-learning-methods.pdf. – Заглавие с экрана. – (Дата обращения 09.06.2017).
9. Learning Artificial Intelligence in Large-Scale Video Games. A First Case Study with Hearthstone: Heroes of WarCraft [Электронный ресурс]. – Режим доступа: http://orbi.ulg.ac.be/handle/2268/183266. – Заглавие с экрана. – (Дата обращения 15.04.2017).
10. Pierre Geurts. Extremely randomized trees. / Pierre Geurts, Damien Ernst, Louis Wehenkel // Machine Learning.– 2006.– №1.– С. 3-42.

11. Will our new robot overlords play Hearthstone with us? [Электронный ресурс]. – Режим доступа: http://cs229.stanford.edu/proj2016spr/report/037.pdf. – Заглавие с экрана. – (Дата обращения 15.04.2017).
12. A* [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/A*. – Заглавие с экрана. – (Дата обращения 09.06.2017).
13. C. B. Browne et al. "A Survey of Monte Carlo Tree Search Methods". In: IEEE Transactions on Computational Intelligence and AI in Games 4.1 (Mar. 2012), pp. 1–43. issn: 1943-068X. doi: 10.1109/TCIAIG.2012.2186810.
14. Reinforcement Learning. Policy Gradient. Marcello Restelli[Электронный ресурс]. – Режим доступа: http://home.deib.polimi.it/restelli/MyWebSite/pdf/rl7.pdf. – Заглавие с экрана. – (Дата обращения 09.05.2017).
15. Neural Networks and Deep Learning [Электронный ресурс]. – Режим доступа: http://neuralnetworksanddeeplearning.com/index.html. – Заглавие с экрана. – (Дата обращения 20.04.2017).
16. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., et al. (2013). Playing Atari with deep reinforcement learning. Technical report. Deepmind Technologies, arXiv:1312.5602 [cs.LG].
17. Blizzard Entertainment [Электронный ресурс]. – Режим доступа: http://eu.blizzard.com/en-gb/. – Заглавие с экрана. – (Дата обращения 09.06.2017).
18. Activation function [Электронный ресурс]. – Режим доступа: https://en.wikipedia.org/wiki/Activation_function. – Заглавие с экрана. – (Дата обращения 09.06.2017).

ПРИЛОЖЕНИЕ

Изображение взято из [1].