

МИНОБРНАУКИ РОССИИ
ФГБОУ ВО «СГУ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»

Кафедра теоретических основ компьютерной безопасности и криптографии

ОБУЧЕНИЕ С ПОДКРЕПЛЕНИЕМ В ВИДЕОИГРАХ

КУРСОВАЯ РАБОТА

студента 3 курса 331 группы
направления 100501 — Компьютерная безопасность
факультета КНиИТ
Окунькова Сергея Викторовича

Научный руководитель

Доцент

И. И. Слеповичев

Заведующий кафедрой

доцент, к.ф.-м.н.

М. Б. Абросимов

Саратов 2022

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	3
1 Основные понятия и определения	4
1.1 Понятия искусственного интеллекта, машинного и глубокого обу- чения	4
1.2 Структура искусственной нейронной сети	5
1.3 Обучение искусственной нейронной сети	7
1.4 Обучение с подкреплением (Reinforcement learning)	8
1.5 Оценка качества обученной модели	11
1.6 Сверточная нейронная сеть	11
2 Задачи, решаемые с помощью обучения с подкреплением	13
3 Алгоритмы RL	15
3.1 Наивный подход	15
3.2 Подход с использованием обычных моделей ИНС	15
3.3 Q-learning	16
3.4 Policy Gradient	17
3.5 Actor-critic	18
4 Практическая часть	19
4.1 Описание используемых инструментов	19
4.2 Описание среды и агента	20
4.3 Обучение с подкреплением в видеоиграх	20
4.4 Процесс обучения агента	20
4.5 Результаты обучения	20
ЗАКЛЮЧЕНИЕ	21
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	22

ВВЕДЕНИЕ

В наше время компьютеры стали очень важной частью нашей жизни. Существует огромный пласт задач, решение которых мы уже не представляем без его использования. Большую часть из этих задач сегодня можно решить с помощью классического программирования, однако другую часть задач решить с помощью него либо очень проблематично, либо вовсе не возможно. Эти задачи как правило связаны либо с искусственным интеллектом, либо с нахождением закономерности в большом объеме данных. Такие задачи решаются с помощью специальных подходов, которые в совокупности называются машинным обучением. Основное преимущество данного подхода заключается в более эффективном методе построения вычислительных систем – обучаемых систем, вместо систем, программируемых.

На сегодняшний день направление машинное обучение занимает одну из ведущих позиций в ИТ, поэтому оно также является одним из самых развивающихся, чем и обусловлено появление каждый год новых алгоритмов и задачи. Если раньше с помощью машинного обучения можно было решать только простые задачи с поиском закономерностей в данных, сейчас существует множество областей, в которых алгоритмы машинного обучения превосходят человека. Вот несколько примеров задач, которые решают с помощью машинного обучения: распознавание и классификация объектов на изображении, классификация текста, распознавание речи, выдача рекомендаций, основанных на наших предпочтениях, перевод текста с картинки, принятие решений по выдаче кредитов, прогнозирование цен на квартиры, технику и другие товары и т. д.. Данная работа посвящена рассмотрению концепции обучения с подкреплением, задачам, решаемым с помощью данного вида обучения, и алгоритмы, построенные для решения этих задач.

В отличие от классического машинного обучения, в обучении с подкреплением изначально нет никакой выборки для обучения модели. Вместо этого, обучение проводится "методом проб и ошибок": агент сам собирает данные о среде, в которой находится, и на основе этих данных и своих действий пытается наиболее эффективно выполнить свою цель, которую он определяет так же сам за счет системы наказаний и поощрений, заданной ему в самом начале.

1 Основные понятия и определения

1.1 Понятия искусственного интеллекта, машинного и глубокого обучения

Искусственный интеллект (ИИ, AI, Artificial intelligence) - это наука, описывающая создание машины или программы, способной имитировать человеческое поведение для выполнения определенной задачи и обучаться за счет полученной в результате работы информации.

Другими словами ИИ можно описать, как технологию, которая способна мыслить как человек и выполнять функции человека с большей точностью. На практике же достаточно сложно создать идеальную модель ИИ, способного грамотно выполнять заданные ей функции. Поэтому для решения большинства задач в этой сфере используют машинное обучение.

Машинное обучение (Machine learning, ML) - это метод анализа данных, который автоматизирует построение аналитической модели.[1] Данная область ИИ воплощает в себе идею того, что машина способна обрабатывать полученные данные, анализировать их и на основе анализа обучаться с минимальным вмешательством человека.

Однако данная область ИИ все же требует помощи в самом процессе обучения со стороны человека, а именно загрузку корректных данных, рассматривающих все возможные случаи, на которых машина будет обучаться. Иначе говоря, несмотря на то, что данная система обучения машины позволяет решить огромное количество задач, она не способна сама генерировать эти задачи.

Глубокое обучение (глубинное обучение, Deep learning, DL) - это тип машинного обучения, который обучает компьютер выполнять задачи человека.[1]

Данная система обучения вместо того, чтобы ждать тесты от человека, для дальнейшей их обработки по формулам, заданным изначально, сама устанавливает начальные параметры и учит машину обучаться самостоятельно.

Таким образом, если представить AI, ML и DL в виде трех множеств, то мы получим картину, изображенную на рисунке 1:

$$DL \subset ML \subset AI. \quad (1)$$

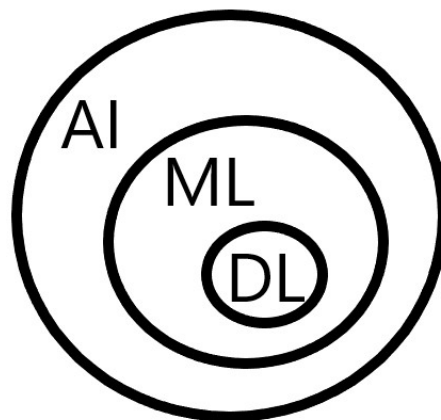


Рисунок 1 – Представление AI, ML и DL в виде множеств

1.2 Структура искусственной нейронной сети

Для решение задач в сфере машинного и глубокого обучения часто используют искусственные нейронные сети.

Искусственные нейронные сети (ИНС, artificial neural networks, ANN) - математическая модель и ее программное воплощение, созданное на основе модели биологических нейронных сетей (сетей нервных клеток организма).

ANN состоит из искусственных нейронов (ИН, artificial neuron, AN), являющихся программной реализацией нейронов живых организмов и представляющих из себя нелинейную функцию, называемую передаточной функцией, от массива входных данных ($[x_1, x_2, \dots, x_n]$). Основная функция ИН - принятие входного сигнала с последующей его обработкой и передачей результата на другие нейроны.

Из всего выше описанного очевидно, что некоторые AN связаны между собой. Как и в биологии данную связь называют синаптической связью.

Синапсом в искусственных нейронных сетях называют связь между нейронами. По нему передаются выходные связи с одного нейрона на другой. У любого синапса имеется такой параметр, как весовой коэффициент w_i , который показывает текущее состояние нейрона, а более сложные синапсы также могут иметь память. Состояние нейрона в определенный момент времени вычисляется как взвешенная сумма его входов:

$$s = \sum_{i=1}^n x_i w_i, \quad (2)$$

где n - число входов нейрона, x_i - i -й входной сигнал, w_i - вес i -го синапса.

В большинстве случаев связь между нейронами представляют в виде матрицы W , которую соответственно называют матрицей веса.

Помимо синапсов важную роль в связи нейронов между собой играют аксоны. **Аксон** - это выходная связь нейрона, с помощью которой выходной сигнал нейрона поступает на синапсы других нейронов. Его значение будет равно:

$$Y = f(s), \quad (3)$$

где s - это состояние нейрона в данный момент, а f - это передаточная функция, в качестве которой чаще всего используются сигмоид, линейный порог, гиперболический тангенс или жесткую пороговую функцию. Данная функция одинакова для нейронов одного слоя, но для нейронов разных слоев она могут выбираться разные функции.

Проанализировав все выше описанное можно представить, как в общем выглядит нейрон:

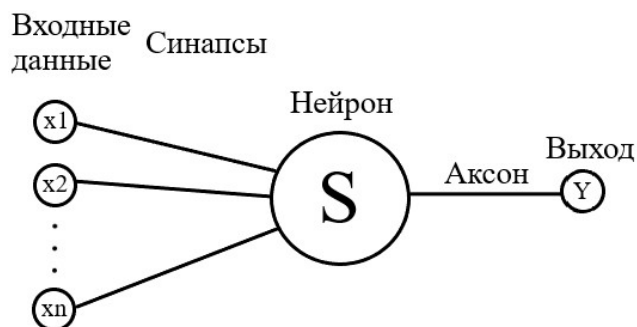


Рисунок 2 – Общий вид искусственного нейрона

Уже на данном этапе достаточно информации, чтобы сделать вывод о том, как схематически можно изобразить искусственную нейронную сеть. Пример такого изображения можно увидеть на рисунке 3.

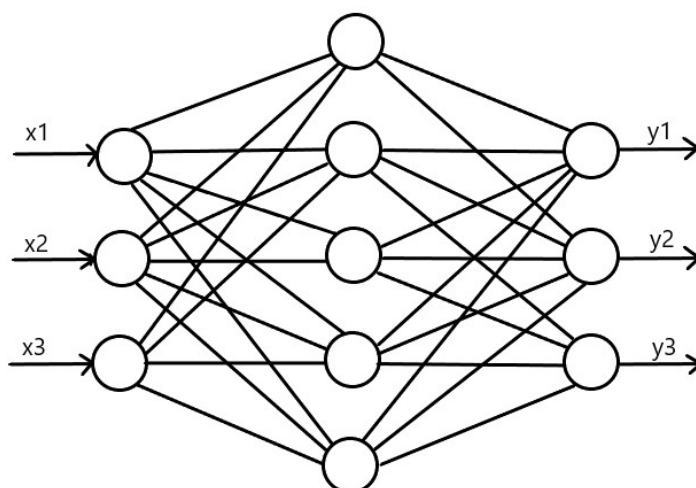


Рисунок 3 – Пример искусственной нейронной сети

1.3 Обучение искусственной нейронной сети

Как говорилось выше, для решения задачи не достаточно просто написать модель нейросети, ее еще нужно и обучить. Под процессом обучения понимается выбор таких параметров, при которых нейросеть решает поставленную задачу с большей эффективностью. Схема обучения изображена на рисунке 5. Математически его можно описать следующим образом. В процессе работы нейросеть, как было выяснено выше, формирует выходной сигнал $Y = G(X)$, являющийся реализацией какой-то функции. И пусть ответом на поставленную задачу будет функция $Y = F(X)$, заданная с помощью параметра входных-выходных данных так, что

$$Y^k = F(X^k), \quad (4)$$

где $k = 1, \dots, N$.

Обучением же будет состоять из генерации функции $G(X)$, близкой к $F(X)$. Оно будет состоять из множества итераций, на каждой из которых функция $G(X)$ будет все ближе и ближе к функции $F(X)$, а для определения степени их близости друг к другу используют некоторую функцию $D_F(G)$, которую называют функцией ошибок или целевой функцией. Каждую такую итерацию называют эпохой. Таким образом обучение ИНС будет проходить до того момента, пока функция $D_F(G)$ не примет минимальное свое значение.

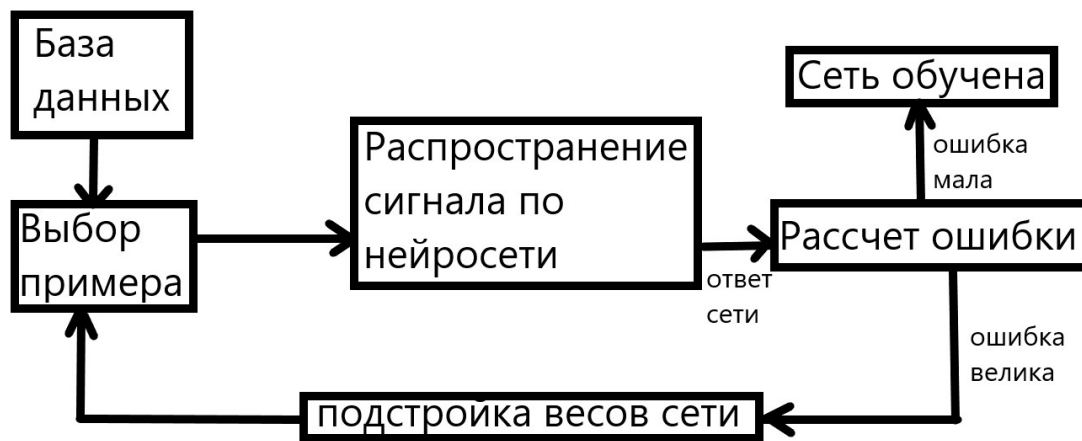


Рисунок 4 – Процесс обучения нейронной сети

Описанные выше параметры делятся на два вида:

1. Гиперпараметры - это такие параметры, которые настраивает сам человек до запуска самого процесса обучения (например количество эпох обучения).
2. Веса модели - это те параметры, которые настраиваются самой моделью при обучении. Они показывают значимость каждого нейрона в функции $G(X)$.

Одним из важнейших гиперпараметров практически любой модели является скорость обучения (learning rate). С помощью него задается скорость схождения функции $G(X)$ к функции $F(X)$ на каждой итерации.

Существует три основные стратегии обучения, используемые для решения разного вида задач:

- с учителем или по другому контролируемое обучение (Supervised Learning)- нейросети на вход подаются данные и она обучается путем обработки этих данных;
- без учителя или по другому неконтролируемое обучение (Unsupervised Learning) - нейросеть обучается в соответствии с некоторым правилом, при этом данные для обучения не требуются;
- обучение с подкреплением (Reinforcement learning) - имеет сходство с обучением с учителем, только в роли учителя выступает настоящая или виртуальная среда.

1.4 Обучение с подкреплением (Reinforcement learning)

Данный вид обучения имеет сходство с человеческим обучением: человек хочет чему-то научиться, для этого он делает какие-то действия, у кото-

рых есть некоторые последствия (как положительные, так и отрицательные), и относительно этих последствий корректирует свои действия в дальнейшем. Другими словами модель, обучаемая с помощью RL, изначально не имеет никаких сведений о среде, в которой находится, но имеет возможность выполнять определенные действия в ней, чтобы лучше понимать эту среду, за счет чего и обучается.

Таким образом фокус обучения с подкреплением делается на регламентированные процессы обучения, при которых алгоритм машинного обучения снабжен набором действий, параметров и конечных значений.[11]

Агентом называют некоторую сущность, которая выполняет определенные действия в среде. Модель же в данном случае по состоянию среды и агента в ней в данный момент прогнозирует действие, которое должен совершить агент для того, чтобы приблизиться к выполнению задачи максимально эффективно. Для достижения этой эффективности инженером изначально задается система штрафов и поощрений, т. е. при достижении определенного состояния агента и среды модель либо добавляются очки (например прохождение агентом некоторого расстояния), либо вычитаются очки (например смерть агента). Иными словами у обучения с подкреплением построено на двух основных принципах, совокупность которых была названа выше эффективностью:

1. Минимизация ошибок (например уменьшение количества столкновений с другими машинами и т. д.).
2. Максимизация выгоды, заданной заранее (например максимально быстрое время прохождения заданной дистанции, минимальное количество расходуемых ресурсов и т. д.).

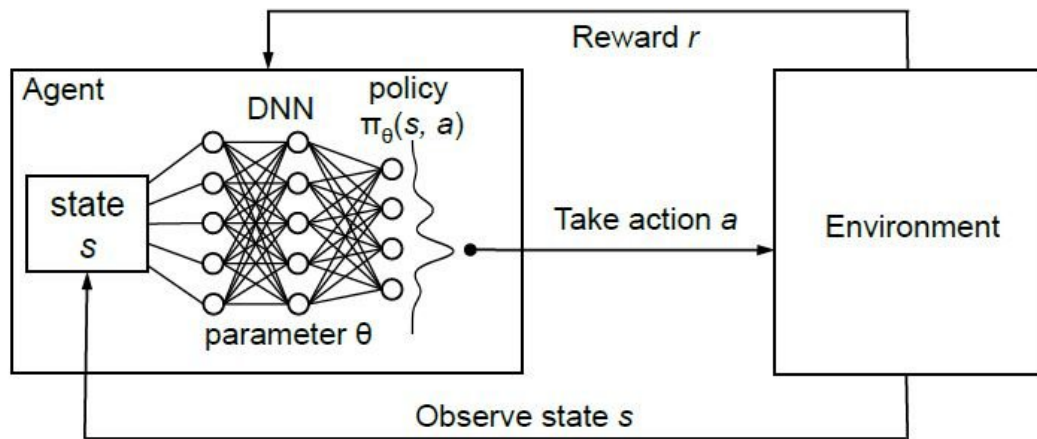


Рисунок 5 – Взаимосвязь агента и среды

Определим терминологию:

1. S_t — состояние среды (state) на шаге t ;
2. a_t — действие агента (action) на шаге t ;
3. r_t — награда (reward) на шаге t ;
4. $\pi(a_t|s_t)$ — policy, стратегия поведения агента, условная вероятность;
5. $a_t \sim \pi(\cdot|s_t)$ — action рассматриваем как случайную величину из распределения π , Мы могли бы рассматривать policy как функцию $\pi : States \rightarrow Actions$, но мы хотим сделать действия агента стохастическими, что способствует exploration. Т.е. мы с некоторой вероятностью делаем не совсем те действия, которые выбирает агент.
6. τ — траектория, пройденная агентом, последовательность (s_1, s_2, \dots, s_n) ;
7. V (value) или E (estimate) - ожидаемая итоговая (награда) со скидкой, в отличие от мгновенной награды R , является функцией политики $E^\pi(s)$ и определяется, как ожидаемая итоговая награда Политики в текущем состоянии s . (Встречается в литературе два варианта Value – значение, Estimate – оценка, что в контексте предпочтительней использовать E – оценка);
8. Q-value (Q) - оценка Q аналогична оценке V , за исключением того, что она принимает дополнительный параметр a_t . $Q^\pi(s_t, a_t)$ является итоговой оценкой политики π от состояния s_t и действия a_t . Рассчитывается с помощью уравнения Беллмана:

$$Q(s, a) = r(s, a) + \gamma \max(Q(s', a)) \quad (5)$$

Оно означает, что максимально возможное вознаграждение (Q) агента в состоянии s равно сумме моментального вознаграждения r за его шаг и максимально возможного вознаграждения агента из состояния s' помноженное на коэффициент понижения γ

Задача агента — максимизировать expected return:

$$J(\pi) = E_{\tau \sim \pi}[R(\tau)] = E_{\tau \sim \pi} \left[\sum_{t=0}^n r_t \right] \quad (6)$$

Таким образом математически задачу RL можно сформулировать как поиск такой оптимальной стратегии π^* , что $\pi^* = \arg \max_{\pi} J(\pi)$.

Таким образом можно сделать вывод, что обучение с подкреплением полностью завязано на взаимном воздействии агента и среды вне зависимости от алгоритма. О такой системе говорят, что она имеет обратную связь, поэтому ее нужно рассматривать как единое целое, из-за чего линия разделения между средой и агентом в этой системе достаточно условна. Эта взаимосвязь схематично показана на рисунке 5. Эта взаимосвязь рассматривают как последовательность пар state и reward, переходами в которые являются actions агента:

$$(s_0) \xrightarrow{a_0} (s_1, r_1) \xrightarrow{a_1} \dots \xrightarrow{a_{n-1}} (s_n, r_n) \quad (7)$$

1.5 Оценка качества обученной модели

После окончания процесса обучения необходимо проверить насколько хорошо модель обучилась. Для этого используют метрики качества обучений. Существует множество хороших метрик, которые используются для разных задач, поэтому очень важно правильно выбрать метрику для выбранной задачи.

В задачах Reinforcement learning лучшими метриками являются награда или средняя награда за эпоху, количество шагов, которые выполняет модель за эпоху, сожаление, представляющие из себя потери модели при выполнении альтернативного действия.

1.6 Сверточная нейронная сеть

Для некоторых алгоритмов обучения с подкреплением используют сверточные нейронные сети (например алгоритм PPO, который был используемый в практической части данной работы), поэтому их также стоит рассмотреть для лучшего понимания темы данной работы.

Наилучшие результаты в задачах распознавания объектов на изображениях показала **сверточная нейронная сеть (Convolutional Neural Network, CNN, СНС)**, которую называют логическим продолжением идей когнитрона и некогнитрона. Главной причиной успеха данной архитектуры является учет двумерной топологии изображения. СНН работает по принципу масштабирования (свертки) изображения.

Свертка - это операция над исходной матрицей A и матрицей свертки B , размерностью $(n_x * n_y)$ и $(m_x * m_y)$ соответственно, результатом которой является матрица $C = A * B$, размер которой $(n_x - m_x + 1 * n_y - m_y + 1)$, а элементы рассчитываются по формуле:

$$C_{i,j} = \sum_{u=0}^{m_x-1} \sum_{v=0}^{m_y-1} A_{i+u,j+v} B_{u,v}. \quad (8)$$

Свертка изображения

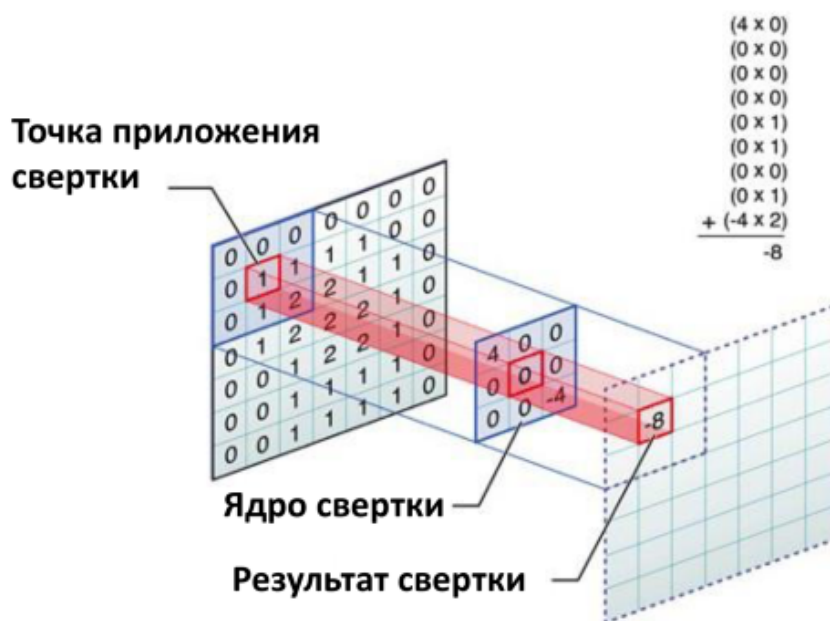


Рисунок 6 – Свертка

Архитектура данного типа ИНС строится на чередовании слоев свертки и подвыборки, которые состоят из карт признаков. Они в свою очередь отвечают за поиск определенных признаков. К примеру одна карта ищет предметы красного цвета, вторая - синего.

2 Задачи, решаемые с помощью обучения с подкреплением

Обучение с подкреплением применяется там, где принятое решение может иметь последствия не только сразу после его принятия, но и спустя некоторый промежуток времени. Поэтому алгоритмам данного вида обучения иногда приходится ждать, чтобы увидеть последствия своих решений. Обычно даже человеку в таких задачах сложно понять, какой порядок действий приводит к какому результату.

К таким задачам относятся:

1. Создание продвинутого ИИ для игр. Сейчас большинство ИИ в играх создаются на основе математических автоматов с множеством состояний и переходов. Однако этот подход является не самым совершенным, и поэтому качество такого ИИ оставляет желать лучшего. Подход же с нейросетью, обученной с помощью RL, является более совершенным. Так такие ИИ могут спокойно обыгрывать чемпионов мира по шахматам или представлять реальную угрозу даже самым умелым игрокам.
2. Обучение модели для автономного вождения. Изначально моделируется несколько различных сред для машины, которая в данном случае является агентом, а затем в них по очереди помещается сам агент для обучения на всех возможных ситуациях. Так сначала модель обучается на пустой дороге, чтобы модель понимало, что агент должен ехать только по ней по ней, затем агента запускают в среду с пешеходами и знаками дорожного движения, чтобы научить модель соблюдать ПДД, а после его начинают запускать в среды, которые представляют из себя различные специфичные ситуации: зимние дороги, непогоду, ремонтные работы, поломанные дороги и т.д. Ярким примером использования алгоритмов RL в данной сфере являются машины компании Tesla.
3. Обучение роботов в робототехнике. Схоже с предыдущим пунктом, меняются лишь цели обучения, возможности действий агента, условия самих сред и их количество в зависимости от самой задачи.
4. Чат боты, основанные на нейронных сетях. Для данной задачи существует несколько подходов. Первый из них представляет из себя создание огромного ансамбля NLP (Natural Language Processing) моделей, работающих как параллельно, так и последовательно. Второй же завязан на одной модели, обученной с подкреплением. В этом случае средой выступают диалоги

с обычным пользователем, а агентом сам чат бот.

5. Рекомендательные системы. Данный класс задач тоже можно решать с помощью RL. Тогда выдача рекомендации того или иного предложения будет являться действием в среде, а средой будет выступать совокупность того места, где будет появляться рекомендация и людей, находящихся в данной среде. Этот подход в перспективе является лучше классического двухуровневого подхода к данной задаче, однако проигрывает ему в скорости обучения, потому что в двухуровневом подходе обучение происходит перед использованием модели, за счет чего мы получаем неплохое качество рекомендаций уже после добавления такой функции, а RL моделью обучается при взаимодействии с ней пользователей.

3 Алгоритмы RL

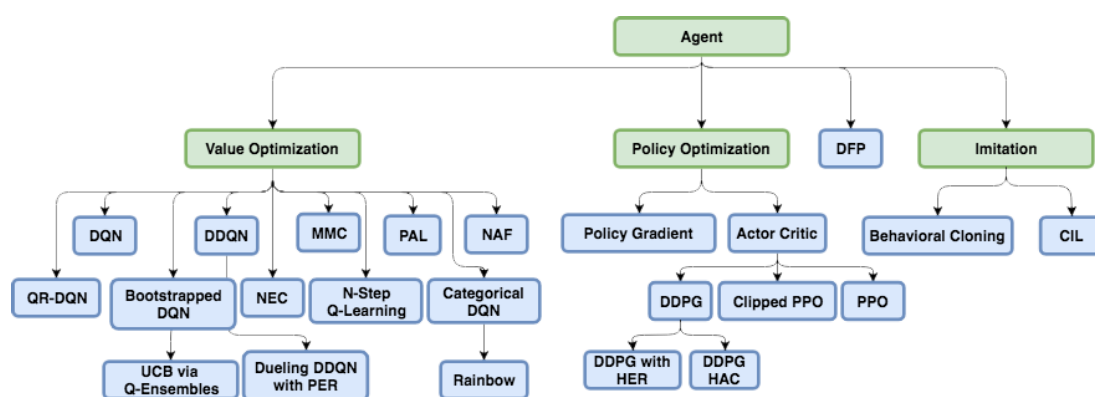


Рисунок 7 – Алгоритмы RL

3.1 Наивный подход

Самый простой подход к данному классу очень просто и не предполагает использование ИНС. Его реализация представляет собой использование динамического программирования и состоит из двух шагов:

1. Перебрать все возможные стратегии.
2. Найти самую оптимальную стратегию, дающую наибольшую награду.

Главная проблема такого подхода - это количество стратегий, которые надо обработать. Их количество может быть не просто очень велико, а бесконечно. Вторая проблема заключается в плохом обобщении такой системы. Использование ИНС хорошо решает эти проблемы, не рассматривая все стратегии, а выбирая, основываясь на своем опыте предыдущем опыте, оптимальную.

3.2 Подход с использованием обычных моделей ИНС

Данную задачу можно также попробовать решить стандартными алгоритмами, обучая их с подкреплением, однако такое решение не принесет нужный результат вне зависимости от сложности модели. Так происходит по одной простой причине, которая заключается в дисбалансе классов, вызванный случайными действиями агента в ситуациях, в которых он не знает как действовать, из-за чего некоторые действия могут вовсе не выполняться. Более простыми словами полезные сигналы теряются на фоне шума, который обусловлен редкими наградами.

3.3 Q-learning

Раньше, когда НС не существовало, а мощностей компьютера не хватало для высоконагруженных вычислений, уже существовал RL. Он выглядел как простая, но очень оригинальная идея: делать случайные действия, а потом для каждой ячейки в таблице и каждого направления движения, посчитаем по формуле уравнения Беллмана (6) насколько хороша эта ячейка и выбранное направление. Чем выше это число, тем с большей вероятностью этот путь ведет к большей выгоде. Само такое число получило название Q (от слова quality — качество выбора, очевидно), а метод — Q-learning.

-0.08	-0.04	-0.04	1.00
-0.08	-0.07	-0.04	-0.88
-0.09	-0.04	-0.03	-0.27
0.84		0.51	-0.08
-0.10	-0.10	-0.05	-0.08
-0.10	-0.10	0.08	-0.08
0.79	-0.10	-0.09	-0.08
-0.12	-0.12	-0.09	-0.08
-0.12	-0.10	-0.09	-0.08

Рисунок 8 – Пример табличного RL

Чтобы побороть проблему обычных ИНС, описанную выше ученые вернулись к классическому табличному Reinforcement Learning, но вместо ручного подсчета значений таблицы в качестве целевой функции стали использовать НС. На основе этого подхода и строятся алгоритмы RL сегодня и, он же является отличием от обычного обучения нейросетей. Классический же алгоритм, в основе которого лежит только заполнение таблицы получил название DQN (Deep Q-learning Network)[20].

В DQN на вход нейросети подается текущая ситуация (state), а на выходе нейросеть предсказывает число Q. А так как на выходе сети перечислены сразу все возможные действия (каждый со своим предсказанным Q), то получается что нейросеть в DQN реализует классическую функцию $Q(s,a)$ из Q-learning.

Следовательно для получения самого оптимального действия в заданной ситуации нужно использовать argmax , для который вернет индекс действия с максимальным Q . Такая политика будет называться детерменистской. Но более оптимальной будет стохастическая политика. При ней выбирается случайное действие из доступных, но пропорционально их Q -значениям (т.е. действия с высоким Q будут выбираться чаще, чем с низким). Такая политика является более оптимальной, так как она выбирает действия, которые в данный момент могут не нести особого смысла в данный момент, но могут в дальнейшем привести к действиям с большей наградой и меньшим штрафам.

Главный минус такого алгоритма, что он работает только с агентом, который может совершать небольшое количество детерминированных действий (при их высоком количестве метод просто не сходится). Чтобы решить данную проблему были придуманы другие алгоритмы, оптимизатором для которых стал модифицированный алгоритм градиентного спуска, получивший название Policy Gradient.

3.4 Policy Gradient

Идея алгоритма очень проста: подавать на вход НС текущее состояние, на выходе сразу предсказывать действия $\pi_\theta(a_t|s_t)$, а затем сравнивать полученную награду за действие со средней наградой. Таким образом по динамике $R(\tau) = \sum_{t=0}^T r_t$ становится возможно вычислять вектор градиент. Эта модификация позволяет свести задачу обучения RL сети к обычной задаче классификации, следовательно в качестве функции потерь можно использовать модифицированную версию кросс-энтропии:

$$loss = -\log(\pi_\theta(a_t|s_t))R(\tau). \quad (9)$$

За счет этого происходит уменьшение шума, который не давал корректно обучать классические модели НС для класса задач RL.

Приемуществом данного метода является возможность обучать модель с большим количеством действий, а также гибкая система поощрений и наказаний. К минусам же можно отнести то, что пересчет весов модели происходит только в конце эпохи, а не с каждым шагом агента.[15]

3.5 Actor-critic

Следующая модификация, которая была добавлена в алгоритмы RL, это добавление явного учителя, который представляет из себя новую модель, которая будет оценивать действия агента. Такую модель называют критиком, а агента - актером.

Актерская модель практически ничем не отличается от стандартной архитектуры агента. На вход так же подается состояние среды в данный момент и на выход выдается действия или вектор вероятностей уместности действий в данной ситуации.

Критическая модель получает на вход так же получает состояние и действие, предсказанное первой моделью, а на выход выводит значение значению Q , которое будет являться функцией от входных параметров $Q(s, a)$. Таким образом за счет оценки критика обучается актер на основе Policy Gradient, а критик будет учиться обычным путем, согласно с реальным прохождением эпизода.

Примером алгоритма использующего классическую реализацию Actor-critic является DDPG. Более продвинутые модели способны сравнивать новую политику с предыдущей и на основе этого корректировать свои веса. В этом случае для расчета градиента используется не просто функция $Q(s, a)$, а $A(s, a) = Q(s, a) - V(s)$. В данном случае функция $A(s, a)$ как раз и показывает насколько после предпринятых действий станет лучше, чем текущая ситуация $V(s)$ (в самой простой реализации $Q(s, a)$ можно заменить на r и таким образом $A = r - V(s)$). Примером таких алгоритмов являются A3C/A2C. Их главным минусом является резкое изменение весов моделей из-за чего процесс спуска становится более стохастическим, поэтому более поздние алгоритмы, такие как Proximal Policy Optimization (PPO)[15] и Trust Region Policy Optimization (TRPO)[15] имеют ограничение на изменение весов (что-то вроде gradient clipping в рекуррентных сетях для защиты от взрывающихся градиентов, только на другом математическом аппарате), один из которых и используется в практической части данной работы.

4 Практическая часть

4.1 Описание используемых инструментов

В качестве языка программирования для данной работы был выбран Python 3.9.5, потому что на сегодняшний день данный язык программирования он является самым привлекательным языком для работы с глубоким обучением, имея только одну хорошую альтернативу в лице языка R. Это связано не только с простотой работы с языком, но и с большим списком фреймворков для ML.

Чтобы проводить эксперименты с моделью и средой была использована платформа Jupyter Notebook. С помощью этой платформы создается локальный сервер, на котором можно работать с файлами с расширением `.ipynb`. Плюсом таких файлов является возможность запускать не весь код, написанный на языке Python, а лишь его части, хранящиеся в отдельных ячейках. Это очень удобно, потому что весь код можно хранить в одном файле и запускать только необходимые в данный момент его части, за счет чего становится проще отлаживать конечную программу.

Для обработки состояния среды с помощью скриншота была выбрана библиотека OpenCV. Она за счет своего широкого функционала обработки изображений любого формата часто используется в задачах компьютерного зрения. В данный функционал входит возможность накладывать фильтры на изображение, покадрово обрабатывать видеозаписи, изменять размерность изображений и многие другие полезные функции.

Также в качестве инструмента для создания среды, в которой будет обучаться модель была выбрана одна из лучших библиотек, созданных специально для RL, OpenAI Gym. Эта узкая направленность библиотеки и является ее главным плюсом. С помощью нее можно эффективно определить правила, по которым существует среда (за что агент будет получать награду, а за что штраф, как обрабатывать каждое состояние, что будет происходить, после окончания каждого эпизода, количество возможных действий агента и т.д.).

В качестве фреймворка для работы с моделью была выбрана библиотека Stable Baselines3, содержащая в себе модели RL, написанные с помощью более широкоиспользуемой библиотеки PyTorch. Поэтому плюсы библиотеки PyTorch также сохраняются в библиотеке Stable Baselines3.

Для визуализации кривых обучения, представляющих из себя метрики обучения был использован очень популярный инструмент tensorboard, созданный

специально для этого. Его плюс в том, что в отличие от популярных фреймворков визуализации `matplotlib` или `seaborn` нет необходимости писать небольшой скрипт на языке `python`, а достаточно лишь одной команды в терминале для того, чтобы создать на локальном сервере доску с визуализацией результата обучения.

4.2 Описание среды и агента

4.3 Обучение с подкреплением в видеоиграх

В видеоиграх получать информацию о среде проще всего через скриншот того, что происходит на экране. Казалось бы, в этом случае можно просто свести задачу к классической классификации изображений, однако тут появляется две серьезные проблемы. Во первых нужно собрать достаточно данных и разметить их вручную, что не является легким процессом, так как речь идет о миллионах картинках разных ситуаций для хорошего обобщения модели, а вторая проблема была описана выше - в данной задаче для корректной работы модели нужно не просто предсказать метку изображения, которое будет указывать действие, но и понимать, как это действие в будущем повлияет на конечный результат. Поэтому для данного класса задач более эффективно использовать RL алгоритмы. Однако эти алгоритмы будут иметь много сходств с алгоритмами классификации изображений. Исходя из всего этого можно сразу предположить примерную архитектуру нейронной сети для данного класса задач: первые слои будут являться слоями свертки, для более тщательного анализа картинки на признаки, далее будут присутствовать несколько скрытых слоев для обработки этих признаков, в результате работы которых будет получен вектор размерностью $n \times 1$, содержащий вероятность действий, по которому можно понять какое действие лучше всего выполнить в данный момент и на сколько вероятность этого действия выше, чем у остальных. Далее достаточно просто с помощью функции `softmax`, которая возвращает индекс самого максимального элемента в векторе, получить само действие. Главное же различие в том, что вместо заранее размеченной метки изображения за ответом корректности нашего предсказания действие поадется в среду и реакция среды на это действие и будет являться ответом.

4.4 Процесс обучения агента

4.5 Результаты обучения

ЗАКЛЮЧЕНИЕ

В данной работе были рассмотрены следующие темы:

1.

В заключение данной работы нужно отметить, что сегментация изображения является очень важным средством для решения такого пласта задач, который связан с компьютерным зрением. Так как сегментация - это первый этап необходимый для анализа изображения. Поэтому с каждым годом появляется все больше и больше новых более оптимальных алгоритмов сегментации изображения. Подтверждение чего можно было увидеть в данной работе.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1 Пять технологий искусственного интеллекта, о которых вам нужно знать [Электронный ресурс] – URL: https://www.sas.com/ru_ru/insights/articles/analytics/five-ai-technologies.html (дата обращения 27.04.2021) - Загл. с экрана. Яз. рус.
- 2 Нейронные сети для начинающих. Часть 1 [Электронный ресурс] – URL: <https://habr.com/ru/post/312450/> (дата обращения 13.05.2022) - Загл. с экрана. Яз. рус.
- 3 Нейронные сети — математический аппарат [Электронный ресурс] – URL: <https://basegroup.ru/community/articles/math> (дата обращения 13.05.2022) - Загл. с экрана. Яз. рус.
- 4 Нейронные сети, или как обучить искусственный интеллект [Электронный ресурс] – URL: <http://internetinside.ru/neyronnye-seti-ili-kak-obuchit-iskuss/> (дата обращения 13.05.2022) - Загл. с экрана. Яз. рус.
- 5 Как работает нейронная сеть: алгоритмы, обучение, функции активации и потери [Электронный ресурс] – URL: <https://neurohive.io/ru/osnovy-data-science/osnovy-nejronnyh-setej-algoritmy-obuchenie-funkcii-aktivacii-i-poteri/> (дата обращения 13.05.2022) - Загл. с экрана. Яз. рус.
- 6 Сверточная нейронная сеть, часть 1: структура, топология, функции активации и обучающее множество [Электронный ресурс] – URL: <https://habr.com/ru/post/348000/> (дата обращения 13.05.2022) - Загл. с экрана. Яз. рус.
- 7 Обучение с подкреплением в машинном обучении [Электронный ресурс] – URL: <https://evergreens.com.ua/ru/articles/reinforcement-learning.html> (дата обращения 16.05.2022) - Загл. с экрана. Яз. рус.
- 8 Как искусственный интеллект играет в «Змейку» [Электронный ресурс] – URL: <https://habr.com/ru/company/skillfactory/blog/520992/> (дата обращения 16.05.2022) - Загл. с экрана. Яз. рус.
- 9 Обучение с подкреплением [Электронный ресурс] – URL: https://neerc.ifmo.ru/wiki/index.php?title=Обучение_с_подкреплением (дата обращения 16.05.2022) - Загл. с экрана. Яз. рус.

- 10 Интуитивная основа обучения с подкреплением [Электронный ресурс] – URL: <https://nuancesprog.ru/p/13051/> (дата обращения 17.05.2022) - Загл. с экрана. Яз. рус.
- 11 Гид: алгоритмы машинного обучения и их типы [Электронный ресурс] – URL: https://www.sas.com/ru_ru/insights/articles/analytics/machine-learning-algorithms-guide.html (дата обращения 17.06.2022) - Загл. с экрана. Яз. рус.
- 12 Обучение с подкреплением: разбираем на видеоиграх [Электронный ресурс] – URL: <https://habr.com/ru/company/smileexpo/blog/428329/> (дата обращения 17.05.2022) - Загл. с экрана. Яз. рус.
- 13 Обучение нейросети с учителем, без учителя, с подкреплением — в чем отличие? Какой алгоритм лучше? [Электронный ресурс] – URL: <https://neurohive.io/ru/osnovy-data-science/obuchenie-s-uchitelem-bez-uchitelja-s-podkrepleniem/> (дата обращения 17.05.2022) - Загл. с экрана. Яз. рус.
- 14 Deep Reinforcement Learning: как научить пауков ходить [Электронный ресурс] – URL: <https://habr.com/ru/post/483078/> (дата обращения 17.05.2022) - Загл. с экрана. Яз. рус.
- 15 Intro to Policy Optimization [Электронный ресурс] – URL: https://spinningup.openai.com/en/latest/spinningup/rl_intro3.html (дата обращения 17.05.2022) - Загл. с экрана. Яз. англ.
- 16 Policy Gradient Algorithms [Электронный ресурс] – URL: <https://lilianweng.github.io/posts/2018-04-08-policy-gradient/> (дата обращения 17.05.2022) - Загл. с экрана. Яз. англ.
- 17 СРАВНИТЕЛЬНЫЙ АНАЛИЗ АЛГОРИТМОВ PROXIMAL POLICY OPTIMIZATION И SOFT-ACTOR-CRITIC [Научная статья] – URL: <https://cyberleninka.ru/article/n/sravnitelnyy-analiz-algoritmov-proximal-policy-optimization-i-soft-actor-critic/viewer> (дата обращения 17.05.2022) Яз. рус.
- 18 Учебное пособие по оптимизации проксимальных политик [Электронный ресурс] – URL: <https://www.machinelearningmastery.ru/proximal-policy-optimization-tutorial-part-1-actor-critic-method-d53f9affbf6/> (дата обращения 17.05.2022) - Загл. с экрана. Яз. рус.

- 19 Что не так с обучением с подкреплением (Reinforcement Learning)? [Электронный ресурс] – URL: <https://habr.com/ru/post/437020/> (дата обращения 17.05.2022) - Загл. с экрана. Яз. рус.
- 20 REINFORCEMENT LEARNING (DQN) TUTORIAL [Электронный ресурс] – URL: https://pytorch.org/tutorials/intermediate/reinforcement_q_learning.html (дата обращения 17.05.2022) - Загл. с экрана. Яз. англ.