Слайд 1:  
Доброе утро. Сегодня я бы хотел рассказать Вам про реализацию нейросетевого алгоритма поиска пути в лабиринте. Это уже третий доклад, связанный с лабиринтами. Популярность данной задачи связана с тем, что её можно свести ко многим прикладным физическим задачам.

Слайд 2:  
В своём выступлении я немного расскажу про теоретическую часть проделанной работы, затем перейду к постановке задачи. Основная часть будет посвящена программной реализации. Под конец подведу итог работы: что было получено и расскажу про интересные направления развития данной работы.

Слайд 3:

Современные задачи науки и техники столкнулись с проблемой, что затраты по решению задач с большим объемом данных слишком высоки. Это проявлялось и в затрачиваемой памяти, и во времени выполнения программ. Сейчас происходит бурное развитие методов с помощью машинного обучения. Характерной особенностью данных методов является способность компьютеров “учиться”.

Слайд 4:

Существует 2 концепции машинного обучения:  
1) Обучение с учителем; Агент обучается производить определённые действия на основании предварительно подготовленных выборок.

2) Обучение без учителя. Агент самостоятельно формирует стратегию поведения, опираясь на изменения, производимые его действиями.

Слайд 5:

Частным случаем обучения с учителем является обучение с подкреплением. Агент при взаимодействии со средой получает очки и переходит в новое состояние, исходя из нового состояния агент принимает решение как ему действовать дальше. Цель агента - в конечном итоге получить максимально возможное число очков с помощью метода “проб и ошибок”.

Слайд 6:  
Одним из методов обучения с подкреплением является Q-learning. Метод был предложен Кристофером Уоткинсом в 1989 году. В методе вводится функция от состояния и действия агента Q(s,a), областью значений которой является вознаграждение агента. На слайде представлена формула вычисления нового значения Q. R - вознаграждение. Альфа и гамма параметры Q-learning.

Слайд 7:

Мой коллега связал задачу поиска пути в лабиринте с перемещением дрона из одной точки пространства в другую. Агентом является дрон (БПЛА), а средой - окружающее его пространство. Агенту доступно 8 действий. 6 действий связаны с перемещением, а два других с действием агента с объекта.

Слайд 8:

Программная реализация состоит из трех частей.

1. Создание среды.

2. Основной алгоритм: Q-learning.

3. Исследование алгоритма.

Используемые инструменты:

1. Python,

2. gym от OpenAI,

3. Вспомогательные библиотеки.

Слайд 9:

Для создания среды было задействовано 2 файла. 1) вспомогательный - map\_generation.py. 2) основной - labyrinth.py.

Слайд 10:

Начнем с map\_generation.py. С помощью символов представленных на экране создается символьного поле, целью данного символьного поля провизуализировать как перемещается агент в среде. При создании также создаются пункты назначения для объекта.

Слайд 11:

Теперь перейдем к основному файлу создани среды. В нем происходит вызов функций из первого файла и реализован алгоритм создания среды. Прежде всего для модели с учетом высоты слоя вводится структура данных “ключ-значение” в зависимости от номера слоя. Чтобы в дальнейшем было легче улучшать программу, был введен трехмерный массив, который отвечал за вознаграждение в каждой клетке. Это позволит учитывать сопротивление ветра, плотность, давление воздуха в каждой точке. Также представлены дополнительные функции работы алгоритма. Функция отрисовка хранится тамже.

Слайд 12:  
Среда - это параллелепипед. Он задается тремя параметрами. Если подсчитать количество возможных состояний в среде, то получится следующее. 5 характеризует состояние объекта (в точках сбора + у агента), а 4 - количество точек сбора. Дальше в программе идет заполнение первичной таблицы (матрицы 2D) вознаграждения P. Таблица заполняется в шестивложенном цикле:

1. 3 пространственных параметра (lay, row,column),

2. 2 по состояниям объекта и пунктов назначения,

3. 1 по возможным действиям.

Слайд 13:  
Можно выбрать какое-то случайное число, допустим 442, и посмотреть как выглядит матрица P после заполнения. (action, [(probability, nextstate, reward, done)]). Значения 0-7 - возможные действия агента. done - характеризует, удалось ли выполнить поставленную задачу.

Слайд 14:  
Примерно так выглядит перемещение агента в одном из эпизодов.

Слайд 15:

Теперь перейдем к основному алгоритму. Здесь тоже представлено 2 файла. discrete.py - это вспомогательный файл от OpenAI. В нем представлено 2 функции для обновления среды и для продвижения развития среды на один шаг.

Слайд 16:  
Функции выглядят примерно так.

Слайд 17:

env - это сердце OpenAI. Создается с помощью библиотеки gym. Необходимо только создать среду самостоятельно. В алгоритме объявляются следующие необходимые параметры.

Слайд 18:

Сам алгоритм выглядит следующим образом.

Слайд 19:

В моей работе было исследовано 2 модели. Модель без физических параметров и модель с физическими параметрами. Рассмотрим 1 случай.

Слайд 20:

Зафиксируем gamma с каким-то значением. Альфа положим нулем и будем варьировать число эпизодов обучения. Нужно заметить, что логично ожидать, что при увеличении N ситуация с обучением может стать лучше. Но нет. Из графиков видно, что никакое увеличение N не дает желанного результата.

Слайд 21:

Теперь зафиксируем какое-то N и будем варьировать alpha. Со временем картина станет улучшаться, пока не выйдет на какое-то насыщение. При насыщении агент получает примерно 39 очков (из 50 возможных), а при alpha = 0, вознаграждение равнялось примерно -1000. Существует такое значение alpha, при котором картина начинает ухудшаться. Я назвал это критическим значением альфа и получил, что оно равно 1.45.

Слайд 22:

Проводя аналогичные эксперименты с варьированием gamma, было получены следующие результаты. Также здесь представлено значение для критического гамма. Видно, что картина улучшается при увеличении гамма.

Слайд 23:

Теперь перейдем ко второй модели, при которой учитывается высота слоя. Фиксируем опять же число эпизодов и гамма и варьируем альфа.

Слайд 24:

Можно заметить, что картина заметно хуже по сравнению с предыдущей модели при N=5000. Если увеличить N до 50000, то результат будет намного лучше. Это связано с тем, что модель усложнилась, поэтому агнету необходимо больше времени, что “обучиться”.

Слайд 25:  
Будем варьировать гамма, но уже при N=50000. Опять же наблюдаем, что существует такое значение гамма, при котором происходит насыщение.

Слайд 26:

Под конец хочу продемонстрировать неудачные результаты экспериментов.

Слева показаны неудачные эксперименты, а справа удачные. Слева учет слоя с шагом по вознаграждению 1, а справа 0.5. Вот так вот сильно влияет подбор очков вознаграждения при обучении.

Слайд 27:

В результате исследования, получилось реализовать Q-learning алгоритм и исследовать его поведения в двух моделях. Я получил следующие оптимальные параметры при обучении.

Слайд 28:  
Здесь представлены интересные направления развития данной работы.

Слайд 29: Благодарности