Описание проекта «Snake Game»

Моя работа над данным проектом состояла из следующих шагов:

1. Оформление запроса к chat-GPT для написания среды для игры в змейку и изучение данного кода.
2. Составление шаблона финального кода на основе полученного от chat-GPT, а именно оформление шаблона для написания классов агентов и ключевой функции start\_games, которая бы запускала игры с параллельным обучением.
3. Заполнение класса Random\_Agent
4. Изучение дополнительных материалов про Q-learning. Заполнение класса Q\_learning\_Agent и анализ ошибок посредством просмотра пошагового выполнения алгоритма.
5. После анализа ошибок и изменения системы наград тестирование агента
6. Заполнение класса Logic\_Agent на основе Q\_learning\_Agent. Создание логичных эвристик для действий змейки. Тестирование агента.
7. Корректировка кода для сохранения важных характеристик агентов.
8. Сравнение агентов по их характеристикам и создание гистограмм этих характеристик (например, счета за игру).
9. Окончательное оформление кода для более приятного и понятного его чтения.
10. Составление отчета

Далее я бы хотел рассказать про каждый пункт данной работы отдельно.

Оформление запроса к chat-GPT не потребовало большого времени. Больше времени потребовалось на изучение его кода для использования в своем скрипте.

Затем я обдумал возможные имплементации агентов. Помимо случайного, я подумал о простых логичных правилах для змейки: например, иди к еде и уворачивайся от преград. Также я прочитал об использовании в качестве обучения с подкреплением q-learning и DeepQNetwork. Q-learning показался проще, чем DQN, но достаточно эффективным для не сложной игры. Поэтому выбор остановился на трех агентах: случайный, «логичный» и q-learning. Шаблон для каждого агента состоял из трех функций: инициализации, get\_next\_action и get\_next\_location, как в одном из разборов.

Функция start\_games запускает цикл игр для выбранного агента. Внутри каждой игры, пока не сыграем положенное число игр:

* 1. Инициализируется стартовое положение змейки, еды и состояние среды
  2. Для каждого шага, пока не умер:
     1. Агент для текущего состояния среды выбирает действие и его совершает
     2. Происходит обновление состояния среды
     3. Происходит обновление q-матрицы, если агент обучается

1. Случайный агент выбирает одно движение случайно из трех: вперед, направо и влево.
2. Реализация состояний сред, параметров и функции Беллмана для Q-learning аналогично источникам Интернета( https://8thlight.com/insights/qlearning-teaching-ai-to-play-snake ) и разборам из Youtube-каналов (один из них: https://www.youtube.com/watch?v=je0DdS0oIZk)

Параметры обучения и наград не позволяли моему агенту качественно обучаться. Для выявления ошибок я выводил процесс обучения в текстовом формате, что пошагово делала змейка и как обучалась. Пример игры 500 из 600 игр обучения представлен в файле q\_learning\_agent\_game\_500.

Проблема оказалась в том, что змейка поощрялась очень редко, только за поедание яблока, и из-за частого хода explore весь процесс обучения сбивался. Змейка забывала о награде. Поэтому понабилось ввести награды за ход приближения к яблоку (+1) и за ход удаления от него (-1). Также коэффициент случайного хода был выбран минимальным, чтобы змейка не забывала о наградах. После этого змейка стала обучаться в разы лучше. Также я ужесточил наказание за смерть с -10 до -100, чтобы змейка училась обходить преграды. Награда за яблоко осталась намного меньше, чем за смерть, +10. Также нужно было уменьшить коэффициент дисконтирования будущей выгоды: из-за того, что змейка видит только на одну клетку вокруг себя, угрозы будущей смерти ей кажутся даже там, где их нет, и она начинала кружиться по кругу.

После этих поправок змейка дошла до среднего счета после 600 игр до 58, что хорошо. Почему я считаю этот результат удачным, я объясню позже.

Правила для «логичного» агента были вписаны в его q-матрицу, которая не обновлялась. Тогда для каждого состояния среды змейка действовала так, как я ей указал в q-матрице. Ей также были указаны следующие награды: за смерть -100 , за яблоко +10, за ход ближе к яблоку +1 и за ход дальше от яблока -1.

В среднем на 100 играх змейка набирала 57 очков.

1. Помимо общего числа шагов за игру и числа шагов до первого яблока, я выбрал для сравнения следующие характеристики:

- счет за игру (наиболее интуитивно)

- среднее число шагов до еды

Чем больше счет за игру, тем агент лучше. Чем среднее число шагов до еды меньше, тем агент лучше **при прочих равных**, так как знает, как быстрее дойти до еды.

Поэтому нужно было сохранять эти результаты во время игр агентов.

Автор обучающего видео <https://www.youtube.com/watch?v=je0DdS0oIZk> считает свои результаты хорошими и примерными для обучающего видео. Согласно его статистике, его средний счет был примерно 44, в то время как мои logic\_agent и q\_learning\_agent выдают средний счет 58.

Помимо этого, написанный мною q-агент, благодаря наградам за ход в нужную сторону, обучается намного быстрее, за 600 игр, вместо 7500 игр в обучающем видео.

Два моих агента лучше случайного, так как у него средний счет за игру около нуля. Так как большее число шагов за игру говорит о большей выживаемости агента, два моих агента также лучше случайного. Помимо этого, два моих агента быстрее доходят как до первого яблока, так и вообще до яблока в течение игры. Эти факторы говорят о том, что мои два агента играют значимо лучше случайного агента и q-агента автора из обучающего видео.

Q-агент и логичный агент показывают очень схожие результаты: это абсолютно не удивительно, так как у них одинаковые системы наград. Однако q-агент иногда совершает случайные ходы и смотрит немного на будущие шаги через уравнение Беллмана, в отличие от логичного агента. По этой причине по последним трем характеристикам их результаты статистически значимо отличаются, хотя и совсем немного.

Для воспроизводимости результатов было зафиксировано ядро 42.

1. В конце работы к коду были добавлены описания функций и комментарии для более понятного и четкого чтения.