Задача: Создать агента, способного пройти игру сапер.

Описание игры:
Сапер является всемирно известной игрой, где от игрока требуется открыть все клетки поля на которых нет мин. На рисунке 1 представлено стандартное игровое поле после проигрыша. В процессе игры человек выбирает клетки, в которых предположительно нет мин. После выбора клетки, она открывается либо числом(на рисунке числа 1, 2, 3 или не объемная серая клетка, что эквивалентно 0), которое означает количество мин вокруг данной клетки, либо бомбой, что означает игрок проиграл.

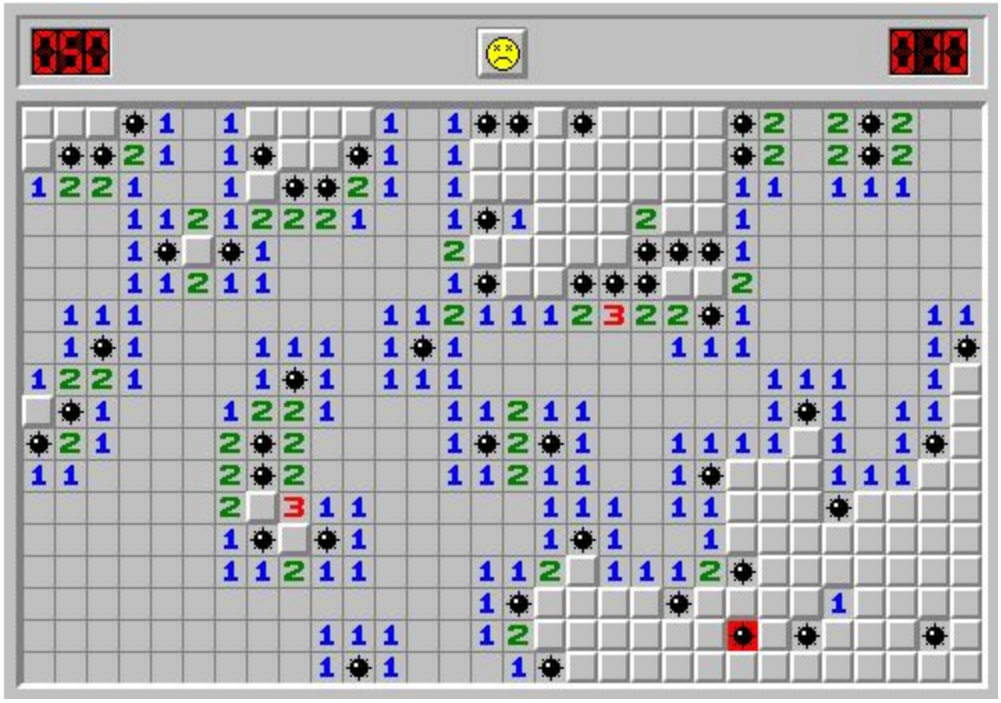

Рисунок 1 - Игровое поле игры сапер.

Подходы к решению:
1. Генетические алгоритмы
2. Обучение с подкреплением

1. Генетические алгоритмы.

В данном случае работа генетических алгоритмов заключалась в следующем: создавалась популяция ботов, каждый из которых играл в сапера, по результатам игры выбирались несколько ботов, набравших максимальное количество очков, которые в результате скрещивания и мутаций формировали новую популяцию.

С помощью генетических алгоритмов не получилось добиться универсального агента для победы в сапере, по крайней мере теми методами, которые использовал я. Поэтому коротко опишу что пробовал сделать.

Поскольку хотелось сделать агента способного играть на поле без привязки к его размеру, то решил прогнозировать для каждой открытой клетки поля возможность хода в восемь ее окружающих клеток (рисунок 2). Потом складывать предсказания для каждого возможного хода по полю и усреднять значения. Результат предсказания для каждого возможного хода относительно клетки получался в результате матричного умножения одномерного вектора длины n, который представлял состояния среды вокруг выбранной точки, и матрицы (n x 8) бота из популяции. Информация о состоянии среды получалась из игрового поля в каждой точке которого записывалось значение от 0 до 8, если точка была открыта, и 9, если точка закрыта. В качестве позиций вектора состояния среды пробовал записывать различные комбинации данных: ohe или обычное значение текущей точки, ohe или обычное представление значения восьми окружающих точек, ohe или обычное значение количества неоткрытых окружающих точек, ohe или обычное значение количества открытых окружающих точек, суммарное значение окружающих точек и т.п. Также пробовал производить аналогичные операции для каждой не открытой клетки. Пробовал объединить эти два подхода, путем создания двух популяций.

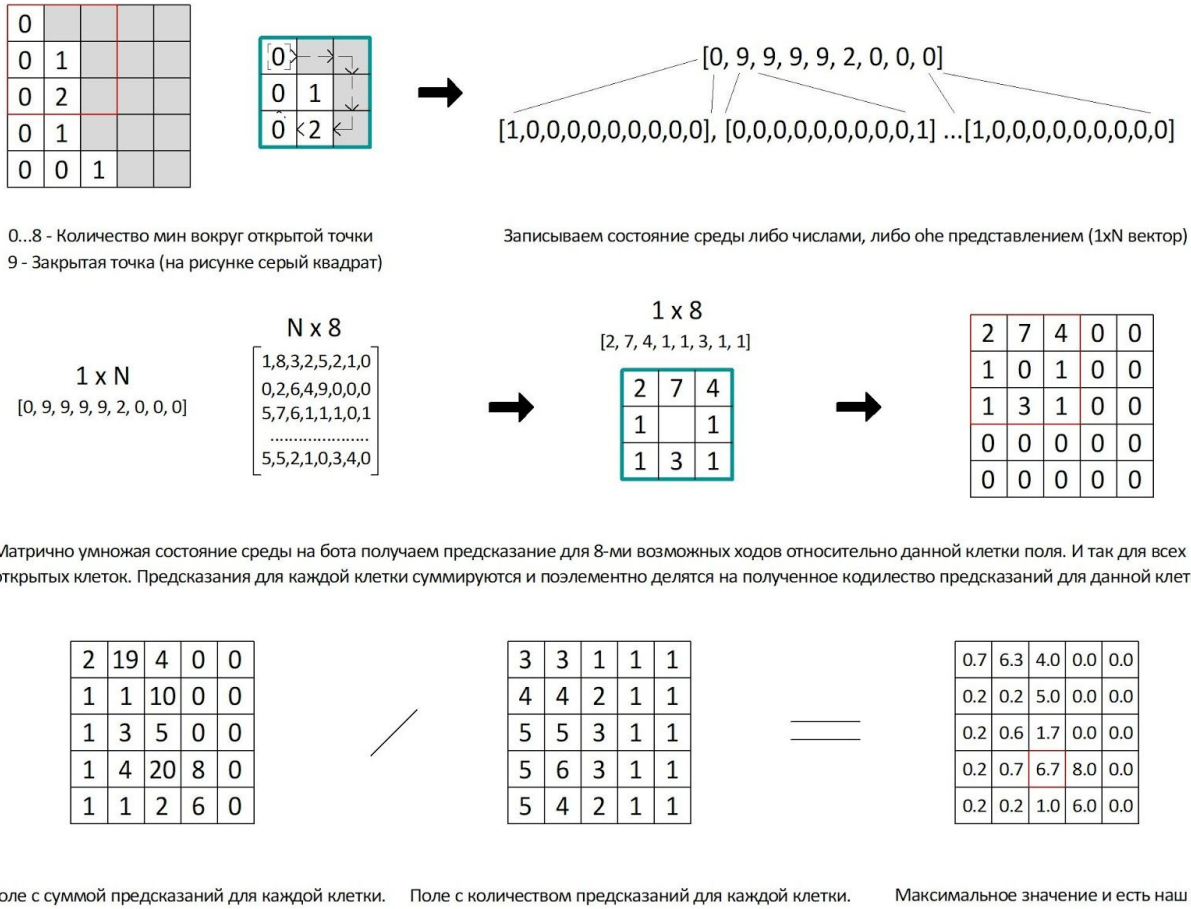

Рисунок 2 - Схема для игры в сапера ботом.

Награды для бота в процессе игры были примерно следующие: 100 - прохождение игры, 1 - за открытие новой клетки, -1 за ход в уже открытую клетку, -100 за открытие клетки с миной (данные значения постоянно менялись в результате эксперимента). Соответственно в результате игры боты набирали очки и сортировались в зависимости от их количества для формирования новой популяции.
Сначала для каждого бота генерировалось новое игровое поле, но поскольку генетика выбирает лучших ботов уже после того как вся популяция сыграла свои партии (пока это писал возникла идея выбирать ботов после каждого хода), то пришло осознание несостоятельности подобной идеи, так как боты сравнивались для совершенно разных задач. Тогда продолжил тренировать каждую популяцию на одинаковом игровом поле. Более того при тренировке на одном игровом поле в результате нескольких эпох генетический алгоритм способен подобрать бота который в состоянии пройти конкретное поле, но при изменении поля проигрывает.

Ссылка на ноутбук с генетикой (он может быть грязноват, так как я этими экспериментами занимался давно и забросил, так что лучше не смотреть :)) -
https://colab.research.google.com/drive/1EzGcsC_l6L45D-FkHcEIzypnIpwrMAFT?usp=sharing

2. Обучение с подкреплением.

Для обучения с подкреплением был выбран алгоритм Q-learning. Он основан на уравнении Беллмана:

$$Q(s,a) = r + \gamma max_{a'} Q(s',a'), \quad (1)$$

которое означает, что значение Q (награда за действие) для определенной пары состояние-действие (*s* и *a* соответственно) должно быть наградой, полученной при переходе в новое состояние (путем выполнения этого действия - r), добавленной к значению наилучшего действия в следующем состоянии ($max_{a'} Q(s',a')$ награда за наилучшее действие в будущем; *s'*, *a'* - будущее состояние и действие соответственно). Число γ определяет насколько ценна для нас информация о награде за будущее действие. Для сапера эксперименты показали, что ценной является точное предсказание награды именно для данного хода, поэтому значение $\gamma = 0$, и уравнение принимает вид:

$$Q(s,a) = r. \quad (2)$$

То есть задача обучить агента предсказывать точное значение для каждой клетки на игровом поле.

Далее я опишу последовательность экспериментов.

2.1 Агент для игры в сапера на поле 8x8 с 10 минами.
Данная конфигурация поля является уровнем “новичек”. Для решения задачи был создан агент на основе нейросети, которая на вход принимает тензор размером 8x8x10, где 8x8 - размер игрового поля, 10 - ohe представление каждой клетки (0...8 - значения для

открытых клеток и 9 - для закрытой). Одна из архитектура сети представлена на рисунке 3. В ходе экспериментов было проверено несколько вариаций архитектур, но поскольку для того чтобы можно было судить о качестве агента необходима длительная тренировка, сильно много я не экспериментировал (Единственное что сейчас подумал, что с dense выходом не попробовал). Проверенные архитектуры сильного различия в результатах не показали.

```
input = Input(shape=(8,8,10))

x1 = Conv2D(32, (5,5), padding='same', activation='tanh')(input)
x1 = Conv2D(32, (3,3), padding='same', activation='tanh')(x1)

x2 = Conv2D(32, (3,3), padding='same', activation='tanh')(input)
x = concatenate([x1, x2])
x = Conv2D(64, (3,3), padding='same', activation='linear')(x)
output = Conv2D(1, (1,1), padding='same', activation='linear')(x)

model = Model(input, output)
model.summary()
```

```
Model: "functional_1"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            [(None, 8, 8, 10)]   0
__________________________________________________________________________________________________
conv2d (Conv2D)                 (None, 8, 8, 32)     8032        input_1[0][0]
__________________________________________________________________________________________________
conv2d_1 (Conv2D)               (None, 8, 8, 32)     9248        conv2d[0][0]
__________________________________________________________________________________________________
conv2d_2 (Conv2D)               (None, 8, 8, 32)     2912        input_1[0][0]
__________________________________________________________________________________________________
concatenate (Concatenate)       (None, 8, 8, 64)     0           conv2d_1[0][0]
                                                                 conv2d_2[0][0]
__________________________________________________________________________________________________
conv2d_3 (Conv2D)               (None, 8, 8, 64)     36928       concatenate[0][0]
__________________________________________________________________________________________________
conv2d_4 (Conv2D)               (None, 8, 8, 1)      65          conv2d_3[0][0]
==================================================================================================
Total params: 57,185
Trainable params: 57,185
Non-trainable params: 0
```

Рисунок 3 - Архитектура сети для поля 8x8.

Награды за ход для агента были следующими: 1 - открытие всех клеток, 0,9 - открытие новой клетки, -0,3 - выбор уже открытой клетки, -1 - выбор клетки с миной. Также для того чтобы агент выбирал клетку поблизости с уже открытыми, добавлен штраф -0,3 за выбор клетки, если в ее окружении нет открытой.

Для выбора следующего хода использовалось максимальное значение предсказания нейросети. На начальных этапах также использовалось стратегия исследования агентом среды, путем случайного выбора действия из всех возможных (64 для поля 8x8).

В процессе обучения были добавлены улучшения процесса тренировки:

1. Нет необходимости обучать агента открывать связанные клетки с нулями самостоятельно, с этим он справляется хорошо, однако примеры для обучения засоряются большим количеством не информативных примеров.
2. На поздних этапах обучения для исследования среды есть смысл случайно выбирать ход только из закрытых клеток. Также дополнительно добавлять в базу для обучения примеры возможных ходов после проигрыша агента.

3. Для более быстрого обучения можно увеличивать плотность мин на поле (если 8x8x10 сделать 8x8x11).

Путем подобной тренировки удалось достичь доли выигрышей 32% для 1000 игр.

2.2 Агент для игры в сапера на поле с произвольными размерами

После достижения хоть каких-то результатов на поле 8x8 возникла идея обучить агента для игры на минимально возможном поле чтобы путем сканирования поля таким агентом суммировать и усреднять значения для всего поля и тем самым находить следующий ход (рисунок 4). Минимально возможным полем был выбран размер 5x5 потому что на нем возможно получение большой вариативности ситуаций и плотности мин. Архитектура сети аналогичная той, что на рисунке 3.

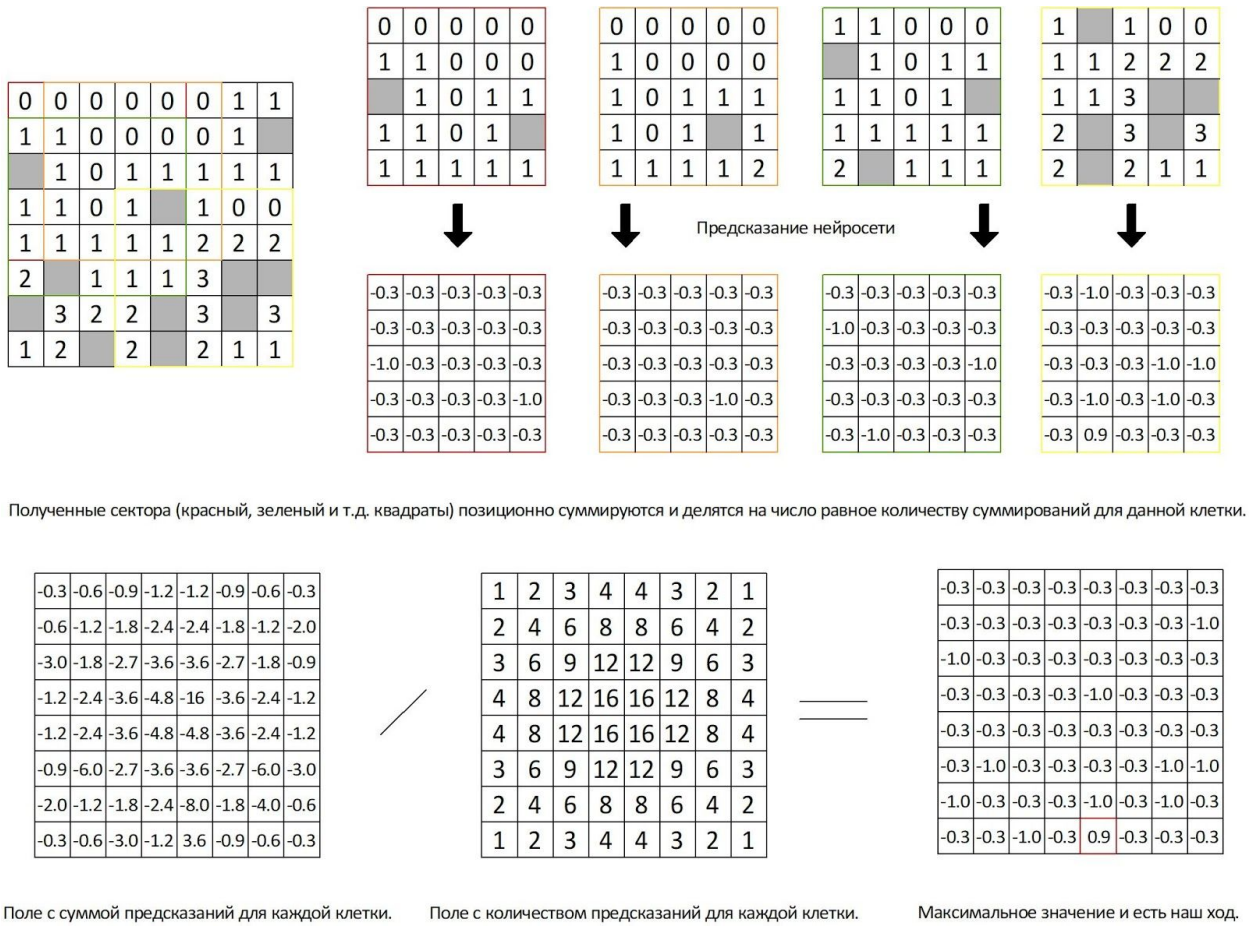

Рисунок 4 - Принцип предсказания хода для игры с произвольным полем.

Сначала обучение проходило на поле 8x8 и каждый игровой сектор размером 5x5, на котором сеть предсказывала значения, заносился в обучающую выборку. Награда за ход назначалась как результат фейкового хода в точку соответствующую полю 8x8. Данный подход показал плохие результаты. Оно и понятно, так как если брать сектор 5x5

вырванный из контекста поля 8x8 и делать на нем предикт, то полученные предсказания не учитывают ситуацию за границами сектора, а она может кардинально менять решение. При таком обучении винрейт на поле 8x8 с 10 минами был в пределах 10%.
Тогда было решено обучать на поле размером 5x5x5, чтобы все ситуации имели однозначное решение. С улучшениями процесса обучения описанными в пункте 2.1 при такой конфигурации поля удалось достичь винрейта 52%. На поле 8x8x10 винрейт 33%.
Дальше была идея обучить агента с базовым полем 7x7x8, так как тогда вариативность состояний при обучении вырастет и можно повысить плотность мин. Агент обучился до винрейта 40% (но здесь была другая архитектура нейросети). И на поле 8x8x10 показал 24% винрейт.
Также была идея, поскольку предсказание сектора вырванного из контекста на границах имеет сомнительное значение, то их искусственно понижать поэлементным делением на уравнивающую матрицу (рисунок 5). Но данное предположение не сработало :).

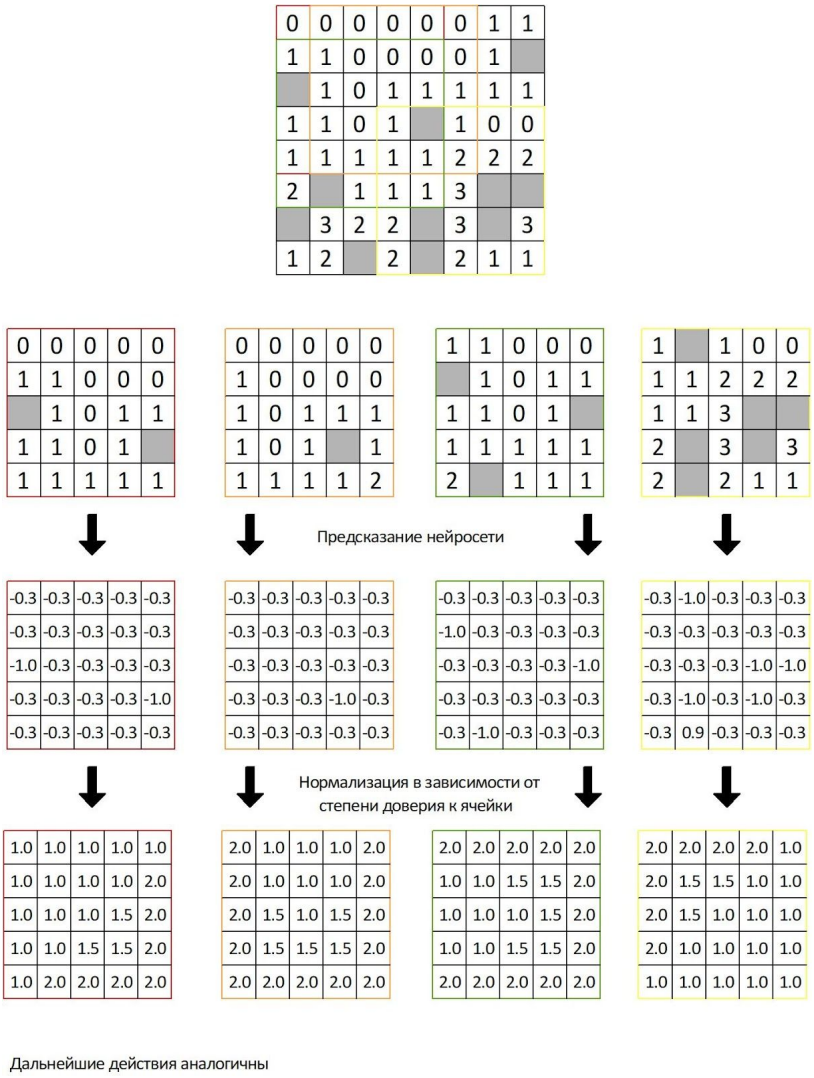

Рисунок 5 - Понижение значений сомнительных предсказаний.

Ссылка на ноутбук с обучением (8x8x10) - https://colab.research.google.com/drive/1dPH8G2wSN7wAnFTS0cENf9mW6ychtWYM?usp=sharing

Ссылка на ноутбук с обучением (5x5x5) для агента с произвольным размером поля - https://colab.research.google.com/drive/1R3lJSJvqcZka6L6LGT7mFxZ61q1vtt0l?usp=sharing

Ссылка на ноутбук с обучением (7x7x8) для агента с произвольным размером поля - https://colab.research.google.com/drive/1oQWC7brYGFf7jt5Efx4pz4LLKSvQUVQM?usp=sharing

Ссылка на ноутбук с результатами (весит 60 мб, может долго грузится) - https://colab.research.google.com/drive/1fhxHf4FbgXv1_BE2U8MYxkRjGzENskwd?usp=sharing