Q-learning в игре Flappy bird

Ларин Иван Олегович
СУНЦ МГУ
Москва, Россия

Содержание

I. Аннотация

Данная работа представляет агента, который сможет играть в игру Flappy bird лучше, чем человек. Текущее состояние агент получает из среды, которое содержит много различных параметров.

Результаты: агент получает в среднем 600 очков и рекорд - 7 000.

II. Abstract

This work represents an agent who can play the Flappy bird game better than a human. The agent gets the current state from the environment, which contains many different parameters.

Results: the agent gets an average of 600 points and a record of 7 000.

Index Terms—reinforcement learning, q-learning, replay buffer

III. Введение

В данной работе агент использует Q-learning для обучения в игре Flappy bird. Q-learning - это алгоритм обучения с подкреплением, который позволяет агенту научиться принимать решения в среде на основе получаемой награды.

Flappy bird - это простая игра, в которой игрок должен управлять птицей, преодолевая препятствия в виде труб. Цель игры - продержаться как можно дольше, не столкнувшись с трубами.

IV. Обучение с подкреплением

Обучение с подкреплением - это метод машинного обучения, в котором агент обучается принимать решения в среде на основе получаемой награды. В отличие от обучения с учителем, где алгоритм получает данные, размеченные экспертами, в обучении с подкреплением агент получает только награду за свои действия и должен самостоятельно научиться принимать решения на основе этой награды.

Формально это можно описать так: в какой-то момент времени агент находится в состоянии S, выбирает действие a, которое доступно в S, делает его, переходит в состояние S' и получает награду r.

Агент должен научиться так выбирать ходы, чтобы суммарная награда была максимальна.

V. Q-learning

Q-learning - один из алгоритмов обучения с подкреплением. Он заключается в создании таблицы (Q-table) $Q(S, a)$, по которой агент будет выбирать a в зависимости от S.

$Q(S, a)$ - возможная награда, умноженная на коэффициенты, которую агент сможет получить начиная с состояния S, сделав действие a.

Изначально таблица заполнена нулями или случайными числами. Она обновляется в какие-то моменты.

Было состояние S, агент сделал действие a, попал в состояние S', за что получил награду r. Тогда пересчитать Q-value можно следующим образом:

$$Q(S,a) \leftarrow Q(S,a) + \alpha \left(r + \gamma \max_{a' \in A(S')} Q(S',a') - Q(S,a) \right)$$

где $A(S')$ - множество возможных ходов из состояния S', α - скорость обучения (learning rate), γ - коэффициент, понижающий влияние следующих ходов на текущее $Q(S,a)$ (discount factor).

Если агент всегда будет ходить только по таблички, то некоторые состояния не будут исследованы. Чтобы избежать этого введем новую переменную - ε, которая изначально равна 1. После каждой смерти будем умножать её на какое-то число или вычитать маленькое число. Теперь когда агент выбирает какое действие сделать будем с шансом ε делать случайное действие.

Replay buffer

Replay buffer - массив из $bufferSize$ последних действий, которые задаются (S, a, r, S'). В какие-то моменты будем выбирать $batchSize$ случайных элементов из него и обновлять Q-table с помощью них.

VI. Среда

В качестве среды для проекта была выбрана PyGame-Learning-Environment. Она содержит различные игры, среди которых есть Flappy bird.

В среде можно получить текущее состояние, которое состоит из y-координаты игрока, вертикальной скорости игрока, расстояния до следующей трубы, y-координаты свободного места в трубе, а также то же самое до следующей за этой трубы.

Но таких состояний слишком много и на них агент будет учиться слишком долго. Поэтому были учтены только такие параметры:

- расстояние до следующей трубы
- разница между y-координатой игрока и y-координатой нижней части "окна"в трубе
- y-составляющая скорости игрока

Но все эти параметры независимы и состояние имеет размерность около $310 \times 600 \times 30$, т.е. примерно $5 \cdot 10^6$ состояний. Нужно сократить их количество, поэтому первые два параметра будут сжаты в 20 и 10 раз соответственно по следующим формулам:

$$x \leftarrow \left\lfloor \frac{x}{20} \right\rfloor \cdot 20$$

$$y \leftarrow \left\lfloor \frac{y}{10} \right\rfloor \cdot 10$$

где x - первый параметр состояния, y - второй.

Теперь размерность состояния не более $16 \times 60 \times 30$. Всего новых состояний не более 30 000.

VII. Эксперименты

Сделаем несколько версий агентов с различными параметрами.

Для сравнения будем строить график средней награды от числа игр. Среднюю награду будем вычислять как среднее арифметическое за последние 30 игр.

Состояние будем описывать в таком формате:

$$(\alpha, \varepsilon, \gamma),\ (bufferSize, batchSize),\ flag,$$

где $\alpha, \varepsilon, \gamma, bufferSize, batchSize$ такие же как раньше, $bufferSize$ и $batchSize$ равны -1 если replay buffer не использовался; $flag \in \{0, 1\}$ - обновляется ли Q-table когда закидываем состояние в буфер или нет.

Были выбраны следующие варианты:

1) $(0.2, 1, 0.97), (10^4, 10), 0$
2) $(0.2, 1, 0.97), (10^5, 100), 0$
3) $(0.2, 1, 0.97), (10^4, 10), 1$
4) $(0.2, 1, 0.97), (10^5, 100), 1$
5) $(0.2, 1, 0.97), (-1, -1), 1$
6) $(0.2, 1, 0.97), (10^4, 3), 0$

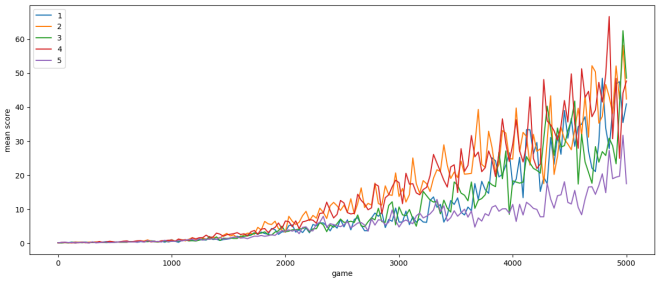

Рис. VII-.1: Результаты агентов с различными параметрами за 5000 игр.

По графику (Рис. VII-.1) видно, что $flag$ особо не влияет если есть replay buffer, поэтому больше не будем его учитывать. Лучше всего получились 2 и 4 потому что в них больше всего раз обновлялась Q-table, хуже всего - вариант без replay buffer.

Но, наверное, в данном случае важнее смотреть на качество обучения не по количеству игр, а по времени, затраченному на обучение. Это тоже не очень корректно т.к. другие процессы могут замедлять действие этого и все будет происходить хуже, но был выбран именно такой вариант.

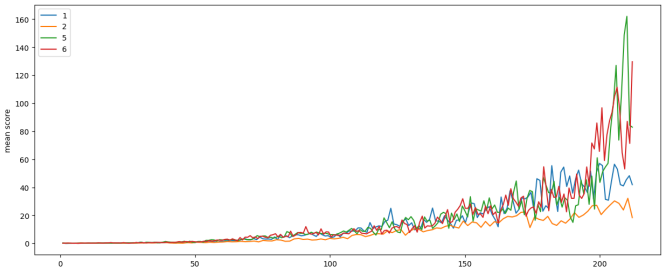

Рис. VII-.2: Результаты агентов с различными параметрами за 15 мин.

Из графика (Рис. VII-.2) понятно, что $5, 6$ лучше, чем $1, 2$, но непонятно кто из них лучше. Для последующего обучения был выбран агент с параметрами 5.

VIII. Заключение

Агент сыграл 10 000 игр в процессе обучения.

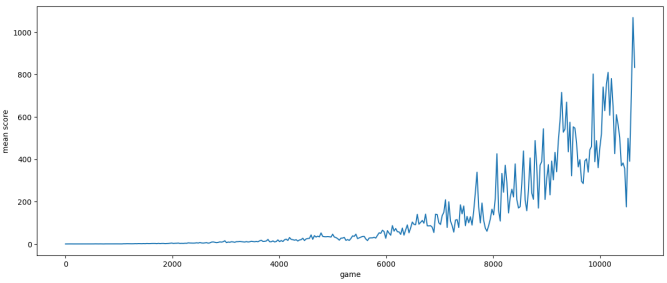

Рис. VIII-.1: Обучение за 10 000 игр.

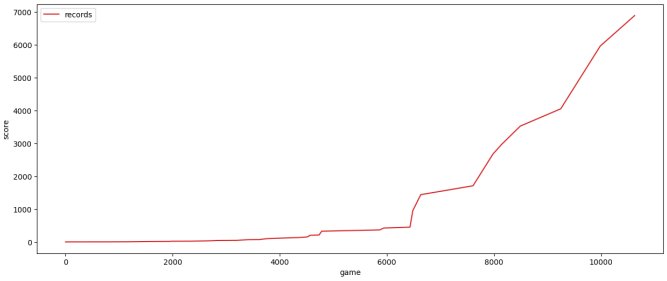

Рис. VIII-.2: График рекордов в процессе обучения.

После тренировки агент показывает такие результаты (Рис. VIII-.3):

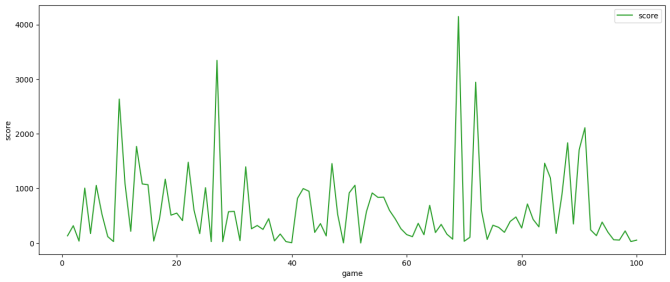

Рис. VIII-.3: Результаты тренированного агента.

После 6 000 игр агент живет достаточно долго, из-за чего на одну игру уходит много времени.

Сам проект с исходным кодом можно найти на GitHub.

Список литературы

[1] Beakcheol J, Q-learning Algorithms: A Comprehensive Classification and Applications

[2] William F, Revisiting Fundamentals of Experience Replay