Московский государственный университет имени М. В. Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра математических методов прогнозирования

Каюмов Эмиль Марселевич

Сборка кубика Рубика с помощью обучения с подкреплением

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

Научный руководитель:
д.ф-м.н., профессор
А. Г. Дьяконов

Москва, 2019

Содержание

Аннотация

В последние годы активно развивается обучение с подкреплением — область машинного обучения, в которой агент обучается не на готовых примерах, а в результате взаимодействия со средой. Достигнуты впечатляющие результаты, начиная с игр Atari в 2013 году и заканчивая AlphaZero в играх го и шахматах в 2018. Агент, ориентируясь только на правила игры, на собственном опыте научился играть лучше человека.

В данной работе сделана попытка применить методы обучения с подкрепления к задаче сборке Кубика Рубика. Кубик Рубик с точки зрения обучения с подкреплением — среда с большим количеством состояний и редким обратным сигналом. Предложены способы применения Deep Q-learning к сборке Кубика Рубика и использования дерева поиска для улучшения результата жадной стратегии. Показана успешность решения для среды с низкой сложностью без использования человеческих знаний, исследованы некоторые аспекты настройки агентов.

1 Введение

Актуальность?

В данной работе ...

План работы?

2 Постановка задачи

2.1 Задача обучения с подкреплением

В отличие от классической задачи обучения с учителем в обучении с подкреплением обучение происходит не на основе набора данных, полученного внешним процессом, а на основе взаимодействия в рамках достижения некоторой цели.

Агентом называют единицу, которая обучается и принимает решения. Агент взаимодействует с окружением. Взаимодействие представляется в виде последовательности дискретных шагов $t = 0, 1, \ldots$. На шаге t агент совершает действие a_t, в ответ на которое получает от окружения новое состояние среды s_{t+1} и числовую награду за совершённое действие r_{t+1}. Агент максимизирует суммарную награду в процессе взаимодействия с окружением. Схематично взаимодействие агента с окружением изображено на рис. (1).

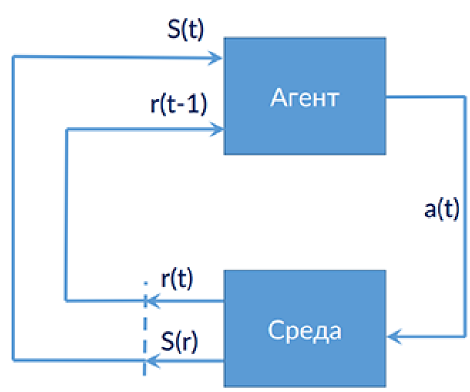

Рис. 1: Один шаг взаимодействия агента с окружением

Обучение — процесс поиска стратегии действий агента $\pi(s_t)$, которая максимизирует суммарную награду $\sum_t r_t$. Агенту неизвестны функции перехода $p(s_{t+1}|a_t, s_t)$ и награды $p(r_t|a_t, s_t)$.

2.2 Кубик Рубика

Кубик Рубика — механическая головоломка, изобретённая в 1974 году Эрнё Рубиком. В классической версии головоломка представляет собой пластмассовый куб 3x3x3 с 54 видимыми цветными наклейками. Грани большого куба способны вращаться вокруг 3 внутренних осей куба. Каждая из шести граней состоит из девяти квадратов и окрашена в один из шести цветов. Повороты граней позволяют переупорядочить цветные квадраты множеством различных способов. Задача игрока заключается в том, чтобы «собрать» или «решить» Кубик Рубика»: поворачивая грани куба, вернуть его в первоначальное состояние, когда каждая из граней состоит из квадратов одного цвета.

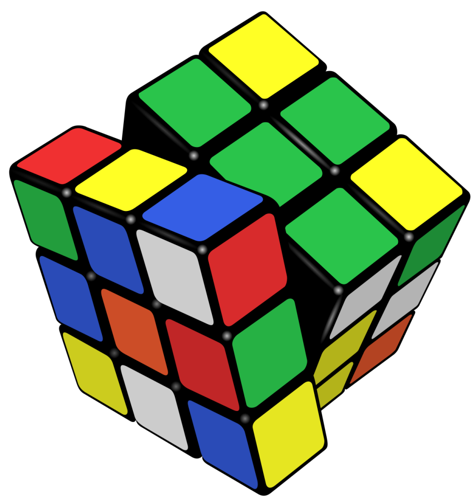

Рис. 2: Классический Кубик Рубика

Число различных состояний Кубика составляет $43 * 10^{18}$ (без учёта ориентации центральных неподвижных кубиков). За счёт большого числа состояний Кубик является сложной для решения головоломкой. Сущестуют различные алгоритмы по сборке Кубика. Алгоритмы, ориентированные на человека и соотвественно простоту запоминания, являются неоптимальными с точки зрения количества совершаемых действий. С 1981 года ведутся исследования по определению минимального количества ходов, необходимого для сборки Кубика из любой позиции. В 2014 году было показано, что Кубик может быть собран за 26 ходов (если считать только действия «поворот на 90 градусов») и за 20 ходов (если также считать за одно действие повороты на 90 или 180 градусов) [1], [2]. Существуют также машинные алгоритмы для сборки Кубика, например, алгоритмы Коцембы или Корфа, ориентировнные на сборку за минимальное число ходов [3].

Всего у Кубика 6 граней, каждую из которых можно вращать по и против часовой стрелки. Таким образом, взаимодействие с Кубиком заключается в 12 возможных действиях. Будем называть Кубиком весь Кубик Рубик, кубиками – составные части большого Кубика, наклейками – грани кубиков.

С точки зрения обучения с подкреплением Кубик описывается с помощью цветов на гранях, функция перехода детерминированная (возможно только одно состояние Кубика при применении конкретного действия в конкретном состоянии), награда равна нулю при переходе во все состояния кроме финального.

Целью данной работы является применение методов обучения с подкреплением к задаче сборки Кубика Рубика.

3 Подходы

В этом разделе рассмотрим существующие подходы обучения с подкреплением к задаче сборки Кубика Рубика и техники, используемые в рамках данной работы.

3.1 Известные подходы

Существует несколько попыток применения обучения с подкреплением к задаче сборки Кубика Рубика.

supervised

unpublished

та самая статья

3.2 Deep Q-learning

Одним из базовых методов обучения с подкреплением является метод Q-обучения, предложенный в 1992 году [4]. В этом методе используется понятие функции ценности пары состояние и действие $Q(s, a)$, которая определена для некоторой стратегии π и является математическим ожиданием дисконтированного вознаграждения $Q^{\pi}(s, a) = \mathbb{E}_{\pi,s,a}(R_t|s_t = s, a_t = a)$. Детерминированная стратегия агента восстанавливаеся по функции ценности: $\arg\max\limits_a \pi(a|s) = \arg\max\limits_a Q(s, a)$.

Алгоритм Q-обучения восстаналивает оптимальную функцию ценности. Для этого используется факт, что функцию можно выразить через саму себя. Для оптимальной функции ценности:

$$Q^*(s_t, a_t) = \mathbb{E}(r_{t+1} + \gamma \max_a Q^*(s_{t+1}, a))$$

В случае конечного пространства состояний S небольшой размерности саму Q-функцию можно хранить в виде таблицы и обновлять с помощью следующего соотношения:

$$Q(s_t, a_t) = (1 - \alpha)Q(s_t, a_t) + \alpha(r_{t+1} + \gamma \max_a Q(s_{t+1}, a))$$

В случае пространств большой размерности хранение и обновление функции в виде таблицы не представляется возможным, поэтому вместо таблицы строится апроксимация Q-функции. Например, это делается с помощью нейронных сетей [5]. Для этого существует два подхода. В первом из них на вход сети подаётся описание среды, сеть выдаёт одновременно значения Q-функции для всех возможных действий. Во втором подходе сеть оценивает значение Q-функции для конкретного действия, которое подаётся на вход сети вместе с состоянием среды. В данной работе будем рассматривать первый подход.

В процессе обучения Q-функции для агента используется механизм Experience Replay [5]. Его суть заключается в том, что вся история взаимодействия агента с окружением сохраняется в пямять ограниченной длины по стратегии FIFO. Каждый батч для обновления весов сети берётся случайным образом из этой памяти. Ограничение на объём памяти позволяет использовать только свежие примеры для обучения, так как взаимодействие агента с окружением в процессе обучения агента меняется. Использование же примеров только с последнего эпизода взаимодействия сказывается на процессе обучения негативно, так как примеры в рамках одного батча будут скоррелированы.

Если при обучении Q-функции в виде нейронной сети подсчёт градиентов производить с помощью этой же Q-функции, то Q-функция будет завышать значения. Самый простой способ обойти эту проблему — использовать для подсчёта градиентов зафиксированную Q-сеть, веса которой периодически синхронизируются с обучаемой сетью. Альтернативный подход заключается в использование второй Q-сети для вычисления действия, по котором достигается максимум Q-функции [6].

При применении DQN на этапе обучения сети необходимо одновременно улучшать Q-функцию с точки зрения точности предсказаний и с точки зрения исследования новых состояний. Существует две базовых стратегии исследования среды. ϵ-жадная стратегия с вероятности ϵ выбирает случайное действие, с вероятностью $1-\epsilon$ — действие, максимизирующее Q-функцию. Стратегия исследования по Больцману сэмплирует каждое действие с вероятностями, полученными после применения softmax-преобразования к Q-значениям.

3.3 Дерево поиска

Обученная Q-сеть уже способна решать задачу. Обычно для этого применяется жадная стратегия выбора действия, максимизирующая Q-функцию на каждом шаге. Однако важным этапом при применении алгоритма является ...

4 Эксперименты

В этом разделе опишем конкретнее используемые модели, эксперименты и результаты.

В экспериментах использовалась библиотека PyCuber для симуляции Кубика Рубика и Pytorch для обучения сети и реализации DQN. Код экспериментов опубликован [7].

Основной функционал качества «доля успешных сборок Кубика Рубика». В случае с тестированием на этапе обучения DQN накладывался лимит в 100 шагов в рамках одного эпизода. При применении дерева поиска накладывался лимит в 500 итераций спуска по дереву до листьев.

Коэффициент дисконтирования равен 0.9, веса target-сети обновлялись каждые 500 итераций, размер батча при обучении 512. В качестве метода оптимизации использовался Adam. Размер Experience Replay равен 1000000.

Кубик представляет собой сложную среду с большим количеством состояний, поэтому в экспериментах сложность среды уменьшалась. Начальное состояние среды задавалось как произвольное состояние в x шагов от успешной сборки Кубика. Увеличение x приводит к усложнению сборки Кубика. $x \approx 100$ соответствует произвольному состоянию Кубика (большие значения x не делают сборку сложнее).

4.1 Описание кубика

Описывать среду, то есть состояние Кубика можно двумя способами:

1. По цветам наклеек, то есть описывать цвета на каждогой из граней. В таком случаев Кубик описывается с помощью бинарного вектора длины $6 \times 9 \times 6 = 324$.

2. По позициям кубиков, то есть описывать местоположение каждого конкретного кубика. В таком случаев Кубик описывается с помощью бинарного вектора длины $8 \times 8 \times 3 + 12 \times 12 \times 2 = 480$.

Первой подход требует от агента дополнительно решать задачу сопоставления цветов одного кубика (чтобы понять, где находится некоторый угловой кубик, надо найти три его наклейки и по ним определить местоположение).

В эксперименте выяснилось (рис. (3)), что оба способа показывают сравнимое качество, поэтому для уменьшения размерности входного вектора использовался первый подход.

4.2 Фиктивная функция награды

Особенностью Кубика Рубика является то, что сигнал несёт только финальное состояние. Естественной функцией награды является индикатор успешности сборки,

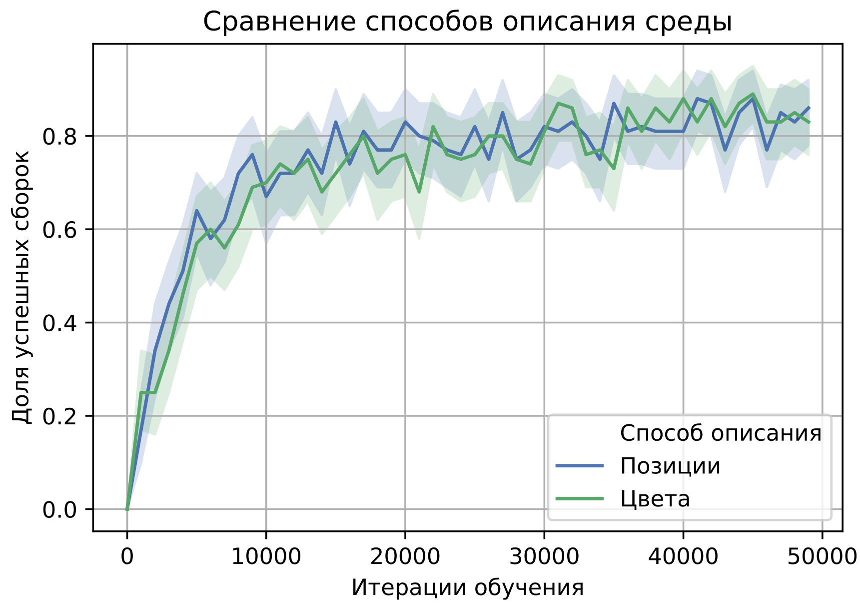

Рис. 3: Сравнение способов описания состояний через позиции кубиков и через цвета

однако с такой функцией сложно вести обучение модели из-за недостаточной обратной связи. Был предложен другой вариант функции награды: jценка состояния через долю наклеек (видимых сторон кубиков), совпадающих по цвету с наклейками собранного Кубика. Для стандартного Кубика это сумма $3 \times 3 \times 6 = 54$ индикаторов. Недостатком является то, что совпадение по цвету не означает нахождение на конкретной позиции нужного кубика. Награда после каждого шага равна разности оценок до и после шага.

4.3 Deep Q-learning

В качестве сети для Q-функции использовалась полносвязная сеть с 3 скрытыми слоями (2048/512/128 нейронов) и ReLU в качестве функции активации.

Обучение производилось на средах двух сложностей в 5 и 10 шагов до финального состояния. На рис. (4) можно увидеть прогресс при обучении модели (в качестве тестового окружения выступает Кубик сложностью в 5 шагов). Было получено, что качество почти не отличается от сложности обучающего окружения, поэтому дальнейшая работа велась на основе DQN, обученного на окружении сложности 5.

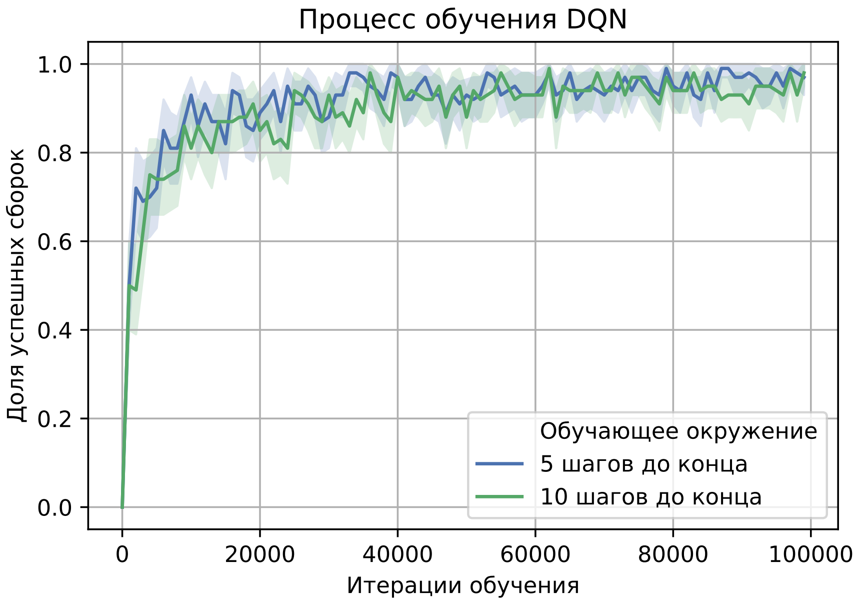

Рис. 4: Процесс обучения DQN для разных по сложности обучающих сред

4.4 Применение дерева поиска

5 Заключение

Список литературы

[1] The diameter of the rubik's cube group is twenty / Tomas Rokicki, Herbert Kociemba, Morley Davidson, John Dethridge // *SIAM Review.* — 2014.

[2] *Rokicki Tomas.* God's number is 26 in the quarter-turn metric. — 2014.

[3] *Kociemba Herbert.* Two-phase algorithm details.

[4] *Watkins Christopher JCH, Dayan Peter.* Q-learning // *Machine learning.* — 1992.

[5] Playing Atari with Deep Reinforcement Learning / Volodymyr Mnih, Koray Kavukcuoglu, David Silver et al. // *CoRR.* — 2013.

[6] *Van Hasselt Hado, Guez Arthur, Silver David.* Deep Reinforcement Learning with Double Q-learning. — 2015. — 09.

[7] Реализация и эксперименты. — `https://github.com/emilkayumov/rubiks-cube-reinforcement-learning`. — 2017.

[8] Solving the Rubik's Cube Without Human Knowledge / Stephen McAleer, Forest Agostinelli, Alexander Shmakov, Pierre Baldi // *arXiv preprint arXiv:1805.07470.* — 2018.

[9] Solving the Rubik's Cube Without Human Knowledge / Stephen McAleer, Forest Agostinelli, Alexander Shmakov, Pierre Baldi // *Arxiv preprint, submitted to NIPS 2018.* — 2018.

[10] *Brunetto Robert, Trunda Otakar.* Deep heuristic-learning in the rubik's cube domain: an experimental evaluation // *CEUR Workshop Proceedings.* — 2017.

[11] A Survey of Monte Carlo Tree Search Methods / Cameron Browne, Edward Powley, Daniel Whitehouse et al. // *IEEE Transactions on Computational Intelligence and AI in Games.* — 2012. — 03. — Vol. 4:1. — Pp. 1–43.

[12] *Kocsis Levente, Szepesvari Csaba.* Bandit Based Monte-Carlo Planning. — 2006. — Pp. 282–293.