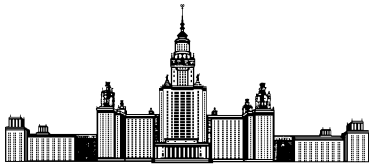
Московский государственный университет имени М. В. Ломоносова



Факультет Вычислительной Математики и Кибернетики

Кафедра Математических Методов Прогнозирования

КУРСОВАЯ РАБОТА СТУДЕНТА 517 ГРУППЫ

«Применение обучения с подкреплением к задаче сборки Кубика Рубика» (Reinforcement Learning for solving the Rubik's Cube)

Выполнил:

студент 1 курса магистратуры 517 группы

Каюмов Эмиль Марселевич

Научный руководитель:

д.ф-м.н., профессор

Дьяконов Александр Геннадьевич

Москва, 2018

Содержание

1 Введение

В последние годы активно развивается такая область машинного обучения, как обучение с подкреплением. Достигнуты впечатляющие результаты, доказавшие применимость и мощь методов, начиная с игр Atari в 2013 году и заканчивая AlphaZero в го и шахматах в 2018. Агент, ориентируясь только на правила игры, на собственном опыте учится играть лучше человека.

В данной работе сделана попытка применить методы обучения с подкрепления к очередной задаче — сборке Кубика Рубика (успешно решена только в мае 2018 [7]). Кубик Рубик с точки зрения обучения с подкреплением — среда с большим количеством состояний и редким обратным сигналом, чем напоминает шахматы. Предложены способы применения DQN к сборке Кубика, показана успешность решения для среды с низкой сложностью без использования человеческих знаний, исследованы некоторые аспекты настройки агентов.

1.1 Q-learning

Одним из самых простых алгоритмов в обучении с подкреплением является Q-learning [8]. Его основная идея заключается в том, что агент для выбора действия использует Q-функцию, оценивающую каждое возможное действие в каждом возможном состоянии. Для настройки Q-функции используется уравнение Беллмана и опыт от взаимодействия со средой. Q-функция в процессе обучения агента обновляется через своё старое значение и награду, полученную агентом в последней игре.

1.2 Deep Q-learning

В классическом вариаенте Q-функция представляет собой таблицу Q-значений для различных состояниям и действий. Очевидно, что размер такой таблицы растёт экспоненциально с ростом размерности состояния. Кроме этого табличный вид функции подходит только для дискретного описания состояний. Обойти эти недостатки можно, заменив Q-функцию на аппроксимирующую функцию, в качестве которой обычно рассматривают нейронную сеть. Сеть будет получать состояние и оценивать Q-значения для всех действий (возможен также вариант с оценкой лишь одного значения, если на вход также подавать действие для оценки).

1.3 Curriculum learning

В средах с возможностью контролировать сложность стартового состояния можно вести обучение агента не со случайного состояния, а постепенно увеличивая сложность. В статье [2] было показано, что такой подход приводит к улучшению финального качества. Идея метода заключается в том, что агент начинает учиться при низком уровне сложности среды, сложность постепенно повышается при стабилизации качества.

1.4 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 вра-

щаться 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 Кубик Рубика» (или «решить»): поворачивая грани куба, вернуть его в первоначальное состояние, когда каждая из граней состоит из квадратов одного цвета.

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 различных алгоритмов по сборке Кубика. Алгоритмы, ориентированные на человека (соответственно 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 поворот на 180 градусов) [3], [6]. Существуют также машинные алгоритмы для сборки Кубика, например, алгоритмы Коцембы или Корфа, ориентировнные на сборку за минимальное число ходов [4].

Кроме классической версии Кубика существует более простая версия 2x2x2 и более сложные версии с любым количеством кубиков вдоль каждого ребра (обычно ограничиваются 13-ю).

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.

2 Подход

Используется Deep Q-learning, в основе которого лежит полносвязная нейронная сеть с тремя скрытыми слоями (150/100/50 нейронов с relu-активациями). Размер входа зависит от способа описания состояния, на выходе из сети вектор из 12 чисел, соответствующий 12 возможным действия Кубика. Агент в процессе обучения использует Больцмановскую политику для выборка хода (семплирование с вероятностями, вычисляемыми softmax-преобразованием). В процессе применения выбирает ход жадно. После каждого сыгранного эпизода производится одна итерация оптимизации параметров сети, батч генерируется случайно из памяти ограниченной длины. В качестве оптимизатора используется Adam.

Естественной наградой для Кубика Рубика выступает сигнал об окончании сборки Кубика. Предложено использовать фиктивную функцию награды, передающую больше информации о процессе сборки агенту. Вычисляется награда как разность оценок состояний кубика до и после хода. Оценка состояния считается как количество кубиков (физических), стоящих на тех же позициях, что и кубики собранного Кубика. Для Кубика 2x2x2 это 8 угловых кубиков, для 3x3x3 соответственно $8 + 12$ (угловые и рёберные) кубиков.

Состояние Кубика описывается через кубики стоящие на угловых и рёберных местах (цвета кубика и его ориентации). Простым способом является one-hot кодирование наклеек Кубика, однако такое описание требует от агента дополнительно по цветам понимать местоположение каждого кубика, что усложняет задачу. Предложено кодировать состояние Кубика не через цвета наклеек, а через позиции кубиков. Мотивация следующая: для решения задачи агенту необходимо поставить на нужные места кубики, цвета на гранях это уже следствие (для удобства человека), поэтому агенту полезнее видеть положение конкретных кубиков, а не цветов на больших гранях. Например, для Кубика 3x3x3 кодирование происходит следующим образом: существуют 8 угловых кубиков, которые могут быть размещены на 8 углах в 3 положениях (ориентация цветов), и 12 рёберных кубиков, которые могут быть размещены на 12 позициях в 2 положениях. Таким образом, используя one-hot кодирование для позиций, получаем кодирующий вектор дляиной $8 \times 3 + 12 \times 12 \times 2$.

3 Эксперименты

В экспериментах использовалась библиотека pycuber для симуляции Кубика Рубика и pytorch для обучения сети и реализации DQN. Код экспериментов опубликован [9].

Кубик представляет собой сложную среду с большим количеством состояний, поэтому в экспериментах сложность среды упрощалась. 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).

Часть экспериментов проводилась на упрощённой версии Кубика Рубика 2x2x2. Он имеет то же число угловых кубиков (8) и не умеет ребёрных и центральных кубиков. Часть – на классической версии Кубика 3x3x3.

Тестирование агента производилось с ограниченеим эпизода в 100 шагов. Основной метрикой является процент успешных сборок Кубика. Коэффициент дисконтирования награды равен 0.9. Веса target-сети, по которой вычисляются q-значения, обновляются каждые 500 эпизодов.

3.1 Функции награды

Особенностью Кубика Рубика является то, что сигнал несёт только финальное состояние. Естественной функцией награды является индикатор успешности сборки, однако с такой функцией сложно вести обучение модели из-за недостаточной обратной связи. Были предложены и сравнены следующие варианты функции награды:

1. Оценка состояния через количество наклеек (видимых сторон кубиков), совпадающих по цвету с наклейками собранного Кубика. Для Кубика 2x2x2 это сумма $2 \times 2 \times 6 = 24$ индикаторов для каждой наклейки, для 3x3x3 соответсвенно $3 \times 3 \times 6 = 54$. Недостатком является то, что совпадение по цвету не означает нахождение на конкретной позиции нужного кубика. Награда после каждого шага равно разности оценок до и после шага.

2. Оценка состояния через количество кубиков (физических), стоящих на тех же позициях, что и кубики собранного Кубика. Для Кубика 2x2x2 это 8 угловых

кубиков, для 3x3x3 соответственно $8 + 12$ (угловые и рёберные) кубиков. Эта функция так же имеет недостаток, связанный с отличием от реальной цели сборки. Например, финальный перед сборкой поворот изменяет оценку состояния с 4 до 8, при этом существуют состояния, где не на своих позициях находятся лишь 2 кубика (то есть состояние оценивается в 6). Аналогично награда состоит в разности оценок до и после шага.

Рис. 1: Сравнение функций наград на сборке Кубика 2x2x2 с 5 и с 10 шагами от финального состояния

На рис. (1) изображен процент успешной сборки Кубика Рубика для различных функций наград и сложности сред. Заметим, что для простой среды с 5 ходами от финального состояния (слева) качество двух альтернатив выше, чем у естественной награды почти на 10 п.п.. На более сложной среде с 10 ходами до финального состояния (справа) естественная награда уже не позволяет выучить сборку Кубика даже в малом проценте случаев. Это можно объяснить тем, что агент не получает положительного сигнала от сборки Кубика, так как крайне редко добирается до финиша. Альтернативные награды имеют схожее качество между собой и за 50000 эпизодов учатся успешно собирать Кубик в 37% случаев.

3.2 Обучение с увеличением сложности

Постепенное увеличение сложности среды позволяет обучить агента быстрее и в некоторых случаях достигнуть более высокого финального качества.

	5 s.s.	10 s.s.	15 s.s.	20 s.s.
Train with 5 s.s.	87%	39.5%	14.5%	7%
Train with 5+10 s.s.	89%	38.5%	18.5%	6.5%
Train with 5+10+15 s.s.	88%	36%	15%	7%
Train with 5+10+15+20 s.s.	84.5%	32%	13%	6%

Таблица 1: Качество сборки Кубика 2x2x2 с последовательным увеличением сложности (по 50000 эпизодов, по горизонтали) на средах с разной сложностью (по вертикали)

Применительно к Кубику Рубика сначала агент учится собирать Кубик с состояния в нескольких шагах от успешной сборки, далее сложность среды (количество ходов от финального состояния увеличвается). Такой подход был сравнен с обычным обучением с нуля.

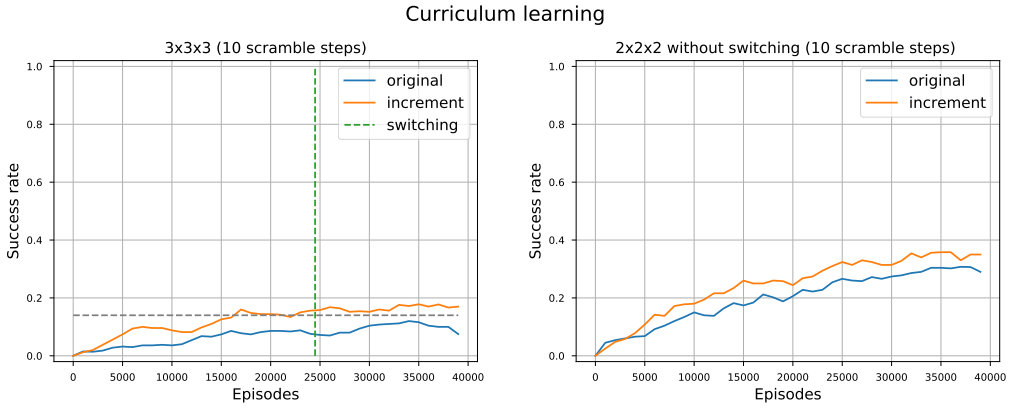


Рис. 2: Обучение с увеличением сложности среды. Слева для Кубика 3x3x3 (вертикальная черта соответствует моменту увеличения сложности), справа 2x2x2 без изменения сложности и обучением с меньшей сложностью.

Для Кубика 3x3x3 замечено (рис. (2) слева), что обучение на более простой среде с 5 ходами от финального состояния показывает более высокое качество на более сложной среде, чем при обычном обучении на среде с 10 ходами от финального состояния. Объяснить это можно тем, что процесс сборки для разных сложностей среди схож, однако исследование среды в простом случае агент делает эффективнее и чаще достигает успешной сборки. При увеличении сложности среды до 10 ходов от финального состояния также получен некоторый прирост в качестве, но лишь около 2-3 п.п..

Для Кубика 2x2x2 аналогично получено (рис. (2) справа), что более простая среда показывает более быстрое обучение с более высоким качеством. При повышении сложности среды (таб. 1) замечено, что среда в 10 ходов от финального состояния ещё позволяет агенту повышать качество сборки для сред различной сложности, но при большей сложности среды начинается деградация и качество у агента падает. Обучение велось по 50000 эпизодов на каждом из 4 уровней сложности.

Таким образом для Кубика Рубика эффект от постепенного увеличения сложности заметен слабо. Эффективней обучать Кубик на более простой по сравнению с тестовой средой.

3.3 Влияние длины тренировочного эпизода

Одним из параметров при настройке агента является ограничитель на длину эпизода. Меньшая длина не позволяет агенту зависать в бесперспективных состояниях (что попадает в обучающую выборку)>, но при этом ограничивает изучение агентом среды.

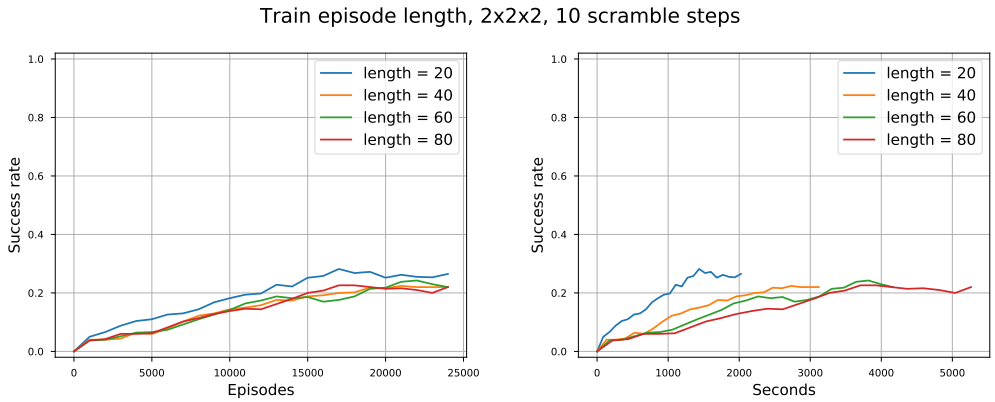


Рис. 3: Сравнение использование разных длин тренировочных эпизодов в количестве эпизозов (слева) и секундах (справа)

На рис. (3) изображено качество в зависимости от разных ограничений на длину эпизода по количеству эпизодов (слева) и количеству времени (справа). Заметим, что наименьшая длина эпизода позволяет не только экономить время в процессе обучения (производим оптимизацию весов сети чаще), но и быстрее обучаться с точки

зрения количества эпизодов. Это можно объяснить тем, что память, которую мы используем для генерации батча, содержит меньше случаев "блуждания"вдали от финального состояния, которое возникает при большой длине эпизода. Таким образом, можно заметно ограничивать длину эпизода в процессе обучения.

3.4 Способы описания среды

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 3x3x3 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 3x3x3 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).

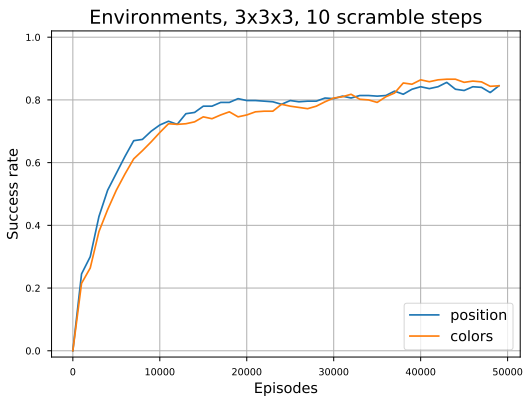


Рис. 4: Сравнение способов описания состояний через позиции кубиков и через цвета

В эксперименте выяснилось (рис. (4)), что описание через позиции кубиков показывает то же самое качество, что и описание через цвета кубиков.

3.5 Кубики 2x2x2 и 3x3x3

Кубик 2x2x2, несмотря на меньшее количество кубиков, имеет тот же набор действий и тот же показатель успешной сборки, что и классический вариант 3x3x3.

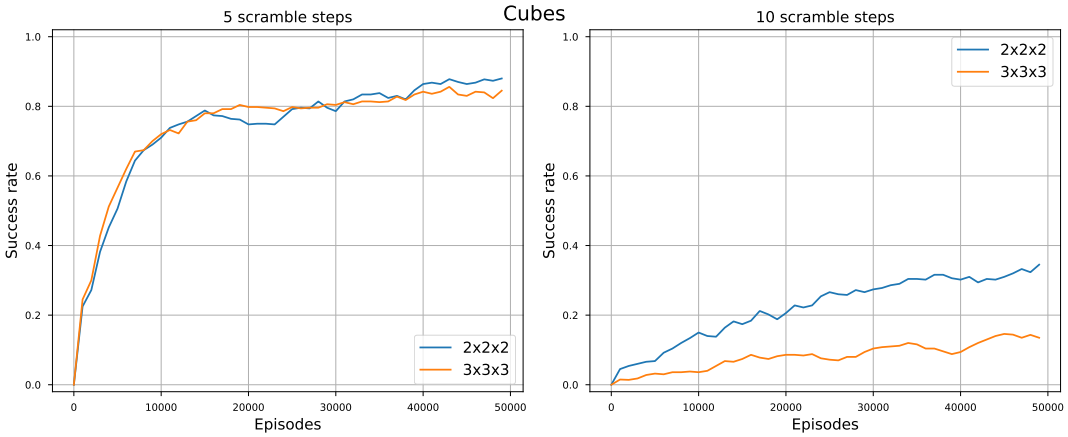


Рис. 5: Сравнение прогресса в обучении Кубика 2x2x2 с более сложным 3x3x3

Эксперименты показали (рис. (5)), что в условиях простой среды в 5 ходов от финального состояния агенты выучиваются до одного качества сборки Кубика. Однако при повышении сложности среды до 10 ходов от финального состояния агент начинает заметно хуже справляться с кубиком 3x3x3. Это можно объяснить тем, что только при большей сложности проявляется дополнительная сложности сборки рёберных кубиков. В простой среде в 5 ходов от финальной сборки рёберные кубики «не успевают» перемешаться между собой.

4 Заключение

Получено, что собирать Кубик с помощью DQN можно только для сред с небольшой сложности, когда Кубик близок к финальному состоянию, однако не используя при этом человеческих знаний.

Наилучшее достигнутое качество для уменьшенного Кубика Рубика 2x2x2 составило:

1. 89% (5 ходов от финального состояния)

2. 38.5% (10 ходов от финального состояния)

3. 18.5% (15 ходов от финального состояния)

4. 7% (20 ходов от финального состояния)

Наилучшее достигнутое качество для классического Кубика Рубика 3x3x3 составило:

1. 85% (5 ходов от финального состояния)

2. 18% (10 ходов от финального состояния)

В экспериментах замечено, что:

1. Фиктивная фукнкция награды позволяет агенту учиться эффективнее по сравнению с естественной функцией награды. В некоторых случаях естественная функция награды не позволяет агенту учиться из-за недостатка обработной связи.

2. Постепенное увеличение сложности среды незначительно помогает агенту учиться. Однако ускорить обучение можно использованием более простой среды.

3. Сокращение длины эпизода позволяет сэкономить время в процессе обучения агента без потери качества.

4. Способ описания Кубика (через цвета или через позиции) не влияет на качество агента.

5. Кубик 2x2x2 даётся агенту проще, однако при низкой сложности среды агент достигает такого же качества и на классическом Кубике Рубика.

В статье этого же месяца [7] предложен подход, использующий поиск по дереву методом Монте-Карло, похожий на алгоритм AlphaZero. Авторам удалось достигнуть 100% успеха сборки Кубика Рубика из любого состояния. Однако медианное время сборки составляет 10 минут на мощном железе, что нельзя сравнить с описанным в данной работе методом, требующий менее 1 секунды на решение или нерешение задачи.

Список литературы

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

[2] Curriculum learning / Y. Bengio, J. Louradour, R. Collobert, J. Weston // *In Proceedings of the 26th annual international conference on machine learning.* — 2009.

[3] The diameter of the rubik's cube group is twenty / T. Rokicki, H. Kociemba, M. Davidson, J. Dethridge // *SIAM Review.* — 2014.

[4] *Kociemba H.* Two-phase algorithm details.

[5] Playing atari with deep reinforcement learning / V. Mnih, K. Kavukcuoglu, D. Silver et al. // *CoRR.* — 2013.

[6] *Rokicki T.* God's number is 26 in the quarter-turn metric. — 2014.

[7] Solving the rubik's cube without human knowledge / S. McAleer, F. Agostinelli, A. Shmakov, P. Baldi // *Arxiv preprint, submitted to NIPS 2018.* — 2018.

[8] *Watkins C. J., Dayan P.* Q-learning // *Machine learning.* — 1992.

[9] Реализация экспериментов. `https://github.com/emilkayumov/rubik-cube-rl`.