УДК 004.89, 004.85

СРАВНИТЕЛЬНОЕ ИССЛЕДОВАНИЕ АЛГОРИТМОВ КЛАССИЧЕСКОГО Q-ОБУЧЕНИЯ И НЕЙРОСЕТЕВОГО Q-ОБУЧЕНИЯ ДЛЯ ОПТИМИЗАЦИИ УПРАВЛЕНИЯ АВТОНОМНЫМ АГЕНТОМ В АНИМАТ-ПОДОБНОЙ СРЕДЕ

Поспелов С.М., Бондаренко И.Ю.
Донецкий национальный технический университет
кафедра прикладной математики и информатики
E-mail: pospielov@gmail.com

В статье рассмотрена задача автоматического управления автономным агентом в анимат-подобной среде. Для обучения оптимальному управлению применялся классический и нейросетевой алгоритм Q-обучения с обратным переигрыванием. Были проведены экспериментальные исследования обоих алгоритмов в анимат-подобной среде и сделаны выводы об их эффективности.

Введение

Метод обучения с подкреплением хорошо показывает себя при обучении интеллектуального агента в незнакомой окружающей среде, даже если среда является нестационарной [1]. Он позволяет автономному агенту обучаться на основе собственного опыта без вмешательства разработчика. Для управления агентом в анимат-подобной среде небольшой и средней размерности хорошо подходит один из вариантов обучения с подкреплением - алгоритм Q-обучения. Оба алгоритма Q-обучения - обратного переигрывания и интерактивного обучения - достаточно эффективны для обучения агента достижению цели в анимат-подобной среде. И алгоритм обратного переигрывания, и алгоритм интерактивного обучения сходятся к оптимальному поведению за достаточно небольшое количество итераций [1]. Безусловным минусом этого алгоритма является то, что на задачах большой размерности он требует очень много памяти и вычислительных ресурсов, что может стать критическим при выборе алгоритма обучения.

Более продвинутым и менее требовательным к ресурсам является метод нейродинамического программирования. Основная идея этого метода - в аппроксимации оптимального поведения, а не в попытке его точного воспроизведения [2]. В данной работе приводятся экспериментальные сравнения двух видов алгоритма Q-обучения: классическое Q-обучение с обратным переигрыванием и нейросетевое Q-обучение с обратным переигрыванием. Подход обратного переигрывания был выбран ввиду того, что он показывает более стабильное схождение к оптимуму в анимат-подобной среде [3].

Формализация задачи управления автономным агентом в анимат-подобной среде

Среда, в которой производились эксперименты с аниматом в данной работе, представлена на рисунке 1. Данная среда представляет собой замкнутое пространство,

в которое помещается анимат [1]. Пространство ограничено с четырёх сторон стеной, в центре которой с каждой стороны есть углубление, содержащее «пищу». Задача анимата – найти «пищу» за наименьшее количество шагов.

На рисунке 1 используются следующие условные обозначения: стена (W), «пища» (F) и пустое пространство(-).

Робот имеет 8 сенсоров, определяющих содержимое смежных с ним ячеек. Таким образом, состояние робота определяется только состоянием соседних ячеек и кодируется по ходу часовой стрелки, начиная с севера.

W	W	W	W	W	W	W	W	W
W	W	W	W	F	W	W	W	W
W	W	-	-	-	-	-	W	W
W	W	-	-	-	-	-	W	W
W	F	-	-	-	-	-	F	W
W	W	-	-	-	-	-	W	W
W	W	-	-	-	-	-	W	W
W	W	W	W	F	W	W	W	W
W	W	W	W	W	W	W	W	W

Рисунок 1. Анимат-подобная среда

4

Если робот(R) находится в положении, изображенном на рисунке 2, то он получает следующую сенсорную информацию: -----WFW.

-	-	-	W	W
-	-	-	W	W
-	-	R	F	W
-	-	-	W	W
-	-	-	W	W

Рисунок 2. Пример положения робота

Всего возможно 17 различных состояний робота, учитывая то, что 9 центральных клеток кодируются одним состоянием.

В каждый момент времени анимат может двигаться в одном из 8-ми возможных направлений: 1 – север, 2 – северо - восток , 3 – восток , 4 – юго- восток , 5 – юг, 6 – юго-запад, 7 – запад, 8 – северо - запад.

Принципы взаимодействия робота и среды таковы:

1) если ячейка, в которую перемещается робот, пустая, то среда позволяет ему переместиться в эту ячейку.
2) если ячейка, в которую перемещается робот, содержит стену, то среда не позволяет ему переместиться в эту ячейку и робот получает наказание (отрицательное вознаграждение).
3) если ячейка, в которую перемещается робот, содержит «пищу», то среда позволяет роботу в неё переместиться и робот получает вознаграждение.

Анализ различных алгоритмов Q-обучения для оптимизации управления автономным агентом.

Алгоритм Q-обучения был предложен Воткинсом (Watkins) в 1989 году [4].

Данный алгоритм работает с Q-функцией, аргументами которой являются состояние и действие. Это позволяет итерационным способом построить Q-функцию и тем самым найти оптимальную политику управления. Целью обучения является максимизация награды r. Выражение для обновления Q-функции имеет следующий вид:

$$Q(s_t,a_t) \leftarrow r_t+\gamma * \max Q(s_t+1,a),\ a \square A \qquad (1)$$

Оценки Q- значений хранятся в 2-х мерной таблице, входами которой являются состояние и действие.

При использовании алгоритма обратного переигрывания, обновление Q-значений производится только после того, как агент достигнет поглощающего состояния (нахождение пищи). Поэтому агент после каждого шага должен запоминать текущее состояние, выбранное действие и непосредственную награду. После достижения агентом поглощающего состояния, у него должен быть сформирован список всех совершенных переходов. Этот список прокручивается в обратном порядке и осуществляется коррекция Q- функции. Алгоритм описан на рисунке 3.

Для всех сохраненных $\{(s_0,a_0,s_1,r_0)\ldots(s_n,a_n,s_{n+1},r_n)\}$ выполнить
1. $t \leftarrow n$
2. $Q_{real,t} \leftarrow Q(s_t,a_t)$
3. $u_{t+1} \leftarrow Q(s_{t+1},a_{t+1})$
4. $Q_{target} \leftarrow r_t+\gamma*[(1-\lambda)*u_{t+1}+\lambda*Q_{real,t+1}]$
5. Изменение поля Q-таблицы $Q(s_t,a_t) = Q(s_t,a_t) + Q_{target}-Q_{real}$
6. Если t=0 выход; иначе $t \leftarrow t$-1 и переход на 2-ой шаг

Рисунок 3. Алгоритм Q-обучения с обратным переигрыванием

Главная особенность нейросетевого алгоритма Q-обучения является использование вместо Q-таблицы нейросеть, которая вычисляет Q-фактор. В остальном алгоритмы идентичны. Алгоритм описан на рис. 4.

Для всех сохраненных $\{(s_0,a_0,s_1,r_0)\ldots(s_n,a_n,s_{n+1},r_n)\}$ выполнить
7. $t \leftarrow n$
8. $Q_{real,t} \leftarrow neurnet(s_t,a_t)$
9. $u_{t+1} \leftarrow neurnet(s_{t+1},a_{t+1})$
10. $Q_{target} \leftarrow r_t+\gamma*[(1-\lambda)*u_{t+1}+\lambda*Q_{real,t+1}]$
11. Обучение нейросети, реализующей $Q(s_t,a_t)$ при помощи алгоритма обратного распространения, где ошибка равна $Q_{target}-Q_{real}$.
12. Если t=0 выход; иначе $t \leftarrow t$-1 и переход на 2-ой шаг

Рисунок 4. Алгоритм Q-обучения с обратным переигрыванием

Экспериментальные исследования

Среда, в которой производятся эксперименты, является немарковской, или принадлежит к классу 2 по классификации, используемой Вильсоном в [5], т. к. данных сенсорного входа не всегда достаточно для однозначного определения действия, которое должен совершить робот в направлении ближайшей “пищи”. Это происходит потому, что в центре данной среды имеется пустое пространство, и, находясь в одной из 9-ти центральных ячеек, робот получает одинаковую сенсорную информацию.

Исследования проводились в среде, описанной в разделе «Формализация задачи управления автономным агентом в анимат-подобной среде».

Для обучения использовались следующие значения вознаграждений:

1) Анимат находит еду: r = 1.
2) Анимат передвигается: r = -0.99.
3) Анимат врезается в стену: r = -1.

Также для обучения использовались коэффициенты обучения γ=0.9 и λ=0.9.

Критерием оценки качества обучения агента будет служить среднее число шагов от начальной точки до «пищи» с каждой клетки пространства [1]. Минимальная сумма шагов до «пищи» с каждой клетки составляет 46 шагов. Схема переходов при минимальном количестве шагов изображена на рисунке 5. В этом случае, максимальное расстояние от начального положения до «пищи» составит 5 шагов, среднее число шагов составит 46/25 = 1,84.

Рисунок 5. Схема переходов при минимальном количестве шагов до «пищи»

Рисунок 6. Схема переходов при минимальном максимальном необходимом количеством шагов до «пищи»

В случае, изображенном на рис. 6, максимальное необходимое количество шагов до «пищи» составит 4 шага, сумма шагов из всех клеток 47 шагов, а среднее число шагов составляет 47/25 = 1.88.

Были проведены 2 серии экспериментов, первая из которых была направлена на оценку эффективности применения классического Q-обучения для управления аниматом, а вторая – на оценку эффективности применения нейросетевого Q-обучения. Каждая серия представляла собой 10 экспериментов. В ходе каждого эксперимента выполнялось: 1) обучение анимата и 2) определение средней скорости нахождения пищи обученным аниматом. Результаты (средние скорости нахождения пищи), полученные в первой и во второй сериях экспериментах, были сопоставлены между собой по методу знакового критерия Вилкоксона. В результате было получено, что при уровне значимости 0.05 различия в поведении аниматов, обученных по алгоритмам классического Q-обучения и нейросетевого Q-обучения, незначимы.

График, показывающий зависимость суммы переходов к «пище» от количества обучающих итераций для обоих алгоритмов изображен на рис 7.

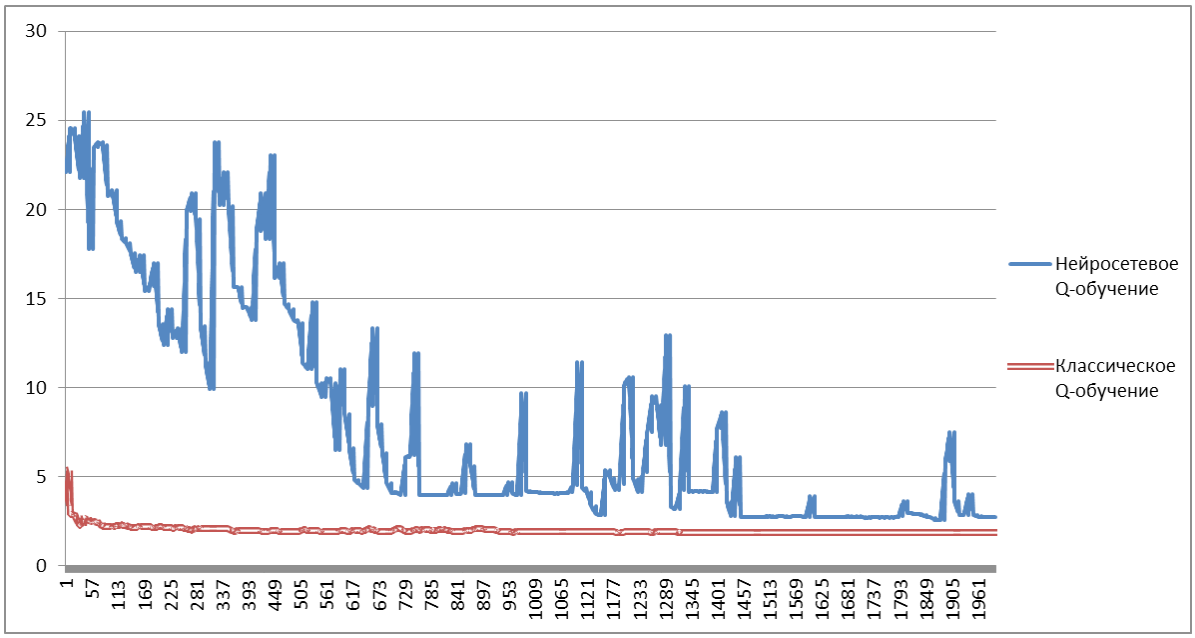

Рисунок 7. Схема переходов при минимальном максимальном необходимом количеством шагов до «пищи»

Заключение

При сравнении двух алгоритмов Q-обучения - обратного переигрывания и интерактивного обучения - оказалось, что оба они достаточно эффективны для обучения агента достижению цели в незнакомой среде. Нейросетевой алгоритм Q-обучения сходится за несколько большее количество итераций, но, тем не менее, обеспечивает квазиоптимальную стратегию поведения анимата. Главное преимущество нейросетевого алгоритма заключается в другом – этот алгоритм позволяет решать задачи произвольной размерности, для которых невозможно построить Q-таблицу.

Дальнейшие исследования будут направлены в сторону усложнения анимат-подобной среды для проверки эффективности работы алгоритмов на ситуациях, более

приближенных к реальным ситуациям внешнего мира, а также в усовершенствовании нейросетевого алгоритма для более быстрой и устойчивой сходимости.

Список источников

[1] Поспелов С.М., Бондаренко И.Ю. Сравнительное исследование алгоритмов q-обучения с обратным переигрыванием и интерактивным обучением для оптимизации управления автономным агентом в анимат-подобной среде // Сб. тр. междунар. научно-техн. конференции «Информационные управляющие системы и компьютерный мониторинг 2012». – Донецк: ДонНТУ. – 2012

[2] Поспелов С.М., Бондаренко И.Ю. Анализ проблем моделирования интеллектуального поведения персонажей в компьютерных играх // Сб. тр. междунар. научно-техн. конференции «Информатика и компьютерные технологии 2010». – Донецк: ДонНТУ. – 2010

[3] Поспелов С.М., Бондаренко И.Ю. Разработка модели интеллектуального поведения персонажа в компьютерной игре robocode на основе метода нейродинамического программирования. // Сб. тр. междунар. научно-техн. конференции «Информационные управляющие системы и компьютерный мониторинг 2011». – Донецк: ДонНТУ. – 2011

[4] Watkins, C., Dayan P., “Q-Learning”, // In: Machine Learning 8, Kluwer Academic Publishers, Boston, 1992 – pp. 279-292.

[5] Wilson S. W., “The Animat Path to AI. // In: From Animals to Animats: Proceeding of the First International Conference on the Simulation of Adaptive Behavior, Cambridge, MA: The MIT Press/Bradford Books, 1991.