Инженерный вестник Дона, №6 (2023)
ivdon.ru/ru/magazine/archive/n6y2023/8477

Применение и сравнение эволюционных алгоритмов в рамках задачи обучения с подкреплением для неустойчивых систем

А.А. Абузяров, А.А. Макаров

Российский государственный университет имени А.Н. Косыгина

Аннотация: Целью данной работы является реализация и сравнение генетических алгоритмов в рамках задачи обучения с подкреплением для управления неустойчивыми системами. Неустойчивой системой будет выступать объект CartPole Open AI GYM, который моделирует балансирование стержня, шарнирно-закрепленного на тележке, которая движется влево и вправо. Задачей является удержание стержня в вертикальном положении максимально продолжительное время. Управление данным объектом реализовано с помощью двух методов обучения: нейроэволюционный алгоритм (NEAT) и многослойный перцептрон с применение генетических алгоритмов (DEAP).
Ключевые слова: машинное обучение, нереволюционные алгоритм, генетические алгоритмы, обучение с подкреплением, нейронные сети.

Обучение с подкреплением (Reinforcement learning) — это наука о принятии решений, обучении оптимальному поведению в окружающей среде для получения максимального вознаграждения. Оптимальное поведение усваивается посредством взаимодействия с окружающей средой и наблюдения за ее реакцией. Обучение с подкреплением очень похоже на обучение с учителем, где в роли «учителя» выступает окружающая среда. Целью обучения с подкреплением является поиск последовательности действий, максимизирующие вознаграждение. Этот процесс аналогичен поиску методом проб и ошибок [1]. Для задач обучения с подкреплением разработано много специализированных алгоритмов: Q-обучение (формирование функции полезности Q и ее максимизации) [2], SARSA (State-Action-Reward-State-Action, оценивание шансов действий, приводящих к наилучшим оценкам) [3], DQN (Глубокие Q-сети, использование нейронный сетей для выбора оптимальных значений) [4] и др. Но поскольку в этих задачах фигурирует максимизация долгосрочного вознаграждения, можно рассматривать их как задачи оптимизации разных типов. Поэтому в обучении с подкреплением есть возможность использования генетических алгоритмов.

В рамках данной работы генетические алгоритмы использованы для поиска оптимальных параметров нейронной сети для управления заведомо неустойчивым объектом с целью проверки способности генетических алгоритмов к управлению такого рода объектами. Результаты исследований будут использованы для применения к промышленным задачам управления.

Объектом управления в рамках исследования выступала среда CartPole библиотеки OpenAI GYM [5]. CartPole моделирует балансировку стержня, прикрепленного к тележке, которая движется влево и вправо по рельсу без трения. Маятник расположен вертикально на тележке, и задача состоит в том, чтобы уравновесить шест, прикладывая усилия в левом и правом направлениях к тележке.

Цель состоит в том, чтобы удержать маятник от падения на как можно большем количестве временных шагов — в данном случае 500 временных шагов. Также есть дополнительные ограничения:

- Стержень отклонился более чем на ±12°;
- Положение тележки больше ±2,4 единицы.

В данной окружающей среде доступно два действия:

- Толкнуть тележку вправо (Действие - 1);
- Толкнуть тележку влево (Действие - 0).

Также среда CartPole возвращает объект observation, который используется в качестве входных данных на каждом временном шаге на основании которых определяется действие:

- положение тележки (от –2.4 до 2.4);
- скорость тележки (от минус бесконечности до бесконечности);
- угол наклона шеста (от –41.8° до 41.8°);
- скорость кончика шеста (от минус бесконечности до бесконечности).

Чтобы решить проблему CartPole, необходимо динамически реагировать на изменения в среде. Когда стержень начинает наклоняться в одном направлении, необходимо толкать тележку в этом направлении, но прекратите толкать, как только стержень начнет выпрямляться.

Существует большое количество нейроэволюционных алгоритмов, которые можно разделить на две группы. Первая группа – алгоритмы производят эволюцию весов сети при заданной топологии сети, ко второй группе относятся алгоритмы, которые помимо эволюции весов производят и эволюцию топологии сети (добавление новых связей, усложнение сети и т.д.) [6].

Для управления объектом CartPole реализованы и сравнены следующие подходы:

- Неэйроэволюционный алгоритм (NEAT) – производит эволюцию весов и топологии сети;
- Многослойный перцептрон (MLP) под управлением эволюционной вычислительной среды (DEAP) – производит эволюцию только весов, топология сети жестко задана и не эволюционирует.

Оба вышеприведенных подхода основываются на понятиях генетических алгоритмов.

Генетические алгоритмы предназначены для решения задач оптимизации и моделирования путём последовательного подбора, комбинирования и вариации искомых параметров с использованием механизмов, напоминающих биологическую эволюцию [7].

Неэйроэволюционный алгоритм (NEAT)

NEAT - Neuroevolution of augmenting topologies (Алгоритм нейроэволюции нарастающей топологии) предназначен для уменьшения размерности пространства поиска параметров посредством постепенного

развития структуры нейросети в процессе эволюции. Эволюционный процесс начинается с популяции маленьких, простых геномов (семян) и постепенно увеличивает их сложность с каждым новым поколением [8].

В данной работе алгоритм NEAT реализован с помощью библиотеки neat-python. Библиотека обеспечивает реализацию стандартных методов NEAT для моделирования генетической эволюции геномов организмов в популяции. Также она содержит утилиты для преобразования генотипа организма в его фенотип (искусственную нейронную сеть) и предоставляет удобные методы для загрузки и сохранения конфигураций генома вместе с параметрами NEAT. Для удобства работы с библиотекой используется конфигурационной файл config-neat для задания первичных параметров и инициализации алгоритма. Ниже рассмотрены основные из них:

- fitness_criterion (max) - функция, вычисляющая критерий завершения из набора значений приспособленности всех геномов в популяции. Значения параметров – это имена стандартных агрегатных функций, таких, как min, max и mean. Значения min и max используются для завершения процесса эволюции, если минимальная или максимальная приспособленность популяции превышает заданный порог fitness_threshold. Когда значение задано как mean, в качестве критерия завершения используется средняя приспособленность популяции. В данном случае требуется максимизация оценки - max;
- fitness_threshold (500) - пороговое значение, которое сравнивается с приспособленностью, вычисленной с помощью функции fitness_criterion для проверки необходимости завершения эволюции. Для решения задачи CartPole критерий завершения равен 500 временным шагам;
- pop_size (250) – количество организмов в поколении;

- reset_on_extinction (False) – флаг определяет необходимость создания нового поколения, если все виды в текущем вымерли;
- num_hidden, num_inputs, num_outputs – количество скрытых, входных и выходных узлов в геномах исходной популяции. Num_hidden = 1, num_inputs = 4, num_outputs = 1;
- activation_options (sigmoid) – функция активации.

На основе библиотеки NEAT реализован свой класс NeatBase, который реализует следующие методы:

- _run – метод предназначен для инициализации и запуска алгоритма. На данном этапе инициализируется алгоритм с основными параметрами из файла config-neat, инициализирует виртуальное окружения для взаимодействия с объектом CartPole и вызывается метод _run_neat;
- _run_neat – метод предназначен для создания первичной популяции и запуска сопутствующих методов для сохранения статистики _stat_reports, создания нейронной сети (эволюционирующей топологией и весами сети), запуск метода _test_genome тестирования сети на объекте CartPole;
- _visualize – метод для визуализации лучшего решения;
- _stat_reports – метод сбора статистики работы алгоритма;
- _test_genome – метод оценки работы генома. Метод взаимодействует со средой CartPole, совершая действие и получая вознаграждение, на основе которого строится приспособленность.

Многослойный перцептрон (MLP) под управлением эволюционной вычислительной среды (DEAP)

Данный алгоритм в отличии от NEAT требует отдельной реализации нейронной сети и генетического алгоритма. Для реализации нейронной сети использована библиотека ScikitLearn (MLPRegressor) [9].

Многослойный перцептрон (МСП) — это класс искусственных нейронных сетей прямого распространения, состоящих как минимум из трех слоёв: входного, скрытого и выходного. За исключением входных, все нейроны использует нелинейную функцию активации. МСП позволяет реализовать сложные отображения между входами и выходами. Для этого подстраиваются внутренние параметры сети такие как: веса, смещения активных нейронов. Для данной задачи используется сеть с одним скрытым слоем, состоящим из четырех нейронов. Входной слой также содержит четыре нейрона (входных значений от объекта CartPole), а выходной слой содержит всего один – действие, которое необходимо совершить. Ниже приведена архитектура сети.

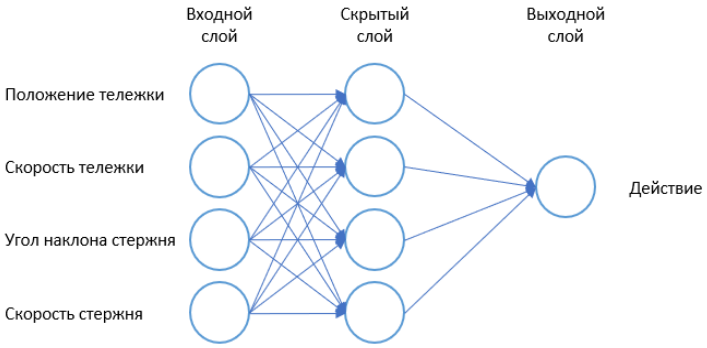

Рис. 1. - Архитектура нейронной сети

Для описания нейронной сети реализован класс MLPCartPole, который реализует следующие методы:

- __init__ - инициализация среды CartPole;
- _create_mlp – инициализации МСП с заданной архитектурой и весами, полученными от класса DeapBase, описанного ниже;
- _test_genome – взаимодействие со средой CartPole и получения оценки на основе действий;

- _visualize – метод визуализации лучшего решения.

DEAP - Distributed Evolutionary Algorithms in Python (распределенные эволюционные алгоритмы на Python) предназначен для быстрой разработки решений с применением генетических алгоритмов и других методов эволюционных вычислений [10]. Для реализации генетического алгоритма использована библиотека DEAP, на основе которой реализован класс DeapBase со следующими параметрами:

- Количество индивидуумов в популяции, вероятность скрещивания и мутации, максимальное количество поколений;
- Определение стратегии приспособления. В случае задачи CartPole – это максимизация оценки FitnessMax;
- Регистрация (представление) индивидуумов ("Individual", list, fitness=creator.FitnessMax);
- Регистрация оператора создания индивидуумов (individualCreator), создающий экземпляр класса Individual, заполненный случайными значениями;
- Регистрация оператора создания популяции (populationCreator);
- Регистрация оператора оценки приспособленности (evaluate) для оценки приспособленности индивидуума;
- Регистрация генетических операторов (select, mate, mutate). Select – турнирный отбор с размером турнира 2 (tools.selTournament, tournsize=2), Mate – ("mate", tools.cxSimulatedBinaryBounded, eta=CROWDING_FACTOR), Mutate – ("mutate", tools.mutPolynomialBounded, eta=CROWDING_FACTOR, indpb=1.0/NUM_OF_PARAMS);
- Применение элитистского подхода, сохранение лучших на данный момент индивидуумов из зала славы в следующее поколение;

После инициализации класса запускается основной метод _run_deap, который вызывает метод _test_genome класса MLPCartPole для оценки приспособленности индивидуума и метод _visualize для отрисовки наилучшего решения.

Тестирование алгоритмов.

Для оценки работы алгоритмов был проведен тест на способность алгоритмов управлять неустойчивым объектом.

NEAT

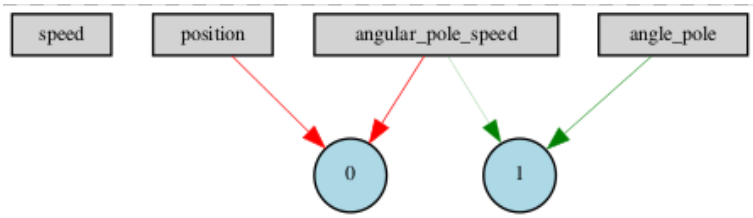

Рис. 2. - Граф генома победителя, кодирующего фенотип сети

Как видно из рисунка 2 сеть отличается от изначально заданной архитектуры. Сгенерированная сеть не имеет скрытых слоев, также важно отметить, что эволюция выработала свою стратегию управления, которая игнорирует скорость тележки, т.е. фактически исключает линейную скорость тележки из уравнения движения.

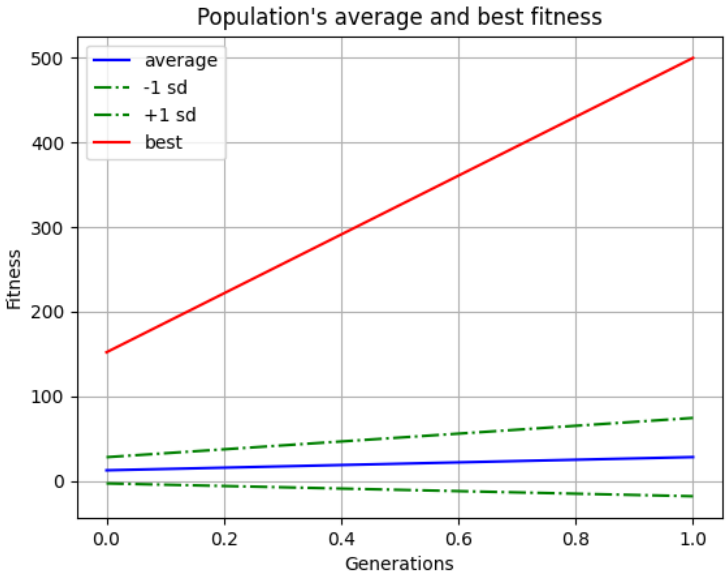

Рис. 3. – Среднее и лучшее значение приспособленности популяции Neat.

Как видно из рисунка 3, что задача по управлению CartPole была решена. Лучший индивидуум достиг максимальной оценки в 500 временных шагов. Также видно, что средняя приспособленность поколения была довольно низкой, но полезные начальные мутации помогли породить определенный тип организмов, которые смогли привести к появлению наилучшей особи.

DEAP

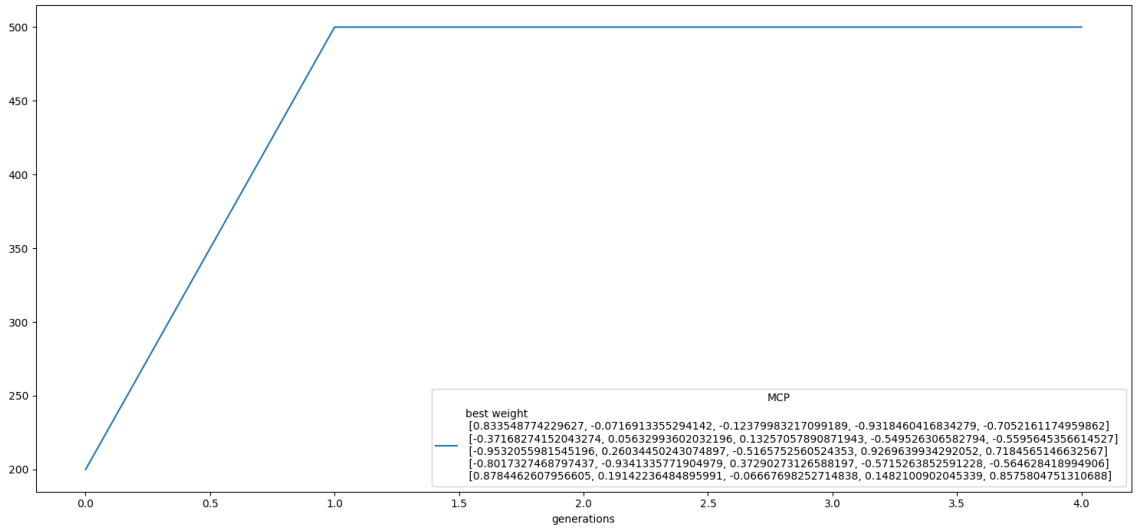

Рис. 4 – Лучшее значение приспособленности популяции Deap.

Как видно из рисунка 4 задача также, как и в случае с NEAT была решена, индивидуум также достиг оценки в 500. В отличии от NEAT, архитектура нейронной сети неизменна, меняются только веса сети. Веса лучшего индивидуума представлена также на рисунке 4 (best weight).

Из тестов видно, что оба алгоритма способны решать задачи по управлению неустойчивым объектом, но алгоритм NEAT является более гибким решением, так как способен подстраиваться под разные объекты управления, путем эволюции нейронной сети в отличии от алгоритма DEAP.

Литература (References)

1. Lim Hyun-Kyo, Ullah Ihsan, Han Youn-Hee. Reinforcement learning-based virtual network embedding: A comprehensive survey, Knowledge Based Systems. URL: sciencedirect.com/science/article/pii/S2405959523000346.
2. Han Abderraouf, Hentout Abdelfetah. Optimal path planning approach based on Q-learning algorithm for mobile robots.
   Knowledge Based Systems. URL: sciencedirect.com/science/article/abs/pii/S1568494620307341.
3. Shi Zhan, Zhu Jian, Wei Huina. SARSA-based delay-aware route selection for SDN-enabled wireless-PLC power distribution IoT, Knowledge Based Systems. URL: sciencedirect.com/science/article/pii/S1110016821007626.
4. Carta Salvatore, Ferreira Anselmo, Podda Alessandro Sebastian. Multi-DQN: An ensemble of Deep Q-learning agents for stock market forecasting, Knowledge Based Systems. URL: sciencedirect.com/science/article/abs/pii/S0957417420306321.
5. Brockman Greg, Terry Jordan. GYM Documentation URL: gymlibrary.dev/environments/classic_control/cart_pole/
6. Kapoor Arpit, Nukala Eshwar, Chandra Rohitash. Bayesian neuroevolution using distributed swarm optimization and tempered MCMC, Knowledge Based Systems. URL: sciencedirect.com/science/article/abs/pii/S1568494622006056.
7. Akcan Hüseyin. A genetic algorithm-based solution to the Minimum-Cost Bounded Error Calibration Tree problem, Knowledge Based Systems. URL: sciencedirect.com/science/article/abs/pii/S156849461830461.
8. Smith Allen W., Korinsky Kirill A. NEAT-Python Documentation URL: neat-python.readthedocs.io/en/latest/

9. Boisberranger Jérémie, Bossche Joris. Scikit Learn Documentation. URL : scikitlearn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html
10. Rainville François-Michel. DEAP Documentation DEAP. URL: deap.readthedocs.io/en/master/