Applying Reinforcement Learning Methods to Unstable Systems

Abuzyarov Alexander A.[1] and Makarov Alexander A.[2]
Russian State University named after A.N. Kosygin (Technology. Design. Art), Malaya Kaluzhskaya st. 1, Moscow, Russia
theschoolofrock9219@gmail.com@, makarov-aa@rguk.ru

Keywords: Machine learning, Reinforcement learning, Genetic algorithms, Bandit policies, Neural networks.

Abstract: The aim of this work is to implement, evaluate and compare reinforcement learning algorithms for the control of an unstable system for further application to industrial problems. An unstable system is a CartPole object from the OpenAI GYM library that simulates the balancing of a rod hinged on a cart that moves left and right. The purpose of controlling this object is to keep the rod in a vertical position for the longest possible time. For control, two methods are used: a multilayer perceptron and bandit policies using genetic algorithms to obtain the best individuals. The results of the two methods are presented and compared in the work.

1 INTRODUCTION

Reinforcement learning is a machine learning technique in which a system (agent) learns through trial and error. The idea is that the agent interacts with the environment, learning in parallel, and receives a reward for performing actions (Sutton, Barto, 2015). Reinforcement learning uses a method of positive reward for the right action and negative for the wrong one. Thus, the method assigns positive values to desired actions to induce the agent, and negative values to undesirable ones. This programs the agent to seek the long-term and maximum total reward in order to reach the optimal solution. These long-term goals do not give the agent the opportunity to stop there. Over time, the system learns to avoid negative actions and performs only positive ones. Reinforcement learning has a large number of applications: industry, business strategy planning, etc.

The application of reinforcement learning is a difficult task, as it requires the elimination of unforeseen errors; for this, in the framework of this work, tests were carried out on a deliberately unstable object. As part of the tests, reinforcement learning algorithms were used, which are trained under the control of genetic algorithms, i.e. the search for optimal solutions is carried out - weights for the neural network and parameters of bandit policies. The research results will be used for application to industrial control tasks.

2 RESEARCH METHODOLOGY

As part of the study, the CartPole environment of the OpenAI GYM library was used. CartPole models the balancing of a rod hinged on a cart that moves left and right on a frictionless rail. The pendulum is placed vertically on the cart and the task is to balance the pole by applying forces in the left and right directions to the cart (Greg Brockman, Jordan Terry).
The goal is to keep the pendulum from falling as many time steps as possible - in this case 500-time steps. There are also additional restrictions:

- The rod deviated by more than ±12°;
- Trolley position is greater than ±2,4 units.

There are two actions available in this environment:

- Push the trolley to the right (Action - 1);
- Push the trolley to the left (Action - 0).

The CartPole environment also returns an observation object, which is used as input at each time step, based on which the action is determined:

[1] https://orcid.org/0009-0003-0149-4591
[2] https://orcid.org/0009-0003-7463-1323

- trolley position (from –2.4 to 2.4);
- cart speed (from minus infinity to infinity);
- the speed of the tip of the pole (from minus infinity to infinity);
- Pole tilt angle (from –41.8° to 41.8°).

In order to solve the CartPole problem, it is necessary to dynamically respond to changes in the environment. When the rod begins to tilt in one direction, you should push the cart in that direction, but stop pushing as soon as the rod begins to straighten.

The reinforcement learning domain consists of several algorithms using different approaches. The differences are mainly related to their strategies for interacting with the environment:

- State-Action-Reward-State-Action (SARSA). This reinforcement learning algorithm starts by giving the agent a coefficient such as on-policy. In this case, policy is the probability by which the algorithm evaluates the chances of certain actions leading to rewards or positive states.
- Q-Learning. This reinforcement learning approach takes the opposite approach. The agent does not receive the policy (on-policy), therefore, its exploration of the environment is more independent (Fuqing Zhao, Qiaoyun Wang, Ling Wang, 2017).
- Deep Q-Networks. This algorithm uses neural networks in addition to reinforcement learning methods. Neural networks independently explore the reinforcement learning environment to select the most optimal value.

To interact with the CartPole environment, two reinforcement learning approaches are considered:

- Multi-armed bandits (SARSA);
- Neural network (Deep Q-Networks).

2.1 Multi-armed Bandits

Multi-armed bandits are a machine learning framework in which a system (agent) must choose actions (arms) in order to maximize its cumulative reward in the long run. In each round, the agent receives some information about the current state (context), then it chooses an action based on this information and experience gained in previous rounds (Nicolò Cesa-Bianchi, 2002). At the end of each round, the agent receives a reward associated with the chosen action.

The multi-armed bandit algorithm implements three main strategies:

- Epsilon-greedy (Alexandre dos Santos Mignon, Ricardo Luis de Azevedo da Rocha, 2017);
- UCB1 (Upper Confidence Bound) (Nevena Radović, Milena Erceg, 2021);
- Thompson sampling (Shipra Agrawal, Navin Goyal, 2012).

Epsilon Greedy performs hand selection with the Greedy Bandit Strategy. After some initial period of random attempts, say 1000 times, the algorithm greedily chooses the best option k in e percent of attempts. For example, if e=0.05, the algorithm chooses the best option 95% of the time, and random attempts the remaining 5% of the time.

UCB1 examines each hand first. Then, for each hand, a metric is calculated that represents the sum of the average win of the hand and some additional value. An additional value follows from Hoefding's inequality (Wassily Hoeffding, 1963), which sets an upper bound on the probability that the sum of random variables deviates from its mathematical expectation.

Thompson sampling (Bayesian bandit algorithm) - choosing the action that maximizes the expected reward with respect to a randomly chosen belief.

The River library (Max Halford, Saulo Martiello Mastelini) was used to study the quality and ability of bandit policies to manage the CartPole environment and the MultiArmedBandits class was implemented. This class implements the following methods:

- __init__ - initialization of CartPole environment;
- Init_bandits – initialization of bandit policies with parameters obtained from the class of genetic algorithms (see “Genetic Algorithms”);
- Get_score – the method is designed to get the score of each individual for each action while managing the CartPole environment;
- Show_plot – the method is designed to plot the best individuals of policy scores for each episode of work.

2.2 Neural Network

The neural network is based on the ScikitLearn (MLPRegressor) library (Jérémie du Boisberranger, Joris Van den Bossche). A multilayer perceptron (MLP) is a neural network consisting of layers, each of which consists of elements - neurons (more precisely, their models). These elements are of three types: sensory (input, S), associative (learnable "hidden" layers, A) and responsive (output, R). MLP allows you to implement complex mappings between inputs and outputs. To do this, it is necessary to adjust

the internal parameters of the network, such as: weights, biases of active neurons. In this problem, we use a network with one hidden layer consisting of four neurons. The input layer also contains four neurons, for each input value received from the environment, and the output layer contains only one - the action that needs to be performed. For this network architecture, it is required to find 20 weights and 5 biases. Following is the network architecture.

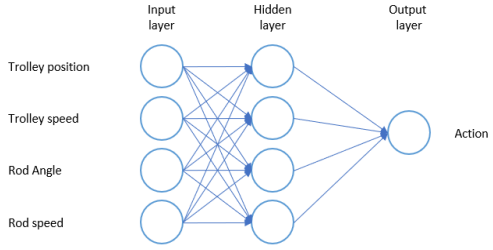

Figure 1: Architecture of the neural network.

To represent the solution for managing the CartPole environment, just like in the case of bandit policies, its own MLP class is implemented.

This class implements the following methods:

- __init__ - initialization of CartPole environment;
- Init_Mlp – initialization of a multilayer perceptron with the required architecture and weights obtained from the GeneticAlgorithm class of genetic algorithms (see “Genetic Algorithms”);
- Get_score - the method is designed to get the score of each individual for each action while managing the CartPole environment;
- Show_plot - The method is designed to plot the best individual MLP scores for each episode of work.

Genetic algorithms are used to find the best solution (neural network weights, bandits policy values).

2.3 Genetic Algorithms

Genetic algorithms are a family of search algorithms whose ideas are inspired by the principles of evolution in nature. By mimicking the processes of natural selection and reproduction, genetic algorithms can find high-quality solutions to problems involving search, optimization, and learning. To select the best solution, the GeneticAlgorithm class is implemented. The class is implemented using the DEAP (François-Michel De Rainville) library, so it is implemented in a certain way:

- Initialization of the class with the necessary parameters (number of individuals in the population, the probability of crossing and mutation, the maximum number of generations, etc.);
- Determining the adaptation strategy. In the case of the CartPole task, this is the score maximization ("FitnessMax", base.Fitness, weights=(1.0,));
- Registration (representation) of individuals ("Individual", list, fitness=creator.FitnessMax);
- Registration of the operator for creating individuals (individualCreator), which creates an instance of the Individual class filled with random values;
- Registration of the population creation operator (populationCreator);
- Registration of the fitness assessment operator (evaluate). The methods of the classes (MultiArmedBandits and MLP) get_score are called (see “Multiarmed bandits”, “Neural network”) to assess the fitness of the individual;
- Registration of genetic operators (select, mate, mutate). Select – tournament selection with tournament size 2 (tools.selTournament, tournsize=2),
  Mate-("mate", tools.cxSimulatedBinaryBounded, low=BOUNDS_LOW, up=BOUNDS_HIGH, eta=CROWDING_FACTOR),
  Mutate - ("mutate", tools.mutPolynomialBounded, low=BOUNDS_LOW, up=BOUNDS_HIGH, eta=CROWDING_FACTOR, indpb=1.0/NUM_OF_PARAMS);
- Applying an elitist approach, keeping the best individuals in the Hall of Fame at the moment into the next generation;
- Run the entire algorithm for 200 episodes and call the show_plot methods of the classes (MultiArmedBandits and MCP) (see “Multiarmed Bandits”, “Neural Eat”) to plot.

3 RESULTS OF THE STUDY

The problem is considered solved when the score reaches 500 (the number of steps at which the rod was held) for 200-time episodes.

Below is the work schedule of bandit politics:

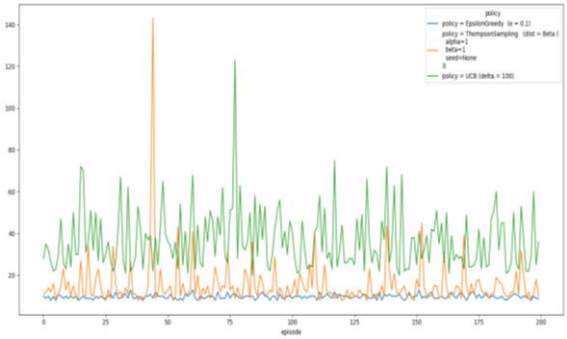

Figure 2: The result of the work of Bandits Policies.

As can be seen from Figure 2, the best policies of all individuals selected by the genetic algorithm were EpsilonGreedy with parameters e = 0.1, ThompsonSampling with parameters dist = Beta (alpha = 1, beta = 1) and UCB with parameters delta = 100. The best among the policy turned out to be the ThompsonSampling policy, which was able to hold the rod for 140-time steps.

Next, tests were carried out using a neural network:

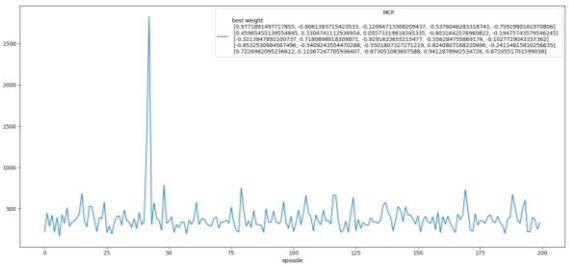

Figure 3: The result of the work MLP.

Figure 3 shows that the best individual is the network with weights = [0.9771891497717855, -0.8061383715423533, -0.12094713308209437, -0.5376046283318743, -0.7592990161970806, 0.45965455139554845, 0.3304741112936954, 0.05573319818345335, -0.8031642576960822, -0.19475743579546245, -0.3213947892100737, 0.7180898018309871, -0.9291633655215477, -0.556284755869176, -0.1027729043337362, -0.8532530984567496, -0.5409243554470288, -0.5501807327271219, 0.8240807168220996, -0.24114815810256635, 0.7226982095236612, 0.11067247705936407, -0.873051083607588, 0.9412878942534726, 0.6720551701599038]. The best result of the network is 2800-time steps.

4 THE DISCUSSION OF THE RESULTS

The aim of the work was to evaluate the ability of reinforcement learning algorithms: SARSA and DQN to manage a deliberately unstable system and compare them with each other.

From the “Research Results” item, it can be seen that ThompsonSampling turned out to be the best policy and individual chosen by the genetic algorithm, the SARSA (Multi-armed bandits) algorithm, but this algorithm could not solve the problem, since it was able to hold the rod for only 140-time steps at the target in 500. The best individual DQN (Multilayer Perceptron) algorithm, in turn, showed a fairly good result, the MLP with the network weights selected by the genetic algorithm was able to hold the rod for 2800-time steps.

5 CONCLUSION

The application of reinforcement learning algorithms to unstable systems shows the ability of such algorithms to control them. To do this, you need to select the correct algorithm and configure its parameters.

This work has shown that the use of the DQN algorithm with the search for the optimal solution using genetic algorithms for the MLP network is capable of solving problems of managing unstable objects.

REFERENCES

Sutton, R. S., Barto, A. G., 2015. Reinforcement Learning: An Introduction. London: The MIT Press Cambridge. 352 p.

Brockman, G., Terry, J., 2019. Gym Documentation. GYM URL: https://www.gymlibrary.dev/environments/classic_control/cart_pole/.

Zhao, F., Wang, Q., Wang, L., 2017. An inverse reinforcement learning framework with the Q-learning mechanism for the metaheuristic algorithm. *Knowledge-Based Systems*, vol. 265.

Cesa-Bianchi, N., 2002. Multiarmed bandits in the worst case. *Knowledge-Based Systems*, vol. 35, p. 91-96.

dos Santos Mignon, A., de Azevedo da Rocha, R.-A., 2017. An Adaptive Implementation of ε-Greedy in Reinforcement Learning. Knowledge-Based Systems, vol. 109, p. 1146-1151.

Radović, N., Erceg, M., 2021. Hardware implementation of the upper confidence-bound algorithm for

reinforcement learning. *Knowledge-Based Systems*, vol. 96, p. 107537.

Agrawal, S., Goyal, N., 2012. Analysis of Thompson Sampling for the Multi-armed Bandit Problem. Proceedings of the 25th Annual Conference on Learning Theory, PMLR 23:39.1-39.26.

Hoeffding, W., 1963. Probability inequalities for sums of bounded random variables. Journal of the American Statistical Association. № 301, p. 13–30.

Halford, M., Martiello, S., 2017. River Documentation. River URL: https://riverml.xyz/0.13.0/.

du Boisberranger, J., Van den Bossche, J., 2014. Scikit Learn Documentation // Scikit Learn URL: https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html.

De Rainville, F.-M., 2013. DEAP Documentation // DEAP URL: https://deap.readthedocs.io/en/master/.