BULGARIAN ACADEMY OF SCIENCES
PROBLEMS OF ENGINEERING CYBERNETICS AND ROBOTICS • 2020 • Vol. **73**, pp. 39-46
p-ISSN: 2738-7356; e-ISSN: 2738-7364
https://doi.org/10.7546/PECR.73.20.04

Developing Monte Carlo Simulator of Reinforcement Learning Type

Georgi Tsochev[1]

[1] *Laboratory of Telematics – BAS, Sofia, Bulgaria*
Emails: gtsochev@cc.bas.bg

Abstract: Monte Carlo methods are a way to solve the reinforcement learning problem based on average test results. To ensure that well-defined results are available, Monte Carlo methods are used only for episodic tasks. The Monte Carlo term is often used more widely in any valuation method whose operation involves significant participation on a random basis. Here it is specifically used for methods based on the average of full results (as opposed to methods that are learned from incomplete results). The paper describes a simulator for estimating raindrops in a specific area using the package matlib.

Keywords: *Monte Carlo, reinforcement learning, simulation, matlib, python*

1. Introduction

The Monte Carlo method is extremely useful in designing businesses, technologies, planning changes in an organization. Instead of costly experiments in kind, a variety of organizational work and equipment can be experimented on a computer. In a number of cases, computer modelling is the only way to get information about the behaviour of the system. Surely the estimates do not come in freely [1]. There is no universal solution to choose from the "magic" software menu that returns the result as soon as you complete your assignment. A good model of the system to be explored should be created to produce results of practical value. It is necessary to thoroughly examine the actual flow of requests, to carry out a time measurement for the operation of the individual nodes, and so on. When the system is in the design phase and does not yet exist, design parameters and technological boundaries are used. In general, it is necessary to know the probability laws of functioning of the individual parts of the

system, which model the way their work fluctuates around the project values. Then the Monte Carlo method allows calculating the probability laws of the entire system, taking into account the interconnection between the parts and parts, no matter how complex it is.

It can be said that human interaction with the environment is the primary way of perceiving new knowledge, which is the essence of learning. From an early age, movement and contemplation are a form of learning without the intervention of a direct teacher. Causal investigative connections that are created during the game provide information about the consequences of actions. Throughout life, learning about the external environment and one's own self is influenced by daily activities. From conversational skills to learning to drive a car, etc. we learn how our decisions and follow-up lead to different results. According to many theories, interactive learning plays a major role in educational activities [2]. In this regard, different approaches to interactive learning have been proposed. For example, some of them are focused on the design and development of an interactive multimedia e-learning system for the purposes of engineering training [3]. Others are focused on the process of teaching and learning complex algorithms that are hard to understand [4] or building e-learning applications for their effective use [5]. There is also an integrated framework that except the necessary course materials including corresponding tests and exercises provides an integrated environment where learners could test the written programming codes [6]. It should be mention that the latest approaches in interactive learning involve gamification too [7].

Reinforcement Learning is learning what to do – how to deal with situations with such actions to get the maximum digital signal of reward. The learner is not told what action to take, but instead has to find alone which actions give the greatest reward by trying them out. Each action affects the subsequent/final rewards, as well as the nearest direct reward in some interesting and challenging actions. For trial and error reinforcement training, aggregate and delayed remuneration are the most important characteristics [8].

Anyone who has owned a pet and trained it has most likely used an approved training form. What is the standard way to get a pet – such as a dog – to carry out certain commands? We give the command to the dog and if he responds in the way we expect, we give him a reward. If he does not answer/react correctly – we do not give him a reward. After a certain number of attempts and rewards on our part, the dog should start responding to our commands only with the right reactions, expecting a reward every time – when he hears “sit” – to sit down. We can say that he has learned to associate certain voice commands and their respective actions with a reward. Reinforcement Learning is both: 1) Class solution methods that work well on a problem, 2) the field that studies problems and the methods for solving them.

It is convenient to use the same name for the three parts, but at the same time it is important to keep the three concepts separate. In fact, the difference between problems and solution methods is very important in Reinforcement Learning. The

inability to make this distinction is a source of much confusion. One of the interesting methods that we will consider and use to develop the Reinforcement training type simulator is the Monte Carlo method.

2. Monte Carlo Methods

Monte Carlo methods require only experimentation – testing a series of states, actions and rewards from actual or simulated interactions with the environment. This teaching is a result of real experience and is astonishing because it does not require prior knowledge of the dynamics of the environment, but it can still achieve optimal behaviour. Training from simulated experience is also strong. Although a model is needed, the model only has to generate test states rather than the full probability distribution of all possible states that are required for dynamic programming. In surprisingly many cases, it is easy to generate an experience taken in accordance with the desired probability distribution, but it is impossible to obtain a full-scale distribution [9].

Monte Carlo methods are based on averaged outcomes of a problem, which can specifically enhance learning. To obtain well-defined results, this method is applied to episodic tasks. In these cases, an experience is divided into several completed episodes, regardless of the chosen actions. When a completed episode is present, the values of the behaviour and the rows can change. Therefore, for step-by-step (online) activities, the method is not used, but may have partial applications in the context of episodicity. In an estimation method whose operation involves significant random participation, the term Monte Carlo is more widely used. Here we use it specifically for methods based on the average of the complete results (as opposed to methods that are learned from incomplete results) [2].

There are several states, each acts as a different problem (such as associative search) and the various problems are interrelated. This means that post-performance compensation in one state depends on the actions taken at a later stage in the same episode. Since all selections of actions are subject to learning, the problem becomes unsettled from an earlier point of view.

To do this, we adapt the idea of repeating the common line of behaviour (GPI) developed in Dynamic Programming. While in Dynamic Programming we have calculated value functions of the Markov Process, Monte Carlo learns the value functions of the test results using the Markov Process. The value functions and their respective lines of behaviour continue to interact to achieve optimality in the same way. In Dynamic Programming, we first look at the predicted problem (calculating vp and qp for a fixed random line of behaviour π), then improving the line of behaviour, and finally the controlled problem and its solution through the GPI. Each of these ideas, taken from dynamic programming, extends to the Monte Carlo case in which we have only an exemplary experience.

In principle, Monte Carlo methods can be used to solve any problem. Under the law of large numbers, integrals described with an expected value of any random

variable, the result can be approximated by taking the empirical value independently of the specified variable. When the probability distribution of the variable is parameterized, mathematicians often use the Markov Monte Carlo collection (shortly: MCMC). The main idea is to create a model of the Markov chain of described joint probability distribution.

The name of the method is chosen for the glory of the capital of Monaco as a European gambling centre. Since this great project first uses computer and the theory of random trajectories to obtain a probable solution to a complex physical problem. This type of mathematical experiments is called Monte Carlo methods. McCracken in 1955 in introducing the early Monte Carlo methods in Scientific American magazine writes: "The Monte Carlo method is primarily used to solve problems that are determined in some important way probability - tasks in which the physical experiment is unfeasible and the creation of exact formula is impossible". The American Mathematicians John Neumann and S. Ullam – 1949, are the creators of the method.

Let's illustrate the capabilities of the method with a geometric example of defining an area of complexity (Fig 1).

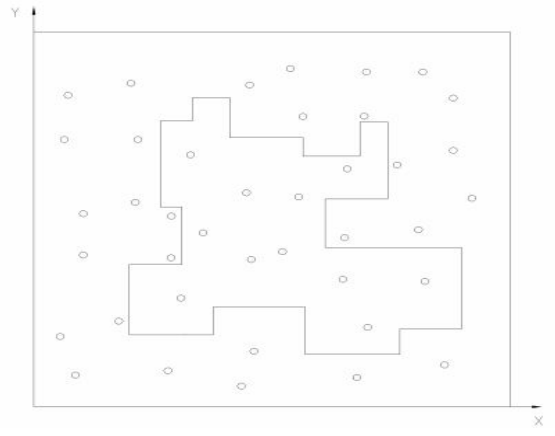

Fig. 1. Geometric example of defining an area of complexity [8]

The square shown in the figure has area 1. Look for the area of the complex figure in the square. Apply N random points in the square. The points in the figure are N' in number. It is obvious geometrically that the area $S = N'/N$. The more N points the greater the accuracy. In the example of Fig. 1 $N = 40$, and $N' = 11$. The calculated area is $S = 11/40 = 0.275$, with the actual area being 0.28.

The Monte Carlo method is not used to determine two-dimensional areas. This is why more precise methods are applied. But when determining areas and volumes of bodies in the Monte Carlo multi-dimensional space, the method is the only option to solve the problem [10]. The advantages of Monte Carlo methods over dynamic

programming are several. For example, without a model of environmental dynamics, optimal behaviour can be found directly from the interaction with it. Trial episodes can also be simulated using the methods in question using simulation or sample models. Monte Carlo methods also allow focusing and evaluating a small part of the conditions without the need to study the overall picture.

Monte Carlo methods operate on an exemplary experience, and this can be used for direct learning without the need for a model. They also do not update their valuation value based on other valuation values.

3. Algorithm for Rain Drops

Problem solving using the Monte Carlo method is based on statistical measurements. In the most general form, if an event will occur under certain conditions (probability P), the generation of the same conditions by a computer can be performed repeatedly. The number of times the event occurs divided by the number of conditions generated should be approximately equal to P [1].

The idea is to simulate random (x, y) points in a 2-D plane with domain as a square of side 1 unit. Imagine a circle inside the same domain with same diameter and inscribed into the square. Then, we calculate the ratio of number points that lied inside the circle and total number of generated points.

The proposed algorithm is composed of the following ten steps:

1. Initialize $circle\_points$, $square\_points$ *and interval to* 0.
2. Generate random point x.
3. Generate random point y.
4. Calculate $d = x * x + y * y$.
5. If d <= 1, increment circle points.
6. Increment square points.
7. Increment interval.
8. If increment < NO_OF_ITERATIONS, repeat from 2.
9. Calculate $pi = 4 * (circle\_points/square\_points)$.
10. Terminate.

4. Simulation and Results

To show clearer results from the simulator, several different simulations will be done with a different number of rain drops. The obtained results from the simulations presented below, besides the python source code compiler, used also Matplotlib.

The obtained results in the case of 100 drops by using the proposed simulator are illustrated in Fig. 2.

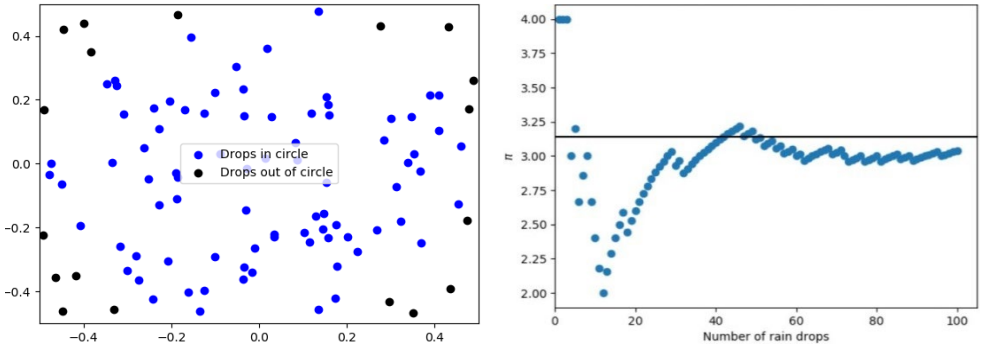

Fig. 2 a) 100 raindrops, b) π estimate against number of raindrops

The results in Fig. 3 represent the simulation of 500 drops.

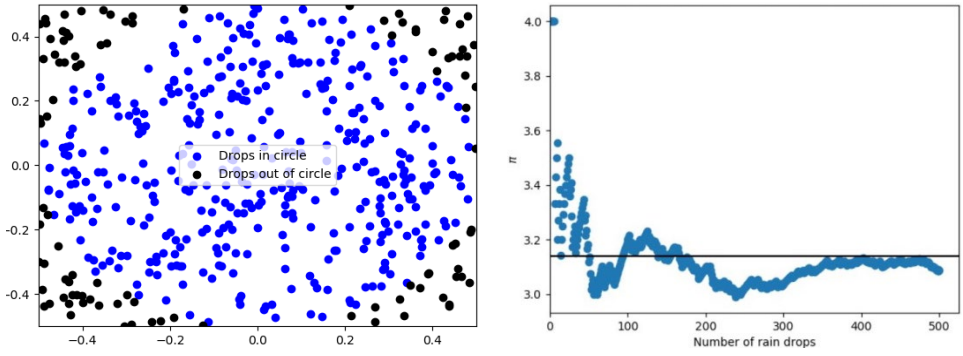

Fig. 3 a) 500 raindrops; b) π estimate against number of raindrops

The corresponding results when simulated 100000 drops are shown in Fig. 4.

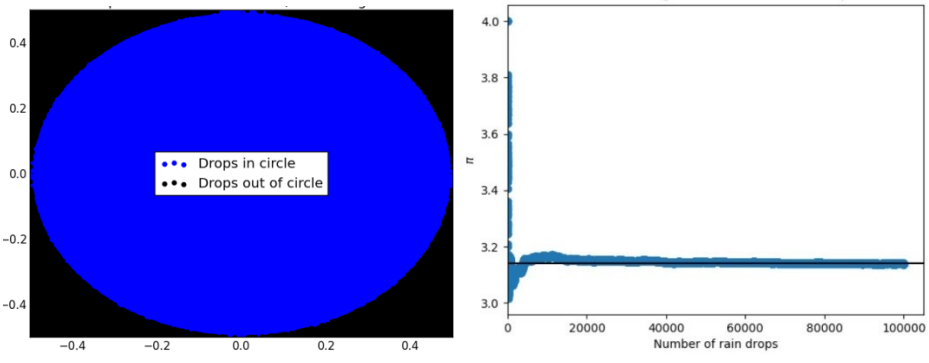

Fig. 4. a) 100000 raindrops; b) π estimate against number of raindrops

Increasing the number of drops up to 1000000 leads to the results as shown in Fig. 5.

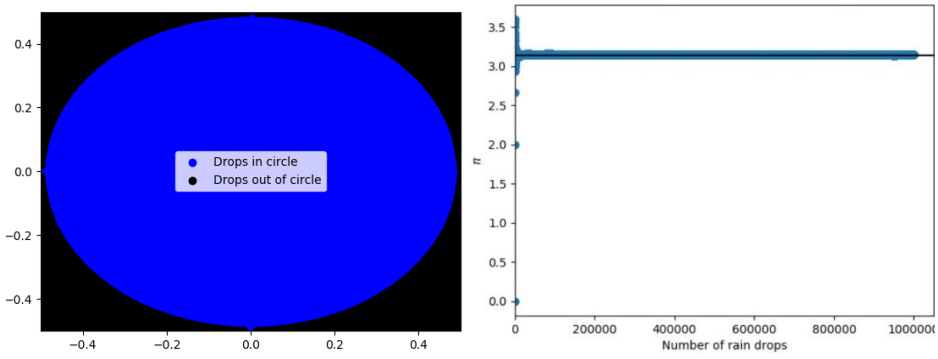

Fig. 5. a) 1000000 raindrops; b) π estimate against number of raindrops

The results show the applicability of Monte Carlo for the purpose of calculating raindrops in a given area.

The example given is not complicated but illustrates how the Monte Carlo method of Reinforcement is applied. The calculation of decision-making features and state-of-the-art optimal strategies is easily achievable through the demonstrated algorithms and methods, provided we have access to the dynamics of the environment – we know the transit function and the instant prize function. In the real world, this is quite honest information that the agent does not have access to while studying.

The results of the Monte Carlo simulation are only approximate to the actual value and are far less accurate. The Monte Carlo method in combination with reinforcement learning and agent based technology can be used for cybersecurity [11]. Agents have a table of ordered pairs: status and possible action, along with estimated reward values. After each game, agents update the rewards values for selected ordered pairs (status and action). Monte Carlo training updates every condition that the agent has gone through with the same reward value. An agent benefiting from reinforcement learning has the following dilemma: choosing between an action that is considered the best (exploitation) or choosing other actions to see if any of these actions is better (exploration). For the Monte Carlo approach, four different research algorithms are used to attempt to address this problem in cyber security, namely e-greedy, Softmax, Upper Confidence Bound 1 and Discounted Upper Confidence Bound [12].

5. Conclusion

Simulation is a type of tool that helps to make analytical decisions. The simulation software provides an opportunity to compare and reasonably evaluate various alternative projects, plans and policies. It finds a particularly valuable application when there is a high level of uncertainty as to the end result of an alternative that is under consideration. The simulation model generates a number of probability scenarios, thus accumulating a quantitative statistical sample of the expected end result, which helps to find a solution to this uncertainty.

The article shows the use of Monte Carlo methods to calculate raindrops in an area. Limitations of the presented simulation are explained. The main reason to use the Monte Carlo method is the possibility to get a quicker response to a question that usually takes a lot of time to resolve. Of course, the method also has its drawbacks. It is not yet generally accepted whether Monte Carlo can be used to simulate systems that are not in equilibrium (i.e. a transient state). Another drawback is that in order to use the Monte Carlo simulation a large number of samples must be generated and this may take a long time to achieve the desired results, since only one sample cannot be generated and immediately used in simulation, and it is necessary to make many samples and get their average value.

References

1. Monte-Carlo Simulation, https://brilliant.org/wiki/monte-carlo/, last accessed 2020/04/10
2. Sutton, R. S., Barto, A. G.: Introduction to Reinforcement Learning (1st. ed.). MIT Press, Cambridge, MA, USA (1998).
3. Borissova, D., Mustakerov, I.: E-learning tool for visualization of shortest paths algorithms. Trends Journal of Sciences Research 2(3), 84–89 (2015).
4. Borissova, D., Mustakerov, I.: A Framework of Multimedia E-Learning Design for Engineering Training. In: International Conference Advances in Web Based Learning – ICWL 2009, Marc Spaniol, Qing Li, Ralf Klamma, Rynson W.H. Lau (Eds.), Springer, Lecture Notes in Computer Science, vol. 5686, pp. 88-97 (2009).
5. Yoshinov, R, Arapi, P., Christodoulakis, S., Kotseva, M.: Supporting personalized learning experiences on top of multimedia digital libraries. International Journal of Education and Information Technologies 10, 152–158, (2016).
6. Mustakerov, I., Borissova, D.: A framework for development of e-learning system for computer programming: Application in the C programming language. Journal of e-Learning and Knowledge Society 13(2), 89–101 (2017).
7. Borissova, D., Keremedchiev, D., Tuparov, G.: Multi-criteria model for questions selection in generating e-education tests involving gamification. TEM JOURNAL – Technology, Education, Management, Informatics 9(2), 779–785 (2020).
8. Othman, A.: Enhancing the Performance of Flexible AC Transmission Systems (FACTS) by Computational Intelligence. Doctoral Dissertations, Aalto University publication series, 174 pages, 2011, http://lib.tkk.fi/Diss/2011/isbn9789526041766/isbn9789526041766.pdf
9. Monte Carlo Method, https://en.wikipedia.org/wiki/Monte_Carlo_method, last accessed 2020/05/01.
10. Petrov, T.: The Monte Carlo Method – Basics And Application In Risk And Reliability Evaluation and Forecasting, http://www.mgu.bg/drugi/proekti/MONTE%20CARLO/Petrov%20T%20-%20SAFERELNET%2005_Monte%20Carlo%20Application%20in%20Safety.pdf, last accessed 2020/05/20
11. Gaidarski, I., Minchev, Z., Andreev, R.: Model driven architectural design of information security system. In: Madureira A., Abraham A., Gandhi N., Silva C., Antunes M. (eds) Proc. of Tenth International Conference on Soft Computing and Pattern Recognition (SoCPaR 2018). Advances in Intelligent Systems and Computing, vol. 942, pp. 349-359, (2020), https://doi.org/10.1007/978-3-030-17065-3_35.
12. Elderman, R., Pater, L.J.J., Thie, A.S., Drugan, M.M., Wiering M.M.: Adversarial reinforcement learning in a cyber security simulation, In: Proc. International Conference on Agents and Artificial Intelligence (ICAART 2017), pp. 559-566 (2017).