CEUR-WS.org/Vol-2965/paper22.pdf

Reinforcement Learning in the Intelligent Robot Control Task

Dmitry Dobrynin[1]

[1] *Federal Research Center "Computer Science and Control" of the RAS, Vavilova street., 44, cor. 2, Moscow, 119333, Russia*

Abstract

The article describes an approach to reinforcement learning in the task of controlling an intelligent robot. The robot control system is built using the DJSM method. The DJSM method is a development of the JSM method, applicable in the case of an open world with an unknown number of examples in advance. The article deals with reinforcement learning for the movement of a wheeled robot along the black line on a limited polygon. The simulation of training in a software simulator is carried out. Examples of training results for various target functions are given. It is shown that the training work can be carried out in real time. The dependence of the learning outcomes on the initial conditions is revealed. The analysis of the obtained results is carried out.

Keywords

Robots, machine learning, reinforcement learning

1. Introduction

Currently, research in the field of intelligent robots is rapidly developing, which is associated with great prospects for the use of such systems. One of the important characteristics of an intelligent robot is its adaptation to the changing environment in which the robot operates. The robot's control system must adjust itself to these changes, which implies that it has learning properties.

Learning from pre-classified examples, known as learning with a "teacher", has serious limitations in cases where it is not known in advance how to teach the robot. The robot's ability to learn independently is desirable. In this case, the work is set as a goal, and the robot develops the sequence of actions necessary to achieve the goal itself. The teacher in this case is the environment in which the robot functions, and the punishment and reward are set using target functions. Reinforcement learning refers to such methods.

Currently, reinforcement learning is actively developing for neural networks, especially for deep neural networks [1]. Reinforcement learning is actively developing for robotics ([2]-[4]). The features of such training are the low learning rate, which is being improved by various methods, limitations on the complexity of world models, difficulties in interpreting the results, and the dependence of the quality of training on the learning environment.

For logical methods of artificial intelligence, reinforcement learning is much less developed. An example of using reinforcement learning for decision trees is given in the paper [5].

The author proposed an approach to teaching with a teacher for the dynamic JSM method (DJSM) in robot training tasks [6]. The present work is an attempt to develop this direction towards reinforcement learning.

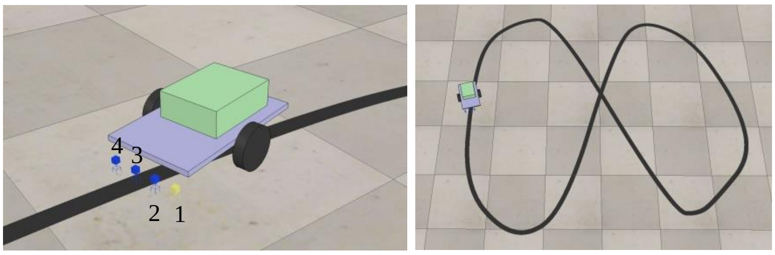
Russian Advances in Fuzzy Systems and Soft Computing: Selected Contributions to the 10th International Conference «Integrated Models and Soft Computing in Artificial Intelligence» (IMSC-2021), May 17–20, 2021, Kolomna, Russian Federation

EMAIL: rabota51@mail.ru (A. 1)
ORCID: 0000-0001-5149-6219 (A. 1)

© 2021 Copyright for this paper by its authors.
Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).
CEUR Workshop Proceedings (CEUR-WS.org)

2. Model task

As a model task for training the robot, the task of moving the robot along the black line was chosen (Figure 1b). This task is often found in mobile robot competitions. Despite its apparent simplicity, it is a good example of a dynamic environment (since the robot itself moves) with a limited number of actions. The robot (Fig. 1a) is a three-wheeled trolley with differential wheels and one support wheel. To detect the line, the robot has four sensors. To simplify the observation of the robot, the first sensor is highlighted in yellow. In Fig .1a digits indicate the sensor numbers.



a) b)

Figure 1: a) robot's model, b) test site for robot.

The current state of the robot is determined by the sensor information - the state of the line sensors, and the control vector - the state of the robot drives. In terms of the DJSM system [6], the current state is one example that has the following representation:

$$P_i = (S_i, C_i) \tag{1}$$

where S_i is the sensor information vector, which includes all the current information about the state of the sensors,

C_i is the control vector, which includes all the current information about the status of the drives.

Each line sensor is encoded with two bits: {1 0} - does not see the line, {0 1} - sees the line. Crossing over the sensor number indicates the inversion of the state. This representation is used in the intersection operation on bit strings. Figure 2 shows an example of a possible representation of one example.

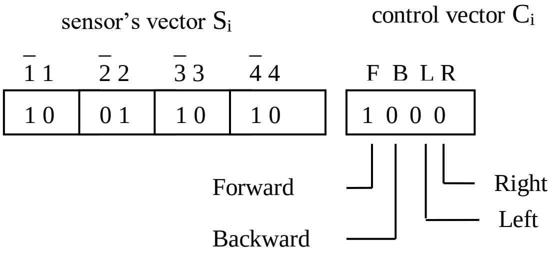


Figure 2: Structure of examples P_i and hypotheses.

The control vector can contain a unit only in one position corresponding to a certain robot movement. This model uses four predefined robot movements: Forward, Backward, Left, and Right. These movements are enough to complete the task of moving along the line. For convenience, the structure of the DJSM hypothesis has the same form as the example.

3. Learning with a teacher in the DJSM method

The dynamic JSM (DJSM) method was proposed in [6]. In contrast to the classical JSM method ([7]-[10]), the dynamic JSM method can be used in an open environment with incomplete information and with an unknown number of examples. In the course of the dynamic JSM method, the knowledge base is open and is formed during the training of the robot, i.e., as new information becomes available. This property of the dynamic JSM makes it possible to use this method to build control systems for an intelligent robot capable of learning.

Let's briefly consider the modes of learning with a teacher and the mode of autonomous work using the DJSM method (see [6], [11]). This will allow us to move on to self-study in the future, which turns out to be similar to learning with a teacher.

In the training mode (see Figure 3a) the "teacher" receives sensory input and generates the control signals necessary for the robot's adequate behavior on the training ground. At the same time, the sensor information and the status of the drives are fed to the DJSM system. A set of receptor signals and current control actions determines one training example P_i. The DJSM system checks it for uniqueness and enters it into its database of facts. After entering a new example in the list of training examples, a hypothesis search is performed. For this problem, a simple DJSM method was used without prohibiting counterexamples (see [11]), since there are no negative training examples in this problem.

Training should be carried out until the knowledge base is no longer replenished with new hypotheses. Obviously, in this case, the training algorithm has gone through all the possible input effects that it is able to respond to, and we can assume that the database of facts is quite complete.

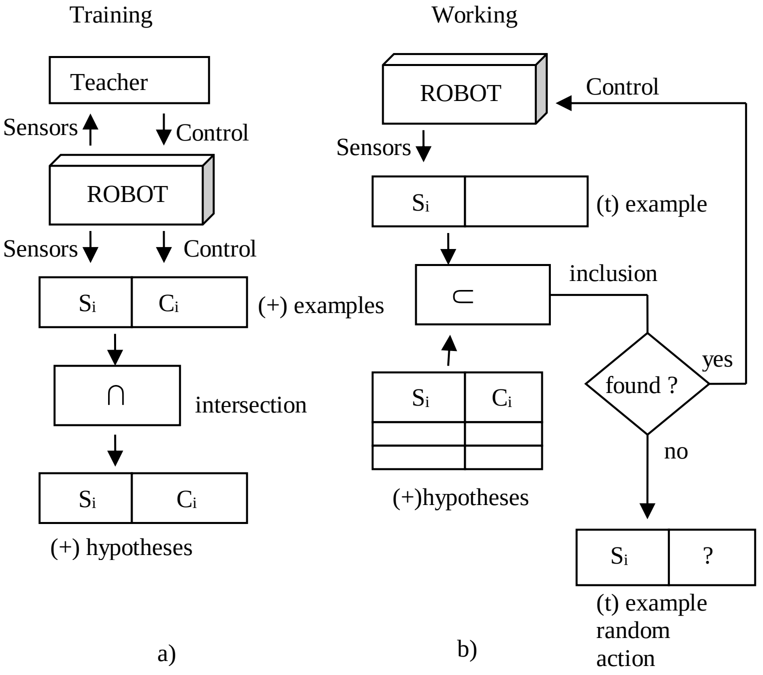


Figure 3: a) training mode, b) working mode.

In the working mode (see Figure 3b), the DJSM system receives the input of the S_i sensor information, from which a test example is formed. The decision is made by checking the embedding of previously obtained hypotheses in this vector. If a hypothesis is embedded in the test vector of receptor signals, then the robot must act in accordance with it. If no hypothesis is found, then this is an unknown state, for which you need to form a random control vector (Figure 3b). If the database of facts is complete, then the behavior of the robot in the operating mode under the control of the DJSM system should not differ in any way from the control of the "teacher".

For testing, various experiments were carried out, both on a software simulator and on the Amur mini-robot created in the Creative Scientific and Technical Laboratory of the Polytechnic Museum [12]. Experiments have shown that this way you can train the robot to behave in real time. The trained robot did not look any different in behavior from the control with the help of a "teacher".

4. Reinforcement learning in the DJSM method

The above method of teaching with the teacher of the DJSM system can be called batch. With this method, the learning and application of the acquired knowledge are separated in time. To switch to self-learning, it is necessary to classify unrecognized examples in the working mode in some way. Additionally, you should introduce rules for encouraging and punishing the robot's actions in the environment.

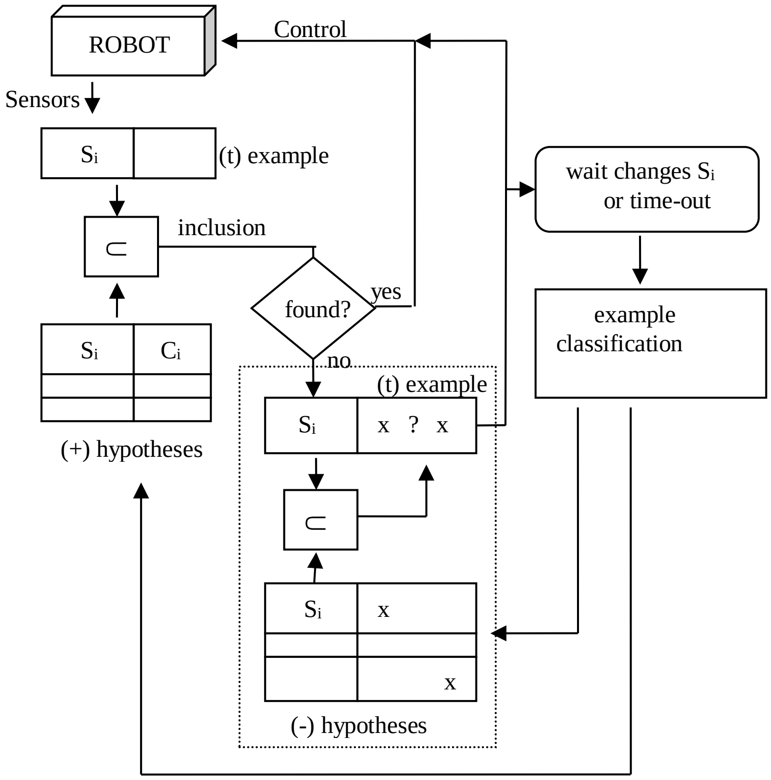


Figure 4: Reinforcement learning of the DJSM system.

Let the base of the robot's (+)hypotheses be empty at the initial time (Fig 4). Then the first example formed from the sensory information will not be recognized and the control system will switch to the classification mode of the (t)example. For a new unknown example, a control vector must be randomly generated. At the same time, negative hypotheses are first searched for, which are embedded in the current sensor vector. If such hypotheses are found, the corresponding actions in the control field are marked as invalid. From the remaining fields of the control vector, one action is randomly selected. Thus, for the current state of the sensor vector, an action will be randomly generated that has not previously led to incorrect results.

The generated control vector controls the robot and then the DJSM system waits for the S_i sensor vector to change. At the same time, it starts to work out a timeout. When the sensor vector changes or a timeout is triggered (the allotted time ends) the DJSM system proceeds to classify the example.

5. Classification of new states

To determine the quality of the action, you need to enter a Target function, which will give information about whether the robot has achieved the goal or not. The achievement of the goal is determined by the sensor vector. The objective function is defined as:

$T(S_i) = 1$ if the goal is reached,

$T(S_i) = 0$ if the goal is not reached.

The classification of examples for the sensor vector Si is carried out as follows:

1) $S_i <> S_i'$, $T(S_i) = 0, T(S_i') = 0$, example → $P(-)$
2) $S_i <> S_i'$, $T(S_i) = 0, T(S_i') = 1$, example → $P(+)$
3) $S_i <> S_i'$, $T(S_i) = 1, T(S_i') = 0$, example → $P(+)$
4) $S_i <> S_i'$, $T(S_i) = 1, T(S_i') = 1$, example → $P(+)$
5) $S_i = S_i'$, timeout = *true*, example → $P(-)$, robot → home

Let’s take a closer look at the classification rules.

In rule 1, the robot moved out of the line ($T(S_i) = 0$) and did not find the line ($T(S_i’)=0$), so this example will be classified as negative examples ($P(-)$).

In rule 2, the robot moved out of the line ($T(S_i)=0$) and found the line ($T(S_i’)=1$), so this example will be attributed to positive examples ($P(+)$).

In rule 3, the robot was moving in the line ($T(S_i)=1$) and lost the line ($T(S_i’)=0$), so this example is referred to as positive examples ($P(+)$). The loss of the line in this case is not a bad thing, since it could have occurred due to circumstances beyond the robot’s control – for example, the robot was moving straight along the line, but met a turn.

In rule 4, the robot was moving in the line ($T(S_i)=1$) and did not lose the line ($T(S_i’)=1$), so this example is referred to as positive examples ($P(+)$).

Rule 5 works in cases where the robot performs an action for a long time that does not lead to a change in the state. Then the robot should be returned to its original state (home), and the action should be classified as incorrect ($P(-)$) (leading to a line).

After classifying an example, i.e. assigning it to positive or negative examples, a hypothesis search is performed. Thus, the knowledge bases of (+)examples and (-)examples are updated with new information. The criterion for completing the self-learning process of the robot is the absence of unknown (unclassified) examples in the operation of the control system.

6. Simulation results

To test the proposed method, a robot simulator was used, in which training was conducted (Fig. 1b). The robot started moving out of the line, got into the line, and then in the process of self-learning, typed hypotheses.

As the target function, we used various conditions for the robot to be in the line. Here are some examples.

Example 1.

Objective function *T(Si)*:

```
    if   Photo2   then true
                  else  false
    endif
```

If the photo sensor 2 sees the line, the function returns *true*, otherwise it returns *false*.

It can be assumed that to implement such an objective function, it is sufficient to turn left if the photo sensor 2 sees the line. And turn right if the photo sensor 2 does not see the line. However, contrary to expectations, only some of these hypotheses were generated during the experiments.

Table 1

Hypotheses for the experiments 1-3

	Experiment 1	Experiment 2	Experiment 3
	$\bar{1}1\bar{2}2\bar{3}3\bar{4}4$ FBLR	$\bar{1}1\bar{2}2\bar{3}3\bar{4}4$ FBLR	$\bar{1}1\bar{2}2\bar{3}3\bar{4}4$ FBLR
(+)	01001010 0001	10010010 0010	00010110 0010
examples	10010010 0010	00100010 0001	00010000 0001
	00010000 0001	00010000 1000	00000010 0001
		10000101 1000	
(-)	10100000 0010	10100001 0001	10100001 0001
examples	10100001 1000		10100001 0010
	10100000 0001		10100001 1000
	10101000 1000		00101001 1000
			01100001 1000
result	good	good	bad

According to the results of self-study, various variants of positive and negative hypotheses are formed in several experiments (Table 1). In the first experiment, only right and left movements are formed. This set of rules allows the robot to move steadily along the line. As can be seen from Table 1, in this variant, there are actions for which the target function gives *false*-this is hypothesis 1. In this case, the robot sees the line with sensor 1, although the target function requires the robot to see the line with sensor 2.In the second experiment (Table 1), the set of control rules contains left and right turns and straight movement, which allows the robot to move steadily along the line. Note that even in this case, there are hypotheses for which the objective function gives *false*. In the third experiment (Table 1), the set of rules does not allow the robot to move steadily in the line, although there are left and right turns. The set of negative hypotheses in this experiment is the largest of the presented ones.

The final result of the training and the differences in the composition of the hypotheses strongly depend on the initial conditions – the starting point before entering the line, the action that the robot performed during this process, and the state of the random number sensor that was used to select random options.

<u>Example 2.</u>

Objective function *T(Si)*:

```
if  Photo1 OR Photo2 OR Photo3 OR Photo4
        then true
        else  false
endif
```

If any of the photo sensors sees the line, the function returns *true*, otherwise it returns *false*.

The results of some experiments are shown in Table 2. As for the first example, different sets of hypotheses are formed with different final results.

Table 2

Hypotheses for the experiments 4-6

	Experiment 4	Experiment 5	Experiment 6
	$\bar{1}1\bar{2}2\bar{3}3\bar{4}4$ FBLR	$\bar{1}1\bar{2}2\bar{3}3\bar{4}4$ FBLR	$\bar{1}1\bar{2}2\bar{3}3\bar{4}4$ FBLR
(+) examples	10000000 0010 01000000 0001	00010000 1000 00100000 0010	01000000 0001 00000101 0001 00100000 0001 00010000 0010
result	good	bad	good

In Experiment 4 (Table 2), an almost ideal result was obtained for controlling a single photo sensor. In Experiment 5 (Table 2), bad rules were obtained that do not allow the robot to move steadily along the line. In Experiment 6 (Table 2), the last two rules are the ideal rules for moving the robot along the line using the photo sensor 2.

An interesting fact is the absence of negative hypotheses in these experiments. We can assume that this is due to the fact that the target function is very “soft”, since any sensor can see the line. Therefore, incorrect behavior as a result of training did not occur, as a result of which negative examples and hypotheses were not generated.

The simulation showed a high speed of the learning process. So for all the experiments, a sufficient number of hypotheses were collected when the robot moved in one circle. The time spent on training was less than one minute.

7. Conclusion

Experiments on the simulator showed the possibility of training with reinforcement for the DJSM method. The quality of training is highly dependent on the initial learning conditions, which does not always lead to good learning outcomes. This property of reinforcement learning is also observed for neural networks, which indicates the deep properties of learning that have not yet been sufficiently studied.

A feature of reinforcement learning for the DJSM method is the high learning rate. In practice, the robot can be trained in real time while working on the test site. Since two similar examples are enough to get the DJSM hypothesis, the knowledge base is set in several similar cases.

The proposed approach to training with reinforcement for the DJSM method can be used to build effectively trainable control systems for intelligent robots. Obviously, this approach needs to be developed in order to reduce the dependence of the quality of training on the initial conditions.

8. References

[1] Richard S. Sutton, Andrew G. Barto Reinforcement learning: an introduction. Second edition//MIT Press, Cambridge, MA, 2018.

[2] Kober J., Bagnell J., Peters J. Reinforcement Learning in Robotics: A Survey. The International Journal of Robotics Research, July, 2013, pp. 1238-1274. doi:10.1177/0278364913495721.

[3] : Liu R.; Nageotte F.; Zanne P.; de Mathelin M.; Dresp-Langley B. Deep Reinforcement Learning for the Control of Robotic Manipulation: A Focussed Mini-Review. Robotics 2021, 10, 22. doi:10.3390/robotics10010022

[4] Schaal S., Atkeson C. (2010). Learning Control in Robotics. Robotics & Automation Magazine, IEEE, vol 17., pp. 20 - 29. doi:10.1109/MRA.2010.936957.

[5] A. Garlapati, A. Raghunathan, V. Nagarajan, B. Ravindran. A Reinforcement Learning Approach to Online Learning of Decision Trees. Technical report, Department of Computer Science, Indian Institute of Technology, Madras, 2015.

[6] Dobrynin D. A. Dynamic JSM-method in the problem of intelligent robot control.// Tenth national conference on artificial intelligence CII-2006, September 25-28, 2006, Obninsk, Proceedings of the conference, M: Fizmatlit 2006, vol. 2., pp. 695-699.

[7] Finn V. K. Plausible reasoning in JSM type intelligent systems //Results of science and technology. Ser. "Informatics". Vol. 15. - M.: VINITI, 1991.

[8] Finn V. K. Automatic generation of hypotheses in intelligent systems/ed by V.K. Finn - Stereotype Publishing House. URSS. 2020. 528 p. ISBN 978-5-397-07090-4.

[9] DSM-method of automatic generation of hypotheses: Logical and epistemological bases / Comp. O. M. Anshakov, E. F. Fabrikantova; ed. by O. M. Anshakov. - M.: LIBROCOM, 2009-433 p.

[10] Vinogradov, D.V. Machine learning based on similarity operation // Communications in Computer and Information Science, Vol. 934. - 2018. - pp. 46-59

[11] Dobrynin D. A. The use of a learning control system for the implementation of robot movement// 15th International Conference on Electromechanics and Robotics "Zavalishin's Readings" ER(ZR)-2020, Proceedings of the conference youth section, SPb.: GUAP, 2020, pp. 128-134. ISBN 978-5-8088-1503-2

[12] Dobrynin D. A., Karpov V. E. Modeling of some simplest forms of behavior: from conditioned reflexes to inductive adaptation//First International Conference "System Analysis and Information Technologies SAIT-2005", Moscow: KomKniga, 2005, vol.1., pp. 188-193