A comparison of exploration strategies used in reinforcement learning for building an intelligent tutoring system

Jezuina Koroveshi[a], Ana Ktona[b]

[a] *University of Tirana, Faculty of Natural Sciences, Tirana, Albania*
[b] *University of Tirana, Faculty of Natural Sciences, Tirana, Albania*

Abstract

Reinforcement learning is a form of machine learning where an intelligent agent learns to make decisions by interacting with some environment. The agent may have no prior knowledge of the environment and discovers it through interaction. For every action that the agent takes, the environment gives a reward signal that is used to measure how good or bad that action was. In this way, the agent learns which are more favorable actions to take in every state of the environment. There are different approaches to solve a reinforcement learning problem, but one drawback that arises during this process is the tradeoff between exploration and exploitation. In this work we focus on studying different exploration strategies and compare their effect in the performance of an intelligent tutoring system that is modeled as a reinforcement learning problem. An intelligent tutoring system is a system that helps in the process of teaching and learning by adapting to student needs and behaving differently for each student. We train this system using reinforcement learning and different exploration strategies and compare the performance of training and testing to find which is the best strategy.

Keywords

Reinforcement learning, exploration strategies, intelligent tutoring system

1. Introduction

Intelligent educational systems are systems that apply techniques from the field of Artificial Intelligence to provide better support for the users of the system [1]. Web-based Adaptive and Intelligent Educational Systems provide intelligence and student adaptability, inheriting properties from Intelligent Tutoring Systems (ITS) and Adaptive Hypermedia Systems (AHS) [2]. [3] defines an Intelligent Tutoring System (ITS) as computer-aided instructional system with models of instructional content that specify what to teach, and teaching strategies that specify how to teach.

Traditional tutoring systems use the one-to-many way of presenting the learning materials to the students.

In this approach every student is given the same materials to learn regardless of his/her needs and preferences. These systems are not well suited for all students because they may come from different backgrounds, may have different learning styles and do not absorb the lessons with the same peace. An intelligent tutoring system customizes the learning experience that the student perceives by taking into consideration factors such as pre-existing knowledge, learning style and student progress. According to [4] an intelligent tutoring system usually has the following modules: the student module that manages all the information related to the student during the learning process; the domain module that contains all the information related to the knowledge to teach, such as topics, tasks, relation between them, difficulty.; the pedagogical module, also called tutor module that decides what, how and when to teach the learning materials.; the graphical user interface module that facilitates the communication between the system and the student. Different

Proccedings of RTA-CSIT 2021, May 2021, Tirana, Albania
EMAIL: jezuina.koroveshi@fshn.edu.al (A.1);
ana.ktona@fshn.edu.al (A.2);

techniques from artificial intelligence can be applied in order to make these systems more "intelligent", but our study is focused on the use of reinforcement learning (RL).

Reinforcement learning is a form of machine learning that is based on learning from experience. The learner is exposed to some environment, for which he may or may not have information, starts making decisions and gets some feedback that gives information telling how good or bad that decision was. Based on the feedback from the environment the learner learns which decisions are more favorable to take. This class of machine learning has been used in modeling and building intelligent tutoring systems such as in the works from [5], [6], [7], [8], [9], [10].

The remainder of this paper is organized as follows: in section 2 we give an overview on reinforcement learning, in section 3 we describe the model that we have used to build an intelligent tutoring system, in section 4 we give the experimental results of training the model using different exploration strategies and is section 5 we give the conclusions of our work.

2. Reinforcement learning

Reinforcement learning is a form of machine learning in which the learner learns some sequence of actions by interacting with the environment. The learner is in a state of the environment, takes some action that moves it from that state to another and after each action the environment gives a reward signal. This reward signal is used to learn which are the best states to be in, and therefore learn which action to take in order to go in those states. A reinforcement learning problem can be modeled as a Markov Decision Process (MDP). A MDP is a stochastic process that satisfies the Markov Property. In a finite MDP, the set of states, actions and rewards have a finite number of elements. Formally, a finite MDP can be defined as a tuple M = (S, A, P, R, γ), where:

- S is the set of states: S = (s1, s2, …, sn).
- A is a set of actions: A = (a1, a2, …, an).
- $\gamma \in [0,1]$ is the discount factor and is used to control the weight of the future reward in comparison to immediate rewards.
- P defines the probability of transitions from s to s' when taking action a in state s:
  $Pss' = Pr\{st+1 = s' \mid st = s, at = a\}$
- R defines the reward function for each of the transitions, the reward we get if we take action a in state s and end up in state s': $Rss' = E\{rt+1 \mid st = s, at = a, st+1 = s'\}$

The goal of the agent is to maximize the total reward it receives. The agent should maximize the total cumulative reward it receives in the long run, not just the immediate reward [11]. The expected discounted reward is defined as follows by [11]:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

The sequence of states that end up in a terminal state is called an episode. The general process of RL may be defined as follows:

1. At each time step t, the agent is in a state s(t).
2. The agent choses one of the possible actions in this state, a(t), and applies that action.
3. After applying the action, the agent transitions in a new state s(t+1) and gets a numerical reward r(t) from the environment.
4. If the new state is not terminal, the agent repeats the step 2, otherwise the episode is finished.

2.1 Exploration and exploitation dilemma

One challenge of reinforcement learning is the tradeoff between exploration and exploitation [11]. As given by [11]: "To obtain a lot of reward, a reinforcement learning agent must prefer actions that it has tried in the past and found to be effective in producing reward. But to discover such actions, it has to try actions that it has not selected before. The agent has to exploit what it has already experienced in order to obtain reward, but it also has to explore in order to make better action selections in the future". There are different strategies that can be used to handle this problem:

1. Random policy: during the training process the agent always chooses random actions. This means that it always explores and does not exploit what it has already learned.
2. Greedy policy: during the training process the agent always chooses the action that gives the best reward. In this way, it is always exploiting the knowledge that has gained and uses it to choose the action that gives the best reward.
3. Epsilon-greedy: this method balances the tradeoff between exploration and exploitation. With probability epsilon ε it chooses a random action, and with probability 1- ε it chooses the best action. The epsilon value decreases with time reducing exploration and increasing exploitation in order to make use of the knowledge is gained.
4. Boltzmann (soft-max) exploration: one problem of the epsilon-greedy method is that the exploration action is selected uniform randomly from the set of actions. This means that it is equally likely to choose the worst appearing action and the second-best appearing action. The Boltzmann exploration uses the Boltzmann distribution [12] to assign a probability to the actions Pt(a):

$$P_t(a) = \frac{exp(Q_t(a)/\tau)}{\sum_{i=1}^{n} exp(Q_t(i)/\tau)}$$

T is a temperature parameter. When T=0 the agent does not explore at all, and when $T \rightarrow \infty$ the agent selects actions randomly.

3. Proposed model

The model that we propose focuses on the pedagogical module of the intelligent tutoring system. This is a system for teaching lessons of Python programming language based on concepts and student knowledge. The learning material is composed of lessons. Every lesson teaches some concepts and may require some previous concepts to be known by the student. In [14] we give a definition of lessons, concepts, student knowledge and how they are related to each other. The student starts learning the course material. The system gives the student a lesson that teaches some concepts. Depending on the student ability to learn, he/she may learn these concepts or not. If the student does not learn all the concepts given by the current lesson, the system cannot give him/her a new lesson. So, the system should make sure that the student has absorbed all the material given by the current lesson before giving the next one. We propose the use of reinforcement learning to train the pedagogical module that based on student knowledge and the concepts that are taught by each lesson to decide what lesson to give him/her. The system will start by giving the first lesson, and then following the student progress will give every other lesson until the end of the course. To model this as a reinforcement learning problem, we need to define the set of states, actions and rewards. In [15] we have given a definition of those elements that create a framework for doing the training using reinforcement learning approach. One problem that arises when dealing with reinforcement learning is the fact that in order to do the training, it is required a relatively large number of iterations and data. This cannot be achieved using real students, because the process would be very long. In [15] we have proposed the use of a simulated student that can be used during the training process. The student has some ability to learn which is given in the form of a learning probability, and this defines his/her ability to learn every concept that is taught by the lessons of the course.

4. Experimental results

We have done the training in a simulated environment by simulating the behavior of the student. For every episode the student starts with knowing random concepts, and the system tries to learn what is the next lesson to give. We have used the DQN algorithm as given by [13], using memory replay and target network. Figure 1 gives the architecture of the target and train networks.

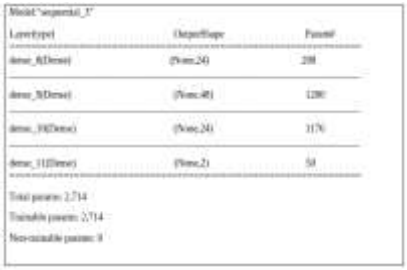

Figure 1: The architecture of the neural network

The hyper parameters used during the training are given in the Figure 2.

Train frequency (after how many episodes will happen the training)	20
Maximal number of steps in episode	100
Learning probability of the simulated student	0.7
Learning rate	0.005
Batch size	64
Gamma	0.85
Maximal memory size	5000
Number of episodes	4000
Epsilon decay (the values used to decrease epsilon)	0.9995
Epsilon minimum	0.01
Epsilon decay frequency (after how many episodes decays the value of epsilon)	100

Figure 2: The hyper parameters used during training/testing

The training is done using different exploration strategies for the same number of episodes. For each of the strategies we give the total reward received for every episode during the training process in figures 3, 4, 5, 6.

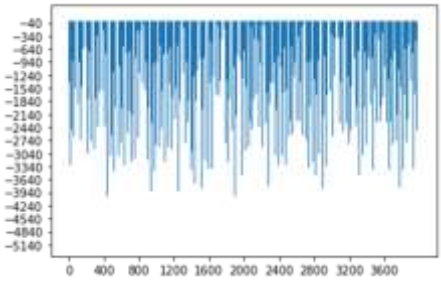

Figure 3: Reward per episode for random strategy

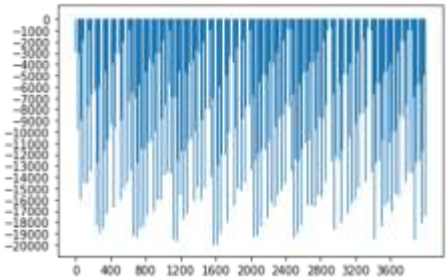

Figure 4: Reward per episode for greedy strategy

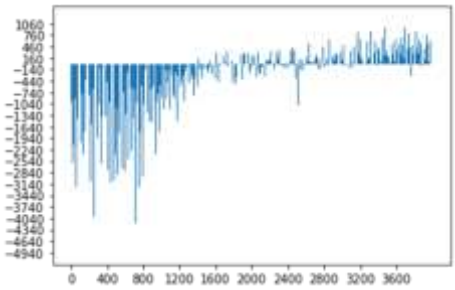

Figure 5: Reward per episode for epsilon-greedy strategy

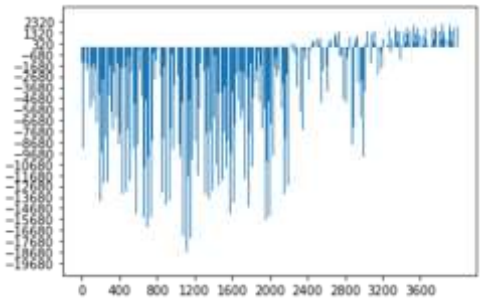

Figure 6: Reward per episode for Boltzmann strategy

4.1. Testing

After we performed the training, we have tested the performance of each of the models learned by using them in simulations, for 100 episodes with a student that knows random concepts and learning probability the same as the one used during the training process. For each of the tests, we show the total reward received and the length for each episode of the training in figures 7 to 14.

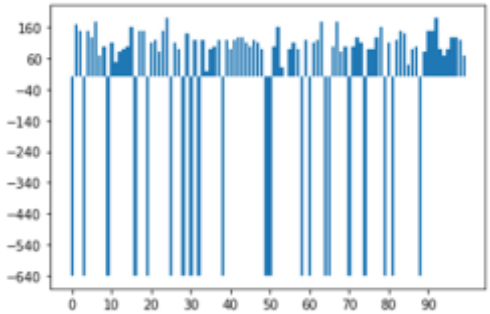

Figure 7: Reward per episode in testing random strategy

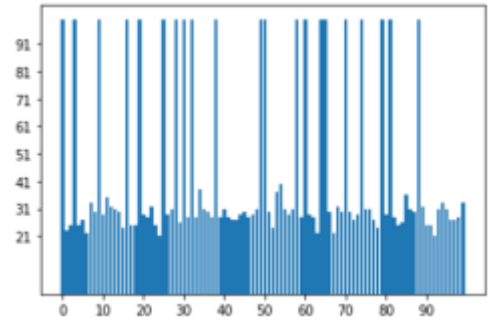

Figure 8: Episode length in testing random strategy

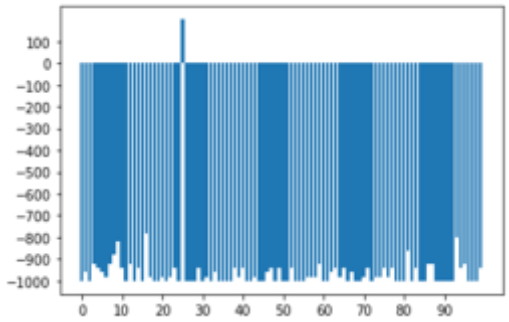

Figure 9: Reward per episode in testing greedy strategy

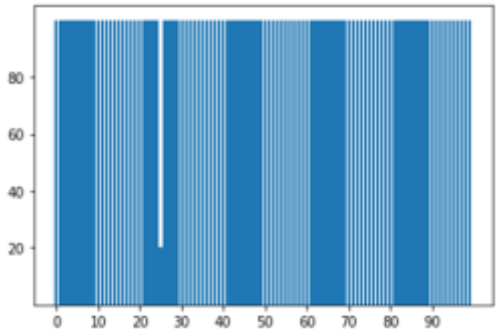

Figure 10: Episode length in testing greedy strategy

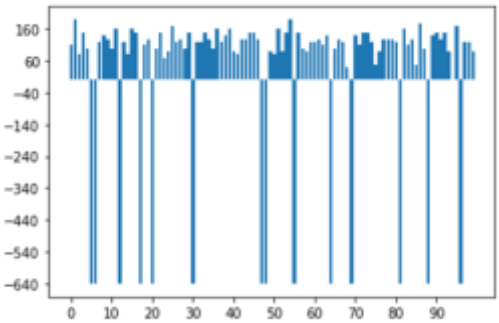

Figure 11: Reward per episode in testing epsilon-greedy strategy

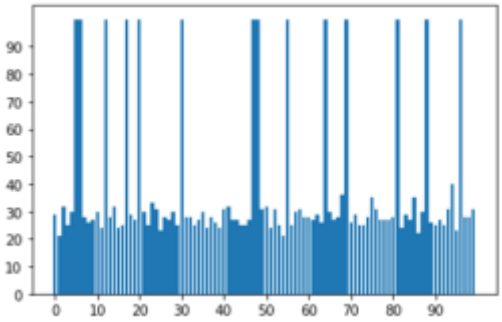

Figure 12: Episode length in testing epsilon-greedy strategy

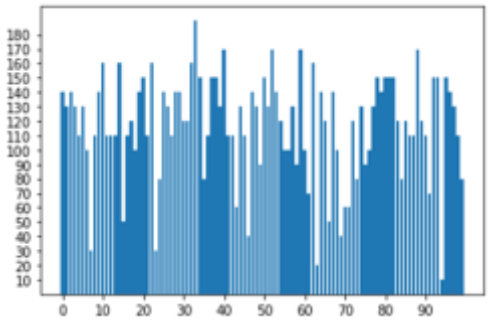

Figure 13: Reward per episode in testing Boltzmann strategy

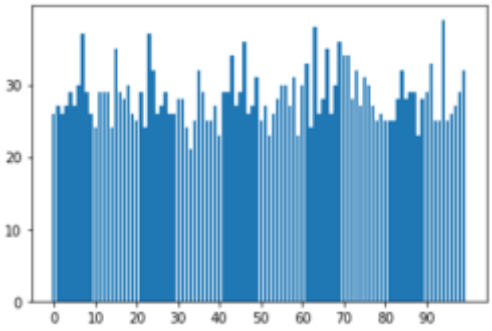

Figure 14: Episode length in testing Boltzmann strategy

5. Conclusion

In this work we have compared the performance of different exploration strategies used in training an intelligent tutoring system using reinforcement learning. We took into consideration 4 strategies: random, greedy, epsilon-greedy and Boltzmann (soft-max). For each of the strategies used, we have considered the reward gained for every episode during the training and testing, to evaluate which one performed better. We saw that during the training phase, random and greedy strategies performed worse.

The reward was negative for every episode, which means that they chose the worst action for most of the time. For the random policy this means that it always explores and never exploits the knowledge. For the greedy policy this means that it always tries to exploit its knowledge, but it never explores for new actions that may be more profitable. On the other hand, the epsilon-greedy and Boltzmann strategies performed best during the training phase, with Boltzmann strategy getting slightly higher rewards. These strategies use a combination of exploration and exploitation, which makes them perform better.

During the testing phase we see that greedy policy performs worse than every other policy. This shows that the system has not learned anything during the training phase. Random and epsilon-greedy policies performed well during the testing phase with almost the same reward gained. Even though random policy performed poorly during the training phase, it did quite well during testing, meaning that the high level of exploration learned some good actions. The Boltzmann policy was the best during the testing phase, getting the highest reward values. This shows that this policy learned better which are the best actions to take. Also, comparing the episode length during the testing phase, Boltzmann strategy has the shortest episode lengths. This shows that it finishes each episode without reaching the episode length limit, meaning that it finishes the episode faster because it takes the right actions.

6. References

[1] Brusilovsky, P. & Peylo, C. (2003). Adaptive and Intelligent Web-based Educational Systems. Inter-national Journal of Artificial Intelligence in Education (IJAIED),13, pp.159-172. {hal-00197315}

[2] Iglesias, A., Martinez, P., & Fernandez, F. (2003). An Experience Applying Reinforcement Learning in a Web-Based Adaptive and Intelligent Educational System. Informatics in Education, 2(2), 223–240. https://doi.org/10.15388/infedu.2003.17

[3] Wenger, E. (1987). Artificial Intelligence and Tutoring Systems. Morgan Kaufman

[4] Burns, H. L. & Capps, C. G. (1988) Foundations of intelligent tutoring systems: an introduction. In Foundations of Intelligent Tutoring Systems (eds M. C. Polson & J. J. Richardson). Lawrence Erlbaum, London, pp. 1–19

[5] Malpani, A., Ravindran, B., & Murthy, H. (2011). Personalized Intelligent Tutoring System using Reinforcement Learning. In Florida Artificial Intelligence Research Society Conference. Retrieved from https://aaai.org/ocs/index.php/FLAIRS/FLAIRS11/paper/view/2597/3105

[6] Martin, K. N., & Arroyo, I. (2004). AgentX: Using Reinforcement Learning to Improve the Effectiveness of Intelligent Tutoring Systems. Intelligent Tutoring Systems, 564–572. https://doi.org/10.1007/978-3-540-30139-4_53

[7] Nasir, M., & Fellus, L. & Pitti, A. (2018). SPEAKY Project: Adaptive Tutoring System based on Reinforcement Learning for Driving Exercizes and Analysis in ASD Children. ICDL-EpiRob Workshop on “Understanding Developmental Disorders: From Computational Models to Assistive Technologies". Tokyo, Japan. ⟨ hal-01976660⟩

[8] Sarma, B. H. S., & Ravindran, B. (2007). Intelligent Tutoring Systems using Reinforcement Learning to teach Autistic Students. Home Informatics and Telematics: ICT for The Next Billion, 241, 65–78. https://doi.org/10.1007/978-0-387-73697-6_5

[9] Shawky, D., & Badawi, A. (2018). A Reinforcement Learning-Based Adaptive Learning System. The International Conference on Advanced Machine Learning Technologies and Applications (AMLTA2018), 221–231. https://doi.org/10.1007/978-3-319-74690-6_22

[10] Wang, F. (2018). Reinforcement Learning in a POMDP Based Intelligent Tutoring System for Optimizing Teaching Strategies. International Journal of Information and Education Technology, 8(8), 553–558.

https://doi.org/10.18178/ijiet.2018.8.8.1098

[11] Sutton, R. S. and Barto, A. G. (2018) Reinforcement Learning: An Introduction (2nd Edition, in preparation). MIT Press.

[12] Barto, A. G., Bradtke, S. J., and Singh, S. P., (1991) Real-time learning and control using asynchronous dynamic programming. University of Massachusetts at Amherst, Department of Computer and Information Science.

[13] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). Human-level control through deep reinforcement learning. Nature, 518(7540), 529–533. https://doi.org/10.1038/nature14236

[14] Koroveshi, J., Ktona, A. (2020). MODELLING AN INTELLIGENT TUTORING SYSTEM USING REINFORCEMENT LEARNING. Knowledge International Journal, 43(3), 483 - 487. Retrieved from https://ikm.mk/ojs/index.php/KIJ/article/view/4745

[15] Koroveshi, J., Ana Ktona. (2021). Training an Intelligent Tutoring System Using Reinforcement Learning. International Journal of Computer Science An Information Technolgy, 19(3), 10-18, http://doi.org/10.5281/zenodo.4661455