TRAINING AN INTELLIGENT TUTORING SYSTEM USING REINFORCEMENT LEARNING

Jezuina Koroveshi

University of Tirana, Faculty of Natural Sciences, Albania,

jezuina.koroveshi@fshn.edu.al

Ana Ktona

University of Tirana, Faculty of Natural Sciences, Albania,

ana.ktona@fshn.edu.al

***Abstract*: In this work we have applied reinforcement learning in building an intelligent tutoring system. An intelligent tutoring system is a computer system that provides personalized learning material to the learner, based on his needs and level of knowledge. Such a system may consist of the following components: the knowledge base, the student's model, the pedagogical module and the user interface. The role of the pedagogical module is to define what is the best learning material to give to the students in order to help them reach their goal towards learning the material of the course. This is a continuation of our previous work that models an intelligent tutoring system as a reinforcement learning problem for teaching different lessons related to Python programming language [1]. In this work, we focus on building the pedagogical module through applying reinforcement learning and the DQN algorithm. To model a problem as a reinforcement learning problem, we should take special care in defining the following components: the state space, the actions and the rewards. Here, we propose a way to organize the state space, the actions and the rewards, in order to train the pedagogical module using reinforcement learning. After defining those elements, we train this module using different parameters and conditions. The training is done in a simulated environment, by simulating the behavior of the student in order to help the training process.**

Keywords: intelligent tutoring system, reinforcement learning, DQN

1. INTRODUCTION

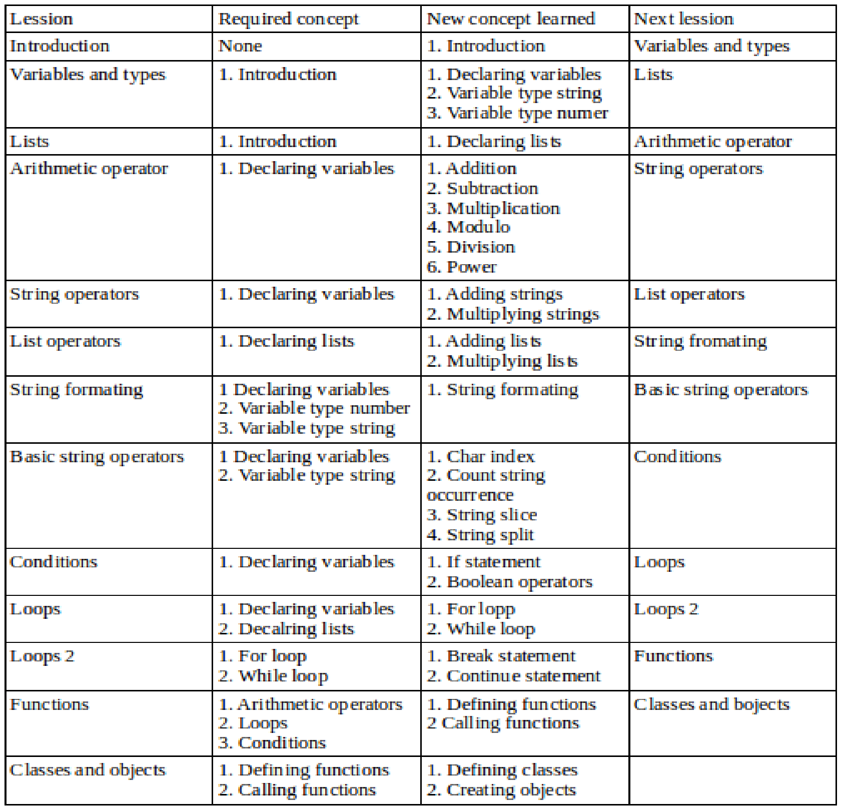
An overview on intelligent turning systems (ITS) is given in our previous study [1]. According to that, an ITS is a system that adapts to every student based on factors such as pre-existing knowledge, learning style and student progress, by making personalized decisions for everyone. In this way, the ITS can customize the learning experience that the students perceive. Principles of artificial intelligence may be applied in the design of such systems in order to make them more intelligent and we focus on applying reinforcement learning for doing so. This class of machine learning algorithms has been applied in different approaches for designing ITS such as in the works from [2], [3], [5], [6], [7], [8].

Here, we extend our previous work for modeling an ITS by focusing on the knowledge factor [1]. The student starts learning the course materials and may or may not have some previous knowledge of the concepts that this course teaches. Each lesson of the course teaches some concepts, and we define an order for the lessons that are to be taught. We assume that the student cannot continue to the next lesson if he/she has not learned all the concepts that are presented in the current lesson. So, it is the duty of the pedagogical module to decide what learning material to give to the student, by determining if he/she should still stay in the current lesson or go to the next one. We model this module using reinforcement learning and then train it to learn what are the best actions to take to help the student learn the course material.

The remainder of this paper is organized as follows: in section 2 we describe the proposed model, in section 3 we describe the simulation that we have done, in section 4 we detail the experiments and the results and in section 5 give the conclusions of our work.

2. PROPOSED MODEL

The model that we propose here is a continuation of our previous work. As explained in [1], the learning material is organized in lessons, where each lesson teaches some new concepts and requires some prior concepts to be known by the student. Here, we have added another relation between the lessons where for each lesson we define what is the next lesson to go. In this way, we create a sequence of lessons that the student must follow in order to complete the learning path. The relation between lessons and concepts is given detailed in Figure 1. The idea presented here is slightly different from the one given in our previous work: before starting the course, the student may or may not have previous knowledge of the concepts that the course teaches; the student starts always from the first lesson; the system should make sure that the student learns all the concepts that are presented in the current lesson before going to the next one.



Lession	Required concept	New concept learned	Next lession
Introduction	None	1. Introduction	Variables and types
Variables and types	1. Introduction	1. Declaring variables 2. Variable type string 3. Variable type numer	Lists
Lists	1. Introduction	1. Declaring lists	Arithmetic operator
Arithmetic operator	1. Declaring variables	1. Addition 2. Subtraction 3. Multiplication 4. Modulo 5. Division 6. Power	String operators
String operators	1. Declaring variables	1. Adding strings 2. Multiplying strings	List operators
List operators	1. Declaring lists	1. Adding lists 2. Multiplying lists	String fromating
String formating	1 Declaring variables 2. Variable type number 3. Variable type string	1. String formating	Basic string operators
Basic string operators	1 Declaring variables 2. Variable type string	1. Char index 2. Count string occurrence 3. String slice 4. String split	Conditions
Conditions	1. Declaring variables	1. If statement 2. Boolean operators	Loops
Loops	1. Declaring variables 2. Decalring lists	1. For lopp 2. While loop	Loops 2
Loops 2	1. For loop 2. While loop	1. Break statement 2. Continue statement	Functions
Functions	1. Arithmetic operators 2. Loops 3. Conditions	1. Defining functions 2 Calling functions	Classes and bojects
Classes and objects	1. Defining functions 2. Calling functions	1. Defining classes 2. Creating objects	

Figure 1. Relation between lessons and concepts

In [1], it is given a clear definition of the set of lessons, concepts and student knowledge. Based on that, we propose here a new set of states, actions and rewards for implementing the pedagogical module as a reinforcement learning problem:

1. State(CurrL, Tc1, Tc2, Tc3, Tc4, Tc5, HasRc1, HasRc2, HasRc3, HasRc4, HasRc5) where:
   a. CurrL has values from the set (0, LI, L2, …, Ln) and shows what is the current lesson given to the student.
   b. TC_i for $i \in [1,5]$ has values from (0, C1, C2, … Cn) and shows what are the 5 concepts that are taught by the current lesson. TC_i may be 0 in cases when the lesson teaches less than 5 concepts. In the implementation, this is translated into a vector with length 5, with values 1 or 0, having the values [0,0,0,0,0] when the lesson teaches no concepts, and values [1,1,1,1,1] when the lesson teaches all 5 concepts.
   c. $HasRc_i$ for $i \in [1,5]$ has value 0 or 1. $HasRc_i$ has the value 1 if the student knows TC_i and 0 otherwise. In the implementation, this is translated into a vector with length 5, with values 1 or 0, having the values [0,0,0,0,0] when the student has not learned any concept from the current lesson, and values [1,1,1,1,1] when the student has learned all 5 concepts of the current lesson.

Example of the state may be as follows:

[1,1,1,1,0,0,0,1,0,0,0,] where:

  a. Element in index 0, with value 1 shows the current lesson is Lesson 1.
  b. Elements from index 1 to 5, show that the lesson 1 teaches only 3 concepts.
  c. Element from 6 to 10, show that the student has learned only the first concept that is taught by this lesson.

2. Action (stay in current lesson, go to next lesson Li) where:
   a. Stay means that the student will be given again the current lesson.
   b. L_i has values from (L1, L2, …. Ln). The system chooses the next lesson to give to the student based on current lesson and what is the next lesson that comes after it
3. Rewards will be:
   a. Negative value (-10) if the student has learned all the concepts that are taught by current lesson and the system gives again this lesson.
   b. Positive value (+10) if the student has learned all the concepts that are taught by current lesson and the system gives the next lesson.
   c. Negative value (-10) if the student has not learned all the concepts that are taught by current lesson and the system gives the next lesson.
   d. Positive value (+10) if the student has not learned all the concepts that are taught by current lesson and the system gives again this lesson.

3. SIMULATION

Training an agent using reinforcement learning requires a very large number of episodes and trials that are not very suitable to perform with real life students. Because of this, we have done the training in a simulated environment, by simulating the behavior of real students. The system is composed of two parts: the tutor and the student. The tutor is the agent that will be trained using the reinforcement learning algorithm and will learn what is the best action to take in respect to the student's current state. The student is a simulation of a real-life student. It has its knowledge base, that consists of concepts that it has learned or not and a parameter-learning probability-that describes its ability to learn new concepts. The learning probability gives the probability for learning a new concept. When the student is given a lesson, for every concept that this lesson teaches, the student learns this concept with probability equal to "learning probability" and does not learn this concept with probability equal to "1 - learning probability". If the learning probability is 1, this means that the student always learns the concepts and if it is 0 the student never learns anything. In our simulation, when the student is given a lesson by the tutor, based on the learning probability, we simulate the fact that the student did or did not learn that concept.

4. EXPERIMENTAL RESULTS

We have used the DQN algorithm as given by [4], using memory replay and target network. Figure 2 gives the architecture of the target and train networks.

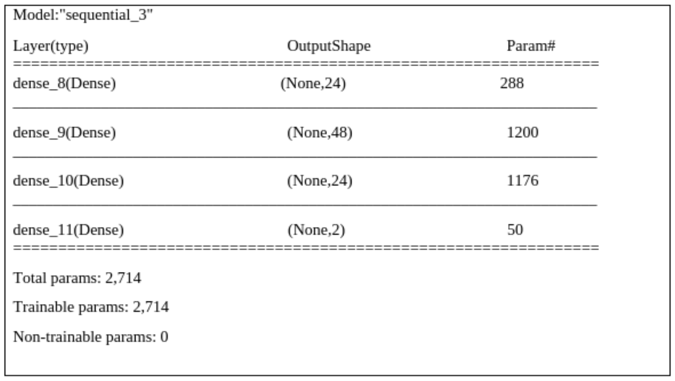

```
Model:"sequential_3"
Layer(type)                 OutputShape              Param#
=================================================================
dense_8(Dense)             (None,24)                288
_________________________________________________________________
dense_9(Dense)              (None,48)                1200
_________________________________________________________________
dense_10(Dense)             (None,24)                1176
_________________________________________________________________
dense_11(Dense)             (None,2)                 50
=================================================================
Total params: 2,714
Trainable params: 2,714
Non-trainable params: 0
```

Figure 2. The architecture of the neural network

We have done the training using two scenarios: 1. the student starts with no knowledge of any of the concepts for every episode, 2. the student starts with knowing random concepts for each episode. Regardless of which scenario is used, there are some hyper parameters that we keep the same and these are shown in Table 1.

TABLE I

HYPER PARAMETERS

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

One important aspect of the training process is the exploration of actions and states regardless of what the agent has learned until now. We have done the training by varying the level of exploration using:

1. Epsilon value that shows the probability with which the agent chooses a random action.
2. Epsilon decay is a value used to decrease the value of epsilon.
3. Minimum epsilon is the minimum value that epsilon can have.
4. Epsilon decay frequency determines after how many steps the value of epsilon will decrease.

The training is done for the two scenarios, by varying the parameters that determine the exploration level. We start with some parameters, perform the training for a certain number of episodes, then change the parameters and restart the training again using the network weights from the previous round. For every training done, as a result we give the total reward earned for each episode. Following we give the trainings that are done for each of the scenarios, the parameters used, and the reward earned.

4.1 Student starts with knowing no concepts for every episode.

We have performed two sets of training for this scenario, by varying the number of episodes for each round.

a. The hyper parameters are given in Table 2.

TABLE 2

HYPER PARAMETERS FOR THE FIRST TRAINING OF THE FIRST SCENARIO

	1st Round	2nd Round	3rd Round	4th Round	5th Round	6th Round
Number of episodes	1000	2000	2000	1000	1000	1000
Initial epsilon	0.9	0.9	0.6	0.2	0.2	0.1
Epsilon decay	0.9995	0.9995	0.9995	0.9995	0.9995	0.9995
Minimum epsilon	0.01	0.01	0.01	0.01	0.01	0.01
Epsilon decay frequency	100	100	50	50	50	50
Results	Figure 3	Figure 4	Figure 5	Figure 6	Figure 7	Figure 8

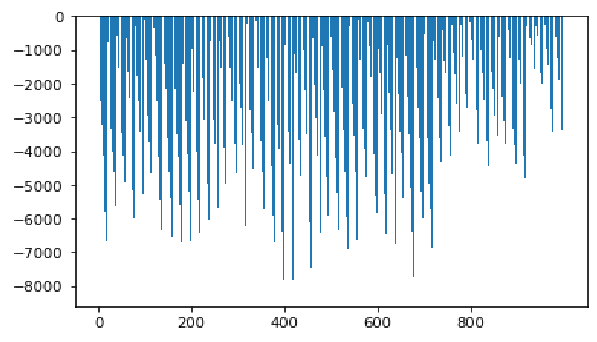


Figure 3.

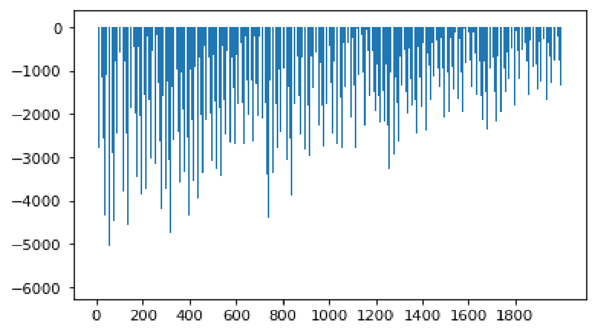


Figure 4.

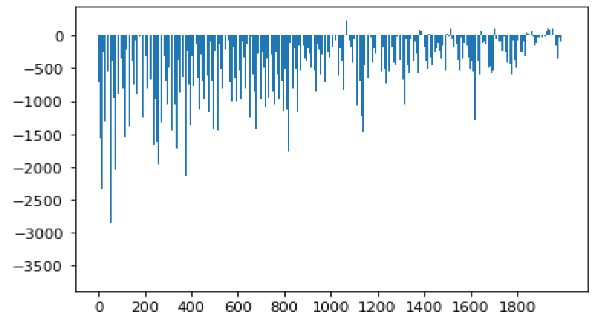


Figure 5.

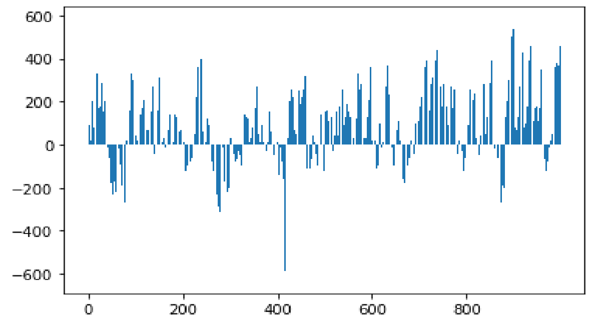


Figure 6.

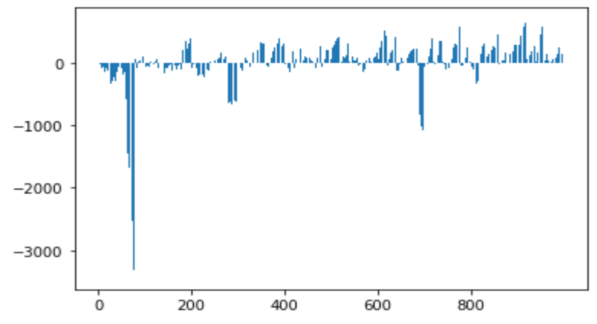


Figure 7.

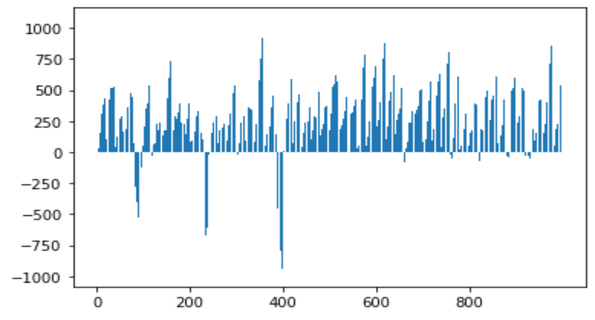


Figure 8.

b. The hyper parameters are given in Table 3.

TABLE 3

HYPER PARAMETERS FOR THE SECOND TRAINING OF THE FIRST SCENARIO

	1st Round	2nd Round	3rd Round
Number of episodes	4000	4000	2000
Initial epsilon	0.9	0.5	0.2
Epsilon decay	0.9995	0.9995	0.9995
Minimum epsilon	0.01	0.01	0.01
Epsilon decay frequency	100	100	100
Results	Figure 9	Figure 10	Figure 11

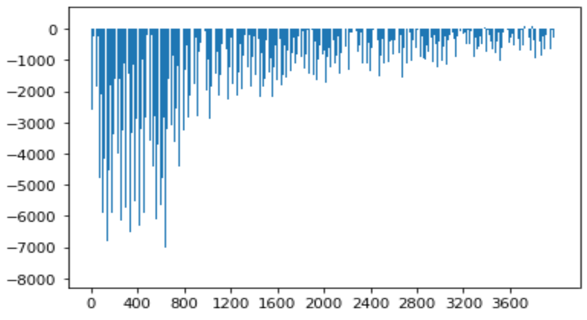


Figure 9.

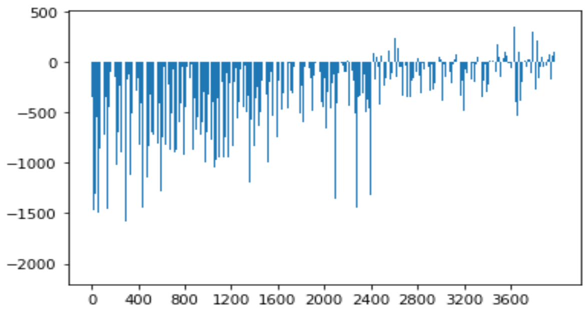


Figure 10.

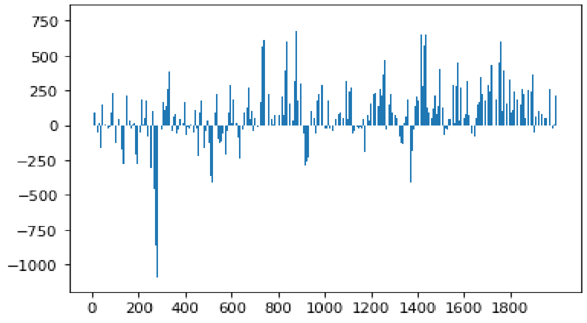


Figure 11.

4.2 Student starts with knowing random concepts for every episode. For this scenario, the hyper parameters are given in Table 4.

TABLE 4

HYPER PARAMETERS FOR THE SECOND SCENARIO

	1st Round	2nd Round	3rd Round
Number of episodes	4000	4000	2000
Initial epsilon	0.9	0.6	0.2
Epsilon decay	0.9995	0.9995	0.9995
Minimum epsilon	0.01	0.01	0.01
Epsilon decay frequency	100	100	100
Results	Figure 12	Figure 13	Figure 14

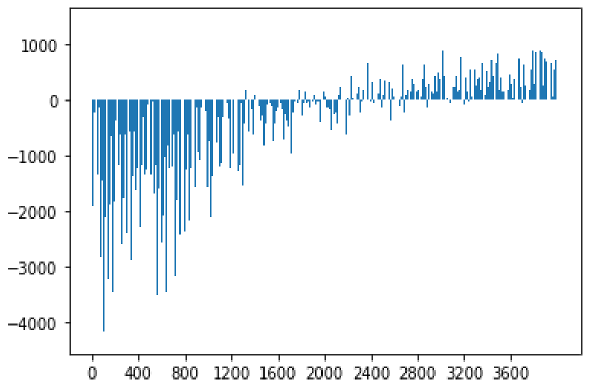


Figure 12.

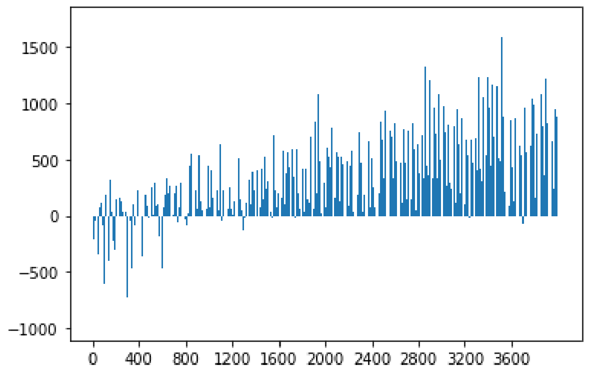


Figure 13.

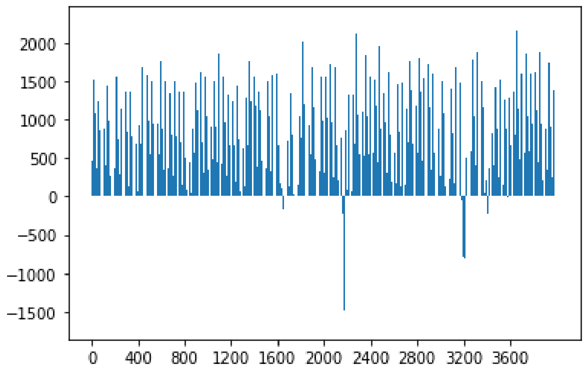


Figure 14.

4.3 Testing

After we performed the training for both scenarios, we have tested the performance of each of the models learned by using them in simulations, for 100 episodes with a student with 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 of each episode. The results are given in the Figures 15 to 20.

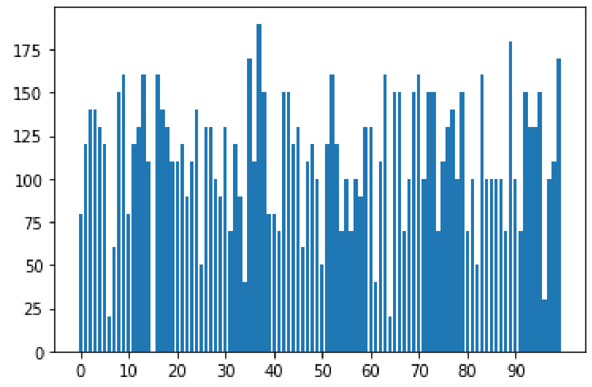


Figure 15. Reward for every episode, for 1st set of the 1st scenario

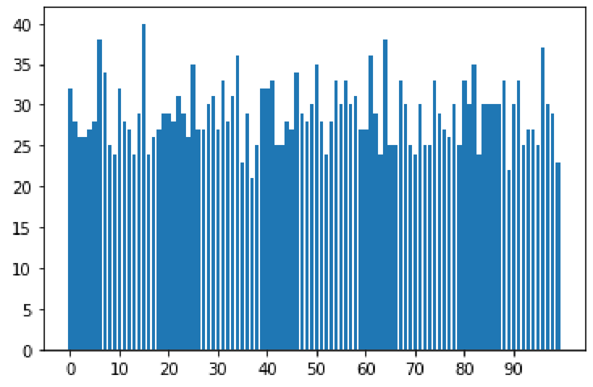


Figure 16. Length for every episode, for 1st set of the 1st scenario

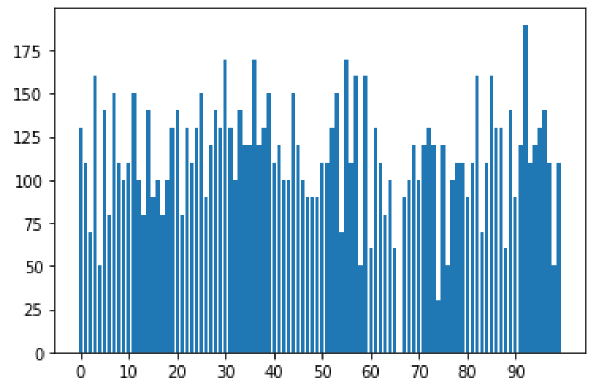


Figure 17. Reward for every episode, for 2nd set of the 1st scenario

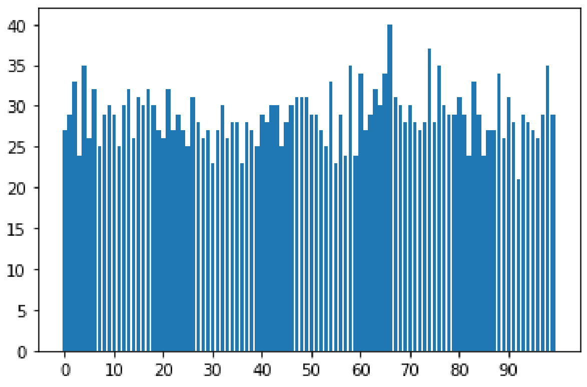


Figure 18. Length for every episode, for 2nd set of the 1st scenario

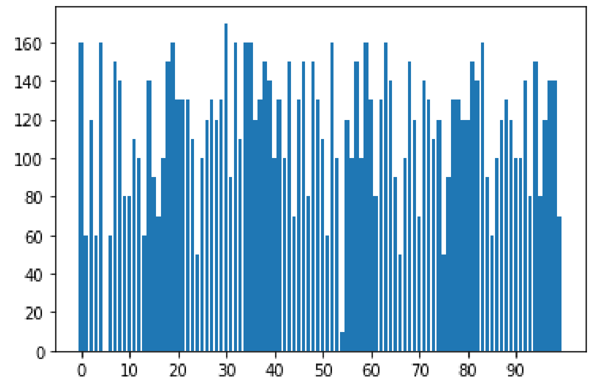


Figure 19. Reward for every episode, for the 2nd scenario

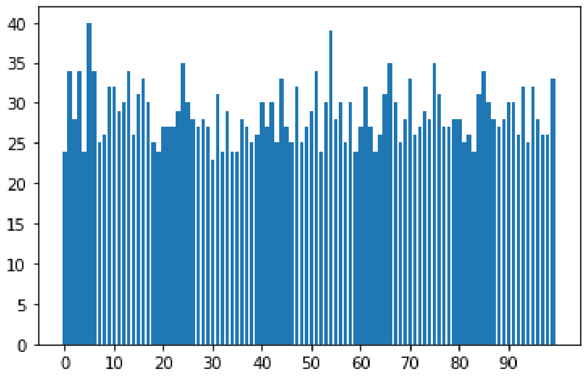


Figure 20. Length for every episode, for the 2nd scenario

5. CONCLUSION

In this work, we have presented a model for an intelligent tutoring system and trained it using reinforcement learning with DQN algorithm. We performed the training in a simulated environment, by simulating the behavior of the student. Also, the training was performed using different hyper parameters for the model and using different scenarios for the student knowledge. As the training and the testing results show, we believe that it is better to perform the training with a student that starts with knowing random concepts for every episode. In this way, there is higher probability to reach more states, compared to the scenario where the student always starts with knowing no concepts. Also, as results show, when using random concepts, the agent starts to get a positive reward earlier in time, meaning that it is learning to take better decisions. Another important factor to keep into consideration is exploration. As we see from the results, it is important to start with high levels of exploration in the beginning of the training and reduce the exploration gradually as the agent starts to learn.

REFERENCES

[1] 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

[2] Malpani, A., Ravindran, B., & Murthy, H. (2011). *Personalized Intelligetn 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*

[3] 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

[4] 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

[5] 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⟩*

[6] 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

[7] 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

[8] 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