MODELLING 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: Intelligent tutoring systems are computer systems that provide personalized instructions to the learner, replacing a human tutor by a machine. They aim to suggest personalized studying strategies taking into consideration student needs, level of knowledge and abilities. An ITS consists of four interacting components: the knowledge base that contains all information to be taught, the student model that contains information related to the student's current knowledge status, the pedagogical module that decides what teaching strategy to use and the interface module that facilitates the communication between the ITS and the student. Researchers have built ITS that teach different subjects such as algebra, equation solving, chemistry, database design etc. To make and ITS more 'intelligent', in their design are applied different artificial intelligence principles. In this work we propose a model that uses reinforcement learning for building ITS that teaches concepts of the python programming language. Reinforcement learning is a form of unsupervised learning in which an agent learns the best policy to act by interacting with the environment through trial and error. After each action that the agent makes, the environment gives a reward that determines if that action was good or bad. The agent has no previous knowledge about the environment, but through interaction and reward it learns what are the best actions. We focus on the pedagogical module of the ITS and model it as a reinforcement learning agent, in order to learn the best strategy for presenting the learning materials to each individual student. The learning materials consist of different lessons related to python programming language. Each lesson teaches some concepts and may require prior knowledge of some other concept. The student decides what concepts wants to learn and may or may not have previous knowledge about those concepts. Taking into consideration this fact, the pedagogical module should determine a strategy to present lessons that teach concepts that the student wants to learn, and make sure that student has acquired knowledge of concepts that are precondition for learning some other concepts. The combination of the concepts that the student wants to learn and concepts that he already knows may be large, so for every student may be needed a different strategy. Because of these, we propose a model that uses reinforcement learning for the pedagogical module so it can be trained to learn the best strategy for presenting the learning materials to each student based on their knowledge and what they want to learn. We propose a way of how to organize the states, actions and rewards that can be used in training this system using reinforcement learning.
Keywords: intelligent tutoring system, reinforcement learning

1. INTRODUCTION

Traditional tutoring systems make use of the one-to-many way of presenting the learning materials to the students, through classroom lectures or online homework. In this approach every student is given the same materials to learn as everybody else, regardless of his needs and preferences. This kind of tutoring systems are not well suited for students that came from different backgrounds, have different levels of knowledge and learning styles and do not absorb the lessons with the same pace. Intelligent tutoring systems differ significantly from their traditional predecessor. Rather than using the one-size-fit-all strategy for delivering the content to the learner, ITS adapt to every student based on factors such as pre-existing knowledge, learning style and student progress. In this way ITS can customize the learning experience that the students perceive. ITS are computer software designed to interact directly with the students and perform functions that traditionally are done by a teacher or a tutor. According to *(Burns & Caps, 1998, pp.1-19)* an ITS 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, task, 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. According to (Grubisic et al., 2015) a system that automatically adapts to the student, based on its assumptions about the student, is referred to as an adaptive system. To make a system adaptable, there are some learner characteristics that need to be taken into consideration such as: student knowledge, learning styles and cognitive abilities. To make an ITS more 'intelligent', in their design may be applied principles from artificial intelligence. In this work we focus on using reinforcement learning for building such systems. Reinforcement learning has been used in different approaches for designing ITS

such as in the works from *(Nasir et al., 2018),* (Shawky & Badawi, 2018, p. 225), (Wang, 2018, p. 554), *(Malpani et al., 2011),* (Sarma & Ravindran, 2007, p. 72), (Martin & Arroyo, 2004, p. 569).

In this work we model an ITS that teaches concepts of the python programming language using reinforcement learning. There are different factors to consider before designing an ITS but our work focuses on the user knowledge factor. This means that users may have prior knowledge about the lessons or may start from the beginning having no knowledge at all. Also, they may choose what they want to learn from the system and skip topics for which they have no interest. Taking these into consideration, the system should determine what learning materials to offer to every student. For example, if a student has some basic knowledge about declaring variables and data types, the ITS should skip those lesson and pass directly to more advanced material. If the student wants to learn some material that requires some prior knowledge about specific concepts the system should give lessons about those concepts first. Since it is the pedagogical module of the ITS responsible for determining what and when to teach something, we focus on the design of that module. Taking into consideration student knowledge and what he wants to learn, we model the pedagogical module as a reinforcement learning that can be trained to learn what is the best strategy of presenting the learning materials to the student based on the knowledge factor.

The remainder of this paper is organized as follows: in section 2 we give a short overview of reinforcement learning, in section 3 we describe the proposed model, in section 4 we give the results and discuss some factors to be taken into consideration and in section 5 give the conclusion of our work.

2. OVERVIEW OF REINFORCEMENT LEARNING

Reinforcement learning is a form of machine learning alongside supervised and unsupervised learning. In this form of learning, an agent learns some sequence of actions only by interacting with an environment. The learning agent may have no prior knowledge of the environment. It takes some action that moves it from one state of the environment to another and after each action the environment gives same kind of reward signal. This reward is used to determine how good or bad is to be on every state. By using this reward function, the agent learns what are the good states or what are the actions that take it to a state with good reward. Only through trial and error by interacting with the environment the agent learns what is the best action to take in each state in order to maximize the reward. According to (Sutton & Barto, 2018a, p. 5) a reinforcement learning system may contain four sub-elements: a policy that defines how the agent behaves at any given time (what action it takes in every state); a reward signal which is sent to the agent from the environment at each time step and is used to define the goal in reinforcement learning.; a value function which indicates how good is a state in the long term, taking into consideration the reward for that state and the rewards of states that are likely to follow; a model of the environment, which may be optional, and is used to make predictions about next states and rewards.

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 $M = (S, A, P, R, \gamma)$, where:

- S is the set of states: $S = (s1, s2, \ldots, sn)$
- A is a set of actions: $A = (a1, a2, \ldots, an)$
- $\gamma \in [0,1]$ 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 accumulated reward it receives by learning to take the actions that give the best rewards for every state.

3. PROPOSED MODEL

The learning material is organized in lessons. Every lesson teaches some concepts and may depend on some other concepts that the student should have learned before. This assumes that the student cannot learn some advanced material if he has not already learned the basics. Also, in this way, the system should make sure that the student has the required knowledge before giving the next lesson. We have organized the learning materials in lessons and for every lesson have defined what are the concepts that it teaches and what are the required concepts that the student should already know. The relation between lessons and concepts is given in Table 1. An example of the relation between lesson and concepts would be as follows. The student wants to learn about loops (for loop, while loop), so

the system should give the lesson “Loops", but these requires that the student already knows how to “Declare variables” and “Declare list” since in this way the loop concepts may be explained better. If the student does not know how to declare variables and list, the system cannot give him the lesson “Loops” but should give him the lesson “Variables and types” and “List” first, since these teach the required concepts for learning loops. After the student learns about “Variables and types” and “List” he can pass on learning about loops.

Table 1. Relation between lessons and concepts

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

Cleary there exists a relation among what the student wants to learn, what he already knows and what he should learn in order to reach his goal. The system should give him only the necessary lessons to help him reach his goal. The model that we propose can be used for training an agent (that will be the tutor module of the ITS) with reinforcement learning, to learn what is the best sequence of lessons to give to the student, until he reaches his goal.
We make some the following definitions about lessons and concepts:

1. The full set of lessons offered by the course is (L1, L2, …. Ln), in our example will be 13 lessons.
2. The full set of concepts related to lessons is (C1, C2, … Cm).
3. For very lesson Li we assume that there may be at most 5 required concepts (RC1, RC2, RC3, RC4, RC5), and at most 5 new concepts that it teaches (NC1, NC2, NC3, NC4, NC5). If a lesson has more than 5 required concept or teaches more than 5 new concepts, it will be better to break this lesson in 2 ore more lessons with no more than 5 concepts. In this way, as will explain latter, we can keep less variables for defining the state.
4. For every student there is the StudentKnowledge($KnowsC_i$,... $KnowsC_m$) where $KnowsC_i$ is 1 if the student knows concept C_i and 0 otherwise.

We propose the following set of states, actions and rewards.

1. State($ReqL_1$,….$ReqL_n$, CurrL, Rc1, Rc2, Rc3, Rc4, Rc5, HasRc1, HasRc2, HasRc3, HasRc4, HasRc5) where:

a. $ReqL_i$, has value 0 or 1, that for every lesson that the course offers, shows if the student has required to learn it(1), or has not required to learn it(0).
b. CurrL has values from the set (0, LI, L2,.. Ln) and shows what is the last lesson given to the student, with 0 showing that the student has not yet started any lesson.
c. RC_i for i ∈ [1,5] has values from (0, C1, C2, … Cn) and shows what are the 5 required concepts for the current lesson. RC_i may be 0 in cases when the lesson has less than 5 required concepts.
d. $HasRc_i$ for i ∈ [1,5] has value 0 or 1. $HasRc_i$ has values 1 if the student knows RC_i and 0 otherwise.

2. Action (Test knowledge taught in current lesson, stay in current lesson, go to lesson Li) where:
   a. Test knowledge means that the system gives to the student a test about the actual lesson. If the student answers correctly this means that he now knows the concepts that are taught by this lesson and student knowledge should be updated in order to be used in the future.
   b. Stay means that the student will be given again the current.
   c. L_i has values from (L1, L2, …. Ln). The system chooses the next lesson to give to the student.
3. Rewards will be:
   a. Positive value if the system gives a lesson that does not require concepts that the student doesn't know
   b. Negative value if the system gives a lesson that requires skills that the student doesn't have
   c. Negative value if the system passes from one lesson to the other without testing the student knowledge first
   d. Positive value if the system testes the student knowledge before passing to the next lesson.
   e. Positive value if the system gives a lesson about concepts that the student does not know.
   f. Negative value if the system gives e lesson about concepts that the student already knows.

By organizing the states, actions and the rewards in this way, the system(agent) will learn what is the best lesson to give to the student, that does not require knowledge that the student does not have. Each test will test specific knowledge, and if the student passes the test, his profile is updated by adding there the newly acquired concepts. In this way, the system now may give a new lesson that requires the concepts that are just learned. If the system gives a lesson that teaches concepts that the student already knows, this will give a negative reward and will force the system to learn that should not give a lesson for something that the student already knows because this will have no effect in his overall knowledge.

4. RESULT AND DISCUSSION

In this work we showed how to build an intelligent tutoring system for teaching concepts of the python programming language. This is treated as a reinforcement learning problem. We focused on the pedagogical module of the ITS that is responsible for deciding the learning materials to be given to each individual student by taking into consideration his knowledge and the concepts that are taught by every lesson. We showed how to organize states, actions and rewards in order to train the pedagogical module using reinforcement learning algorithms, so it can learn what are the best sequences of learning materials to give to each student. Even though we proposed a way for organizing the lesson with the intent of reducing the state space it still can be relatively large, so using tabular methods for solving this reinforcement learning problem may not be efficient. We think that using algorithms that use neural networks for approximating the state space, like the one proposed by (Mnih et al., 2015, p. 532) for playing Atari games, will be a better option. Also, it will be difficult to train the model with real students since the training usually requires many iterations that cannot be done in a live environment. This problem can be handled by simulating different student behaviors until the training is complete. In this way the system can learn even situations that may rarely be seen in real life.

5. CONCLUSSION

An intelligent tutoring system for teaching concepts of programming language can be modelled using reinforcement learning, as showed in this study. Special attention should be given to the way that the states, actions and rewards are organized because they are fundamental in driving the learning in the right direction. Also, minimizing the state space would be desirable since this makes easier the training process.

REFERENCES

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.

Grubisic, A., Stankov, S., & Zitko, B. (2015). Adaptive Courseware: A Literature Review. *J. Univers. Comput. Sci., 21, 1168-1209.*

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*

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

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

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⟩*

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

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

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning, second edition: An Introduction (Adaptive Computation and Machine Learning series)* (second edition). Bradford Books.

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