Personalized Intelligent Tutoring System Using Reinforcement Learning

Ankit Malpani, Dr. B. Ravindran, Dr. Hema A. Murthy {ANKIT,RAVI,HEMA}@CSE.IITM.AC.IN

Keywords: Intelligent Tutoring System, Student Model, Actor-Critic Algorithm, On-Policy learning

1. Introduction

A Personalized Intelligent Tutoring System(ITS) presents individualised inputs to students based on students' needs, adapting to their learning behaviour and uses its knowledge of the topic and experience gained while interacting with students to speed up the students' learning and make the learning process both interesting and challenging.

Such an ITS must have a representation and understanding of a) the topic or subject being taught (Knowledge/Data module), b)the student being taught (Student Model) and c)the methods of instruction to present the inputs to the student in an optimal way (Pedagogic module). The Knowledge module consists of problems with (i)their solutions (or hints) through which a student learns, (ii)their measure of difficulty, (iii)details of the skills required to solve them, (iv)relations or dependencies between different problems and topics etc. The Student Model is based on prior knowledge about the student's learning habits, his external environment and on skills and proficiency required for a topic. The Pedagogic module represents the knowledge a tutor has about the methods of instructions, the optimal way of presenting them to students - the basis on which the ITS suggests questions to a student. It is for learning this Pedagogic module i.e to implicitly train an ITS to teach - that we use Reinforcement Learning.

2. ITS as an RL problem

We look at the ITS as a finite horizon problem i.e. each episode lasts for a fixed finite time. Each question in the data-set is categorized into different types based on its difficulty level along with a weightage (w_i) for each type. At each step the ITS presents a question, records the student's answer and then presents the solution to enable the student to learn. The aim is to present questions such that they maximize the student's learning at the end of the episode based on the weights w_i.

Formulating the ITS as an RL problem allows it to be divided into 4 logical parts as shown in fig 1.

The Teacher is an RL agent that interacts with the student through the environment suggesting a question (the action) based on the current state of the student. At the next time step, it receives an answer from the student and a corresponding reward.
The Environment implements the action suggested by the agent and controls the agent's observations.

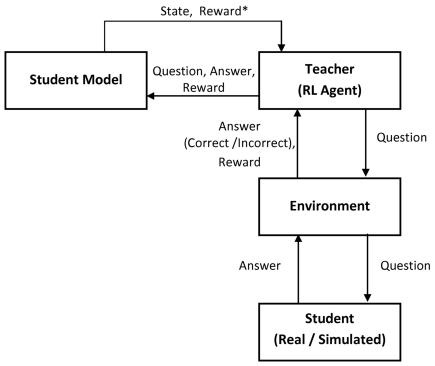

Figure 1. Problem Overview

The Student component represents the student being taught which can either be a real or a simulated student.
The Student Model serves as a state description for the RL agent - providing an estimate of the student state and also a modified reward based on the observations from the environment. The state of the system is estimated with the tuple $< \{p_i | i \epsilon quesType\}, n >$ where p_i is the probability of the student answering a question of i^{th}-type correctly and n is the fraction of the number of questions left to be shown in the episode. Note that the p_is indicate the state of the student. After every k^{th} step, the student model provides a weighted reward $\frac{t}{N} \sum w_i p_i(t)$ to the agent where $p_i(t)$ is the probability of answering question of i^{th}-type correctly at time t in the episode.

It would be difficult for an agent to be able to learn its policies quickly enough from a real student. Hence, the agent is initially trained against a simulated student allowing it to explore different policies and learn; circumventing the risk of having an adverse effect on a real student. It is with this learned policy and the learning mechanism still in place that the agent interacts with the real student to fine-tune its performance to fit each student in a better manner.

Depending on the current state of the student, the simulated student answers the question, views the solutions and updates its state according to equations

$$Pr\Big(P_i(t+1) = c \cdot X_{i,j} \mid Q(t) = j\Big) = P_i(t) \qquad (1)$$

$$Pr\Big(P_i(t+1) = w \cdot X_{i,j} \mid Q(t) = j\Big) = 1 - P_i(t) \qquad (2)$$

where,
$Q(t)$ is an independent random process and $Q(t) \epsilon$ quesType. c, w are constants and

$$X_{i,j} = \alpha_{j,i} \frac{\sum_{k<j} P_k(t)}{\sum_{k<j}} \cdot (1 - P_i(t))$$

$\alpha_{j,i}$ are constants.

Equation 1 is used for questions' answered correctly and equation 2 for questions' answered incorrectly.

The student model also uses the above equations with answers from the student to update its estimate of the student-state. We could use the same parameter values as that of the simulated student to update the student model thus implying that we have complete knowledge of the student which is certainly not true in a real-world scenario.

Thus, the value of each update parameter $\alpha_{j,i}$, c, w needs to be estimated as the agent interacts with the student. As relatively few interactions are available with a real student, estimating each parameter individually is infeasible. Instead, we use a set of student models with different parameter values representing different types of students following the above update equations. The simulated student on the other hand could follow more complicated updates or the same updates but with different parameter values completely unknown to the student model.

A Bayesian estimate of each model being the correct student model is maintained. At each time-step the best model according to the current estimate is selected, moving to this model's estimated current state and suggesting actions based on the model's policy. A certain number of questions are shown such that a student model is selected with reasonable confidence. The model is then fixed and the agent updates its policy based on interactions that follow. An estimate also needs to be made of the start state of the student. At the start, a set of questions of different types need to be presented to the student - recording their answers without disclosing the solutions. The student can now be assumed to be in a fixed start state which can be estimated by the frequency of questions correctly answered. This is then followed by presenting questions (and subsequently solutions) to select a student model as described above.

The student model parameter values once estimated for a student can subsequently be reused when learning different topics but the start state still needs to be re-estimated for each topic. This can also be avoided by using a expert's domain knowledge to construct a Bayesian Belief Network with nodes representing topics and edges the dependencies between the topics.

3. Teacher / RL Agent

We adapt the Actor-Critic algorithm suggested by (Konda et al.) to an average reward formulation using a Tile-coding based function approximator.

We compare our RL Agent (in Red) against a Random

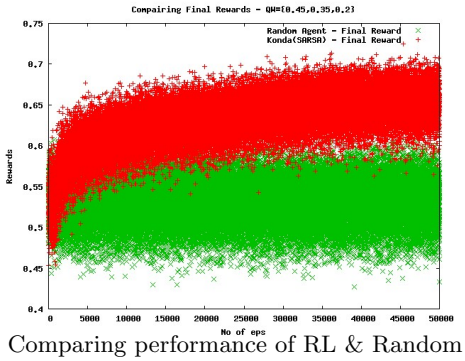

Figure 2. Comparing performance of RL & Random Agent

Agent (in Green) on weighted-rewards ($\sum\ w_i\ p_i^{sim}$) at the end of each episode. Note that p_i^{sim} is the probability of the student (simulated student) answering a question of i^{th}-type correctly and not the Student Model's estimate. The data set has questions of three different types classified on the basis of their difficulty level (into easy, medium, hard) and the student always starts from the fixed state $\{p_e = 0.3,\ p_m = 0.2,\ p_h = 0.1\}$.

Figure 2 compares weighted-rewards at end of episodes of 50 questions, with simulated student parameter values and start state known to the student model. The QuesType weights are $\{w_e = 0.45,\ w_m = 0.35,\ w_h = 0.2\}$.

Figure 3 compares weighted-rewards at end of episodes of 320 questions, with the same fixed student start state as above. The first 20 questions are used by the agent to estimate the start state of the student, the next 75 questions to select the appropriate student model and the last 225 to learn an optimal policy. We simplify our RL problem by using QuesType weights as $\{w_e = 0.6,\ w_m = 0.4,\ w_h = 0.0\}$ and $\alpha_{h,i}$=0.

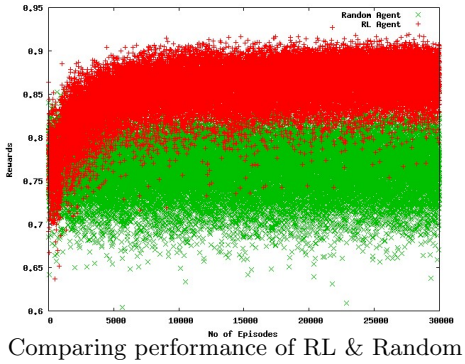

Figure 3. Comparing performance of RL & Random Agent

Question Papers generated with the above agent using problems from a X^{th} standard Mathematics Chapter (OnlineTT) are available at *http://www.cse.iitm.ac.in/ ankit/ITSQuesPaper.pdf*

References

Konda and Tsitsikilis (2000). Actor Critic Algorithms. SIAM Journal on Control and Optimization.

Joseph Beck (1998). Learning to teach with a RL agent. American Association for Artificial Intelligence.

Online Tutoring System, TeNet IIT Madras. http://onlinett.tenet.res.in/