Intelligent Tutoring Systems using Reinforcement Learning to teach Autistic Students

B. H. Sreenivasa Sarma[1] and B. Ravindran[2]
Department of Computer Science and Engineering,
Indian Institute of Technology Madras, Chennai – 36, India.
1. sarmabhs@cse.iitm.ernet.in,
2. http://www.cs.iitm.ernet.in/~ravi/ , ravi@cs.iitm.ernet.in

Abstract. Many Intelligent Tutoring Systems have been developed using different Artificial Intelligence techniques. In this paper we propose to use Reinforcement Learning for building an intelligent tutoring system to teach autistic students, who can't communicate well with others. In reinforcement learning, a policy is updated for taking appropriate action to teach the student. The main advantage of using reinforcement learning is that, it eliminates the need for encoding pedagogical rules. Various issues in using reinforcement learning for intelligent tutoring systems are discussed in this paper.

1 Introduction

A student learns better through one-to-one teaching than through class room teaching. Intelligent Tutoring System (ITS) is one of the best ways of one-to-one teaching. ITS instructs about the topic to a student, who is using it. The student has to learn the topic from an ITS by solving problems. The system gives a problem and compares the solution it has with that of the student and then it evaluates the student based on the differences. The system keeps on updating the student model by interacting with the student. As the system keeps updating the student's knowledge, it considers what the student needs to know, which part of the topic is to be taught next, and how to present the topic. It then selects the problems accordingly.

There are three modules in ITS, namely domain, pedagogical and student modules as shown in Fig.1. The domain module or knowledge base is the set of questions being taught. The pedagogical module contains the methods of instruction and how the knowledge should be presented to the student. The student module contains the knowledge about the student.

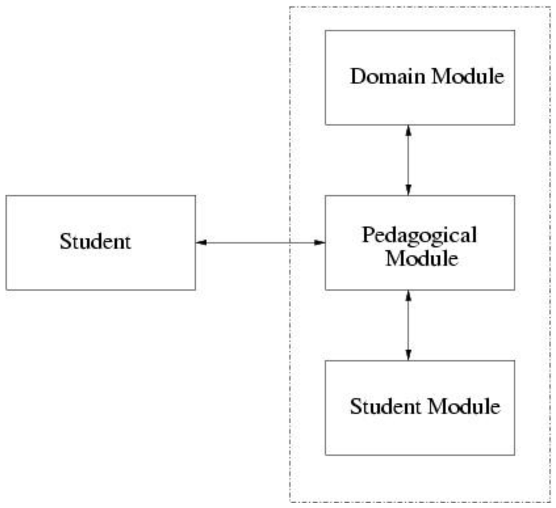


Fig. 1. Block diagram of basic ITS

1.1 Need of an ITS for autistic students

Autism is a semantic pragmatic disorder, characterized by deficits in *socialization, communication and imagination*. Along with the deficits, autistic children may have exceptional learning skills of unknown origin. Many children with autism do make eye contact, especially with familiar people. However, often it is observed that the eye contact is less frequent than would be expected, or it is not used effectively to communicate with others. Our approach mainly focuses on developing an ITS to teach such students.

1.2 Motivation for using Reinforcement Learning

Usually, ITS uses artificial intelligence techniques [5] to customize their instructions according to the student's need. For this purpose the system should have the knowledge of the student (student model) and the set of pedagogical rules. Pedagogical rules are usually in the rule-based form. For example, “if-then” where the “if” part is the student model dependent and the “then” part is the teaching action taken. There are some disadvantages with this method. First, there are many rules which a system can use to teach efficiently, but which are very difficult to encode. Secondly, it is difficult to incorporate the knowledge that human teachers use, but cannot express. The machine tutors have a different set of data available than the human tutors, so the knowledge that could improve the tutor's performance is ignored. Third, rule-based systems are not adaptive to new student's behavior.

The organization of this paper is as follows: Section 2 gives a brief description of reinforcement learning (RL). Section 3 presents the basic idea of using RL for ITS. In Sections 4 and 5, experimental results have been discussed. Some issues in the designing ITS and future work have been discussed in Section 6.

2 Reinforcement learning

RL [9] is learning what to do, how to map situations to actions, so as to maximize a numerical reward signal. The learner is not told which actions to take, as in most forms of machine learning, but instead must discover which actions yield the most reward by trying them. To obtain high reward, an RL 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.

An RL system consists of a *policy*, a *reward function*, a *value function*, and, optionally, a *model* of the environment. A *policy* defines the learning agent's way of behaving at a given time, it is a mapping from perceived states of the environment to the actions to be taken when in those states. A *reward function* defines the goal in an RL problem, it maps each perceived state of the environment to a single number, a *reward*, indicating the intrinsic desirability of that state. The *value* of a state is the total amount of reward an agent can expect to accumulate over the future, starting from that state.

In an ITS, the RL agent acts as the pedagogical module. The RL agent learns a policy for presenting the examples and the hints to the student. There is an ITS called AgentX [8] which uses RL agent as a tutor. In that work, authors used basic RL algorithms like softmax and ε-greedy for evaluating the effects of hints on the student. There are very few proposals for the application of RL in ITS [1, 5]. In [5], authors used the method of clustering students into different levels according to their knowledge, the RL agent then uses this cluster information to take teaching actions. But in our case we have taken the information of individual students, which we expect to provide efficient information to the ITS about a student. In [4], RL is used for modeling a student. They proposed different ways of selecting state variables for an RL agent.

2.1 Mathematical background

This section gives definitions and a brief description of the concepts used in RL. In RL framework, the agent makes its decisions as a function of a signal from the environment's *state, s*. A state signal summarizes past sensations compactly, in such a way that all relevant information is retained. This normally requires more than the immediate sensations, but never more than the complete history of all past sensations. A state signal that succeeds in retaining all relevant information is said to be *Markov*, or to have the *Markov property*.

2.2 Markov Decision Process

An RL task that satisfies the Markov property is called a *Markov decision process* or MDP. If the state and action spaces are finite, then it is called a finite *Markov decision process* (*finite* MDP). Finite MDPs are particularly important to the theory of RL.

It is appropriate to think that a state signal is Markov even when the state signal is non-Markov. A state should predict subsequent states, where the environment model is learned. Markov states are efficient to do these things. If the state is designed as Markov then RL systems perform better than with a non-Markov state. For these reason, it is better to think of the state at each time step as an approximation to a Markov state, though it is not fully Markov. Theory that applies to Markov cases can also be applied to many tasks that are not fully Markov.

2.3 Value Functions

The *value function* is the future reward that can be expected or the expected return. The value functions are defined with respect to particular policies. Let *S* be the set of possible states and *A(s)* be the set of actions taken in state *s*, then the *policy*, π, is a mapping from each state, $s \in S$ and action, $a \in A(s)$, to the probability, $\pi(s,a)$, of taking an action, *a*, when in state, *s*. The *value* of a state, *s*, under a policy, π, defined $V^{\pi}(s)$, is the expected return when starting in *s* and following π thereafter. For MDPs, we can define $V^{\pi}(s)$ formally as

$$\begin{aligned} V^{\pi}(s) &= E_{\pi}\{R_t \mid s_t = s\} \\ &= E_{\pi}\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s\} \end{aligned} \tag{1}$$

Where $E_{\pi}\{\}$ denotes the expected value given that the agent follows policy, π, r_{t+k+1} is the reward for $(t+k+1)^{th}$ time step and γ is the *discount factor*. Note that the value of the terminal state, if any, is always zero. We call the function $V^{\pi}(s)$, the *state-value function* for policy, π. Similarly, the value of taking action *a* in state *s* under a policy, π, denoted $Q_{\pi}(s,a)$, as the expected return starting from *s*, taking the action *a*, and thereafter following policy π:

$$\begin{aligned} Q^{\pi}(s,a) &= E_{\pi}\{R_t \mid s_t = s, a_t = a\} \\ &= E_{\pi}\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a\} \end{aligned} \tag{2}$$

Where $Q_{\pi}(s,a)$ is the *action-value function* for policy, π.

Finding an optimal policy that gives a high reward over long run is the goal of an RL task. For finite MDPs, an optimal policy is a policy, π, that is better than or equal to a policy π', if its expected return is greater than or equal to that of π' for all states. There would be at least one policy, that is better than or equal to other policies, which is called an optimal policy. All the optimal policies are denoted by π^*, and their value functions are denoted by V^* and Q^*.

2.4 Q-learning

Q-learning [9] is a popular RL algorithm that does not need a model of its environment and can be used on-line. Q-learning algorithm works by estimating the values of state-action pairs. Once these values have been learned, the optimal action from any state is the one with the highest Q-value. Q-values are estimated on the basis of experience as in Eq. (3).

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)] \quad (3)$$

This algorithm is guaranteed to converge to the correct Q-values with probability one if the environment is stationary and depends on the current state and the action taken in it.

3 ITS using RL

In this case, RL acts as a pedagogical module which selects appropriate action to teach student by updating Q-values. The RL agent has a function approximator and an RL algorithm, as show in Fig. 2. The state of the student is the summary of a few past training questions asked and the response of the student for those questions. As we are not considering the entire history of questions and answers, this problem is a non-Markov problem. So, to make it more Markov we considered the responses of some of the past questions for the state of the student.

Initially, a random state of the student is considered and a reward for RL agent is obtained. The RL agent takes an action according to the state and reward. The action of the RL agent is to select appropriate question or hint for the student. Each question has its own target. Student is tested with this question as shown in Fig. 2. By following a strategy, a reward for the RL is calculated from the student's response. Then the student is trained with same question as shown in Fig. 3 and state of the student is obtained from the trained output. This process is continued for some number of questions, which is called an *episode*.

A function approximator is used for generalization of states in a large state space. Most states encountered will never have been experienced exactly before. The only way to learn anything at all is to generalize from previously experienced states to one that have never been seen. Many RL algorithms [9] are available which updates Q-values and selects an action accordingly. The knowledge base consists of the questions from a topic, which are represented according to that needed for a student to learn.

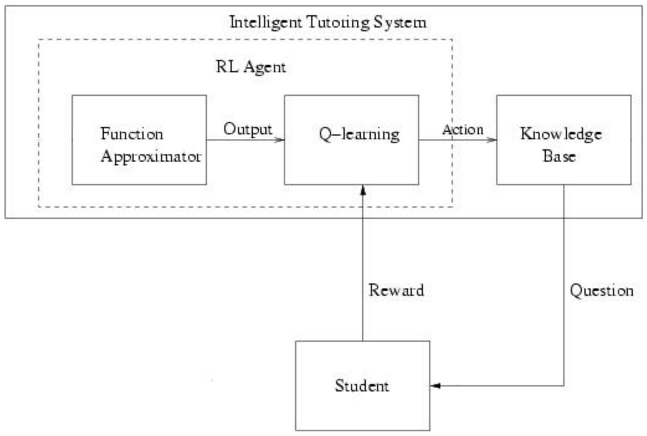


Fig.2. Block diagram of testing phase of the student using ITS with RL.

3.1 Simulated Students

Cohen [6] has shown that Artificial Neural Networks (ANN) with backpropogation can appropriately model the students' selective attention and generalization abilities, who are suffering with autism. The model is based on neuropathological studies which suggest that affected individuals have either too few or too many neuronal connections in various regions of the brain. In simulations, where the model was taught to discriminate children with autism from children with mental retardation, having too few simulated neuronal connections led to relatively inferior discrimination of the input train patterns, consequently, relatively inferior generalization of the discrimination to a novel train patterns. Too many connections produced excellent discrimination but inferior generalization because of overemphasis on details unique to the training set. We have used ANNs to simulate such students in our work.

By using simulated-student, teachers can keep checking the simulated student's knowledge base. If the teachers don't like the recently taken action, they can reset the student's knowledge and try again. Teachers can teach as many times as they can to the simulated student to study the effect of the instructions on the student. Teachers can experiment their tactics with simulated student without fear of failing, which can give negative results with human student.

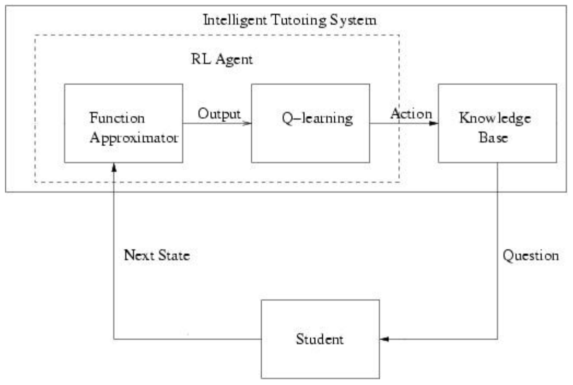


Fig. 3. Block diagram of training phase of the student using ITS with RL.

4 Experiments

We have developed an ITS to teach pattern classification problem. In our case, pattern classification problem is that the student has to classify the pattern (question) given to him. This problem is selected for validating the approach using ANNs, though this is not directly relevant to teaching children. Appropriate question banks should be developed to teach human students.

In pattern classification problem, the knowledge base contains two dimensional patterns from four classes, A, B, C and D, as shown in Fig. 4. The classes are selected in such a way that if a random action is selected, the probability of selecting the pattern from class A is more than from the other classes. The target output for ANN is a four dimensional vector, for example, [1 0 0 0] is the target for class A, [0 1 0 0] is for class B, and so on.

On-line training and testing have been performed on the ANN. The response (output) of ANN is classified into correct (1) and wrong (0) answers. For example, if the target of training question is [0 0 1 0] and if the third output of the ANN is higher than all other outputs then the response is considered as correct, else wrong. The summary of the ANN's response for past 300 questions and the history of responses for past 50 questions are considered for a state of the ANN.

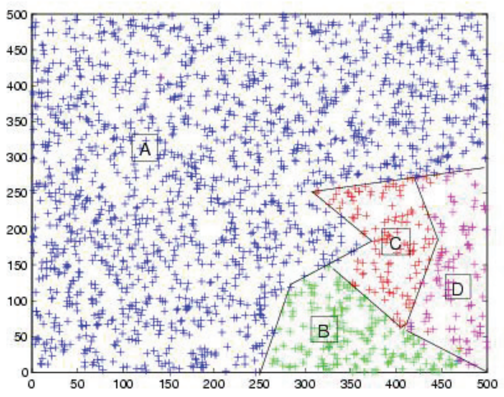


Fig.4. Knowledge base of pattern classes

Among the past 300 questions, let N_A be the number of questions asked from class A. Let N_{AC} and N_{AW} be the number of correct answers and wrong answers for N_A, respectively. Similarly, let $N_B, N_{BC}, N_{BW}, N_C, N_{CC}, N_{CW}, N_D, N_{DC}$ and N_{DW} be the number of questions asked, correct answers and wrong answers from classes B, C and D, respectively. Let x_i be the i^{th} question in an episode. Let z_{i-j}, $1 \le j \le 50$, be the answer for x_{i-j} question. Then the state of the ANN is classes $[N_A N_{AC} N_{AW} N_B N_{BC} N_{BW} N_C N_{CC} N_{CW} N_D N_{DC} N_{DW} x_{i-50} z_{i-50} x_{i-49} z_{i-49} \ldots x_{i-1} z_{i-1}]$ These four form an action set, $A(s)$, for the RL agent. It selects a question from the knowledge base, through policy and ANN is tested with that question. The negative of the Mean Square Error (MSE) of the output of ANN is given as a reward for the RL agent. The same question is used to train the ANN, and the output is used to find the next state of the ANN. This procedure is repeated for 25 episodes, each episode containing 2000 questions. These experiments are done on normal ANN and on ANN model of autistic student.

4.1 RL algorithm

For training of the RL agent, a slightly modified version of Watkin's Q-learning with backpropagation [2] is used. An ANN with single hidden layer is used to learn the $Q(s,a)$ function. The number of input neurons is equal to the dimension of the state, hidden layer contains number of neurons required for feature extraction and number of output neurons equal to the number of actions taken. In this case, we have 72 dimension state, feature size is 80 and 4 actions to be taken.

The activation function for the hidden units is the approximate Gaussian function. Let d_i be the squared distance between the current input vector, s , and the weights in the hidden unit, *j*. Then,

$$d_j = \sum_{i=1}^{72} (s_i - w_{ji}\alpha)^2 \tag{4}$$

Where, s_i is the i^{th} component of s at current time and w_{ji} are the weights of hidden layer. The output, y_i , of hidden unit j is

$$y_j = \begin{cases} (1 - \frac{d_j}{\rho})^2, & if \ d_j < \rho \\ 0, & otherwise \end{cases} \tag{5}$$

where ρ controls the radius of the region in which the unit's output is nonzero and α controls the position of the RBFs in the state space.

Actions are selected ε-greedily, to explore the effect of each action. To update all weights, error back-propagation is applied at each step using the following temporal-difference error

$$e_t = r_{t+1} + \gamma \max_{a_{t+1}}[Q(s_{t+1}, a_{t+1})] - Q(s_t, a_t) \tag{6}$$

Let v_{jl} be the weights of the l^{th} output neuron. Then weights are updated by the following equations, assuming unit *k* is the output unit corresponding to the action taken, and all variables are for the current time *t*.

$$\Delta w_{ji} = \frac{\beta_h}{\rho} e_t y_j v_{j,k} (s_i - w_{j,i}) \tag{7}$$

$$\Delta v_{j,k} = \beta e_t y_i \tag{8}$$

$Q(s_{t+1}, a')$, $\forall a' \in A(s)$, is the product of updated $v_{j,k}$ and the output of function approximator, y_i .

5 Results

The results in Fig. 5 and 6 are obtained for ε=0.2, which means the exploration is done for 20% of the questions. The other parameters are, $\beta_h = 0.09$ β=1.0 and α=200.

Fig. 5(a) shows the average percentage classification for first 500 questions selected for the normal ANN with 5 hidden layer neurons, without ITS and with ITS. Classification of ANN without ITS is around 26%, which is much less than that compared to the classification of ANN with ITS, which is around 70%. Fig. 5(b) shows the histogram of actions taken by ITS. The uniform distribution of the actions shows that the ITS is not stuck in the local optima. ITS selects an action depending on the present state of the student, to increase the future classification rate of the ANN.

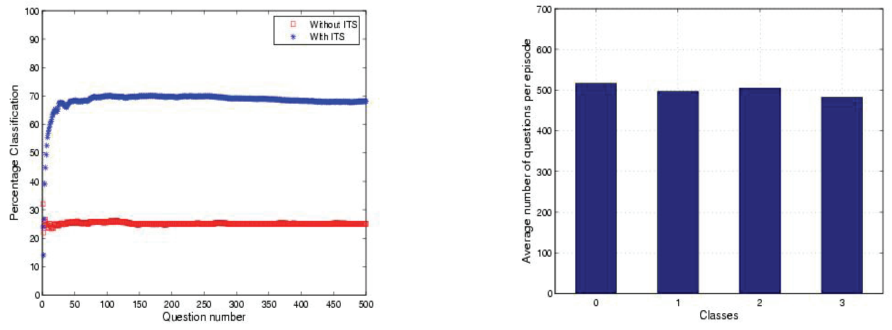


Fig. 5. (a) Percentage classification (left figure) by ANN model of normal student (learning rate 0.2). (b) Histogram (right figure) of actions taken by RL agent, averaged over episodes. (0, 1, 2 and 3 represent classes A, B, C, and D respectively).

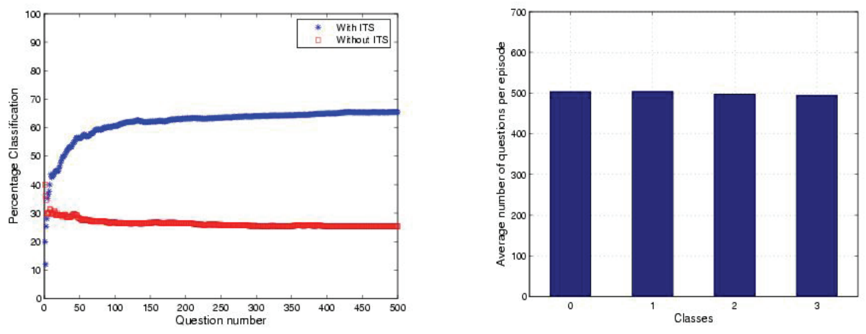


Fig. 6. (a) Percentage classification (left figure) by ANN model of an autistic child (learning rate 0.2). (b) Histogram (right figure) of actions taken by RL agent (0, 1, 2 and 3 represent classes A, B, C, and D respectively)

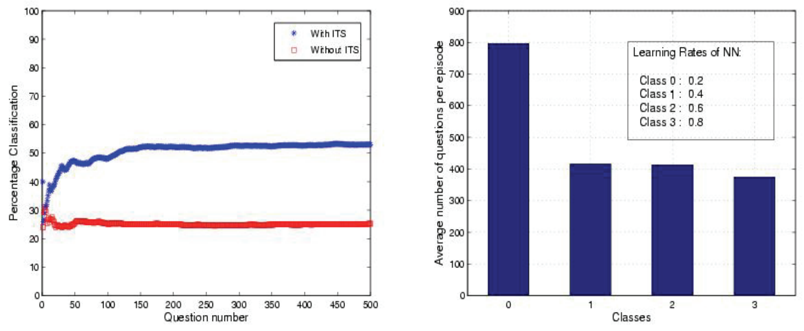


Fig. 7. (a) Percentage classification by ANN model of autistic student with learning rates 0.2, 0.4, 0.6 and 0.8 for classes 0, 1, 2, and 3, respectively. (b) Histogram of actions taken by RL agent, averaged over episodes (0, 1, 2 and 3 represent classes A,B,C, and D respectively)

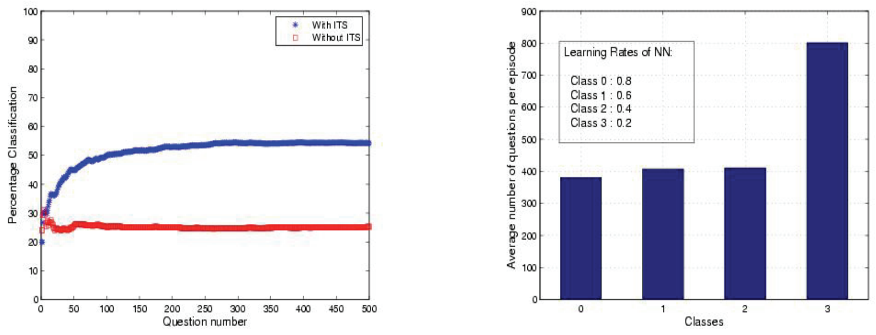


Fig. 8. (a) Percentage classification by ANN model of normal student with learning rates 0.8, 0.6, 0.4 and 0.2 for classes 0,1,2 and 3 respectively (b) Histogram of actions taken by RL agent, averaged over episodes (0,1,2 and 3 represent classes A,B,C, and D respectively)

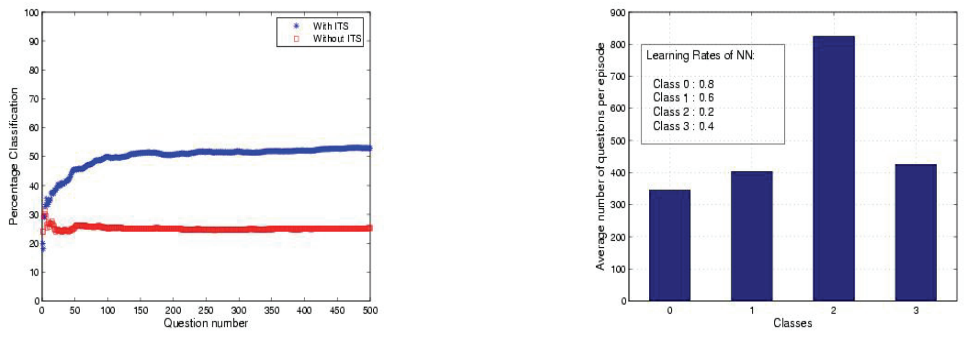
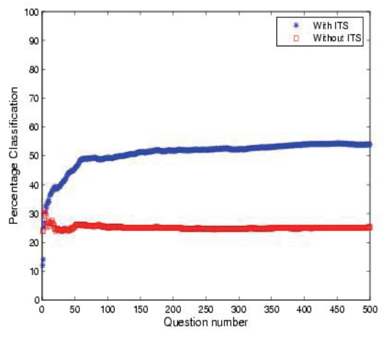


Fig. 9 (a) Percentage classification by ANN model of normal student with learning rates 0.8, 0.6, 0.2 and 0.4 for classes 0, 1, 2, and 3 respectively (b) Histogram of

actions taken by RL agent, averaged over episodes (0,1,2 and 3 represent classes A,B,C, and D respectively)



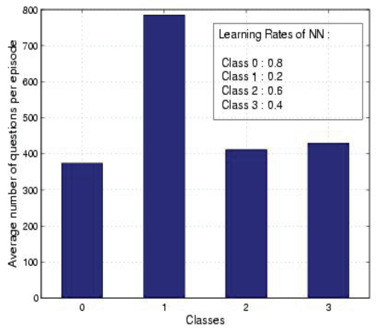


Fig. 20 (a) Percentage classification by ANN model of normal student with learning rates 0.8, 0.2, 0.6 and 0.4 for classes 0, 1, 2, and 3 respectively (b) Histogram of actions taken by RL agent, averaged over episodes (0,1,2 and 3 represent classes A,B,C, and D respectively)

The goal was to develop an ITS capable of adapting to large deviations from normal learning behavior. So, we have simulated models of both autistic student and normal student by selecting more neurons (15 neurons) in the hidden layer than that required (5 neurons, normal behavior) for capturing the information in the input patterns. We have evaluated the ITS using these simulated models. Fig. 6(a) shows the classification rate for ANN model for autistic student, with ITS and without ITS. Classification rate of autistic model can be compared with that of the normal ANN. In both cases, the percentage classification is approaching the same value (70%), indicating the autistic student can be taught effectively using ITS. But autistic student needs more number of questions (around 175 questions) to learn, than that required for a normal student (around 50 questions). Fig. 6(b) gives the histogram of actions taken by RL agent for the ANN model of autistic student.

The policy learned by the ITS seemed to be picking actions at random, uniformly from all the 4 classes. Though this is the desired behavior, we cannot be entirely sure that this was learned. So we tried different experiments which have different desired behaviors. For example, experiments where the student had different learning rates like 0.2, 0.4, 0.6 and 0.8 for classes A, B, C, and D respectively. The ITS learned the appropriate mix of actions to take, as shown in Fig. 7(b). Fig. 7(a) shows the classification performance of such combination. For other combinations of learning rate, similar learned behavior was observed as shown in Fig. 8, 9 and 10.

6 Conclusions and Future Work

Sections 4 and 5 presented the experiments performed to test the application of RL for ITS to teach an autistic student. We conclude that, by considering the history and summary of past few questions as state variables, an autistic student can be taught as effectively as a normal student.

We are now concentrating on improving the present ITS using the hierarchical framework [3]. In a hierarchical framework, entire knowledge base is divided into lessons and each lesson is divided into different categories. The RL agent has to learn two policies, one for picking a lesson and the other for picking a categories within the lesson, which is expected to improve the performance of the ITS.

This can be extended to real world problems like teaching mathematics, where selection of state variables and action variables is much more difficult task. In this paper, we used the history of past 50 questions and summary of past 300 questions as state variables. But in real world situation, we can consider the variables like the amount of time taken by the student to answer a question, history of hints the student requested. More work can be done in selecting state variables, which can improve, not only the percentage classification but also the learning rate. In this case, we have to consider which type of questions form a group, for example, easy questions form a group and tough questions form another group.

Other applications of our work include pattern synthesis and active learning. Pattern synthesis is the process of generating patterns for training and testing a machine. In our case, the two dimensional data generated as a question can be considered as the pattern synthesis problem. Active learning, is the "learning with examples" [7]. This is a closed-loop phenomenon of a learner asking questions that can influence the data added to its training examples. In the case of ITS, the student has the facility to ask for hints for improving his knowledge on the topic, this can be considered as active learning. Active learning provides greatest reward in situations where data are expensive or difficult to obtain.

References

1. B. Abdellah, D. Theo and M. Bernard. An approach of reinforcement learning use in tutoring systems. In *Proceedings of International Conference on Machine learning and Applications, ICMLA'02*, 2002.
2. C. W. Anderson. Q-learning with hidden-unit restarting. In *Proceedings of Fifth International Conference on Neural Information Processing Systems*, pages 81-88, 1992.
3. A. Barto and S. Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamics Systems: Theory and Applications*, 13: 343-379, 2003.

4. J. E. Beck. Modeling the student with reinforcement learning. *Machine learning for User Modeling Workshop at the Sixth International Conference on User Modeling*, 1997.
5. J. E. Beck. Learning to teach with a reinforcement learning agent. *American Association for Artificial Intelligence (AAAI)*, 1998.
6. I. L. Cohen. An artificial neural network analogue of learning in autism. *Biological Psychiatry*, **36**(1):5-20, 1994.
7. D. A. Cohn, Z. Ghahramani, and M. I. Jordan. Active learning with statistical models. *Journal of Artificial Intelligence Research*, **4**: 129-145, 1996.
8. K. N. Martin and I. Arroya. AgentX: Using reinforcement learning to improve the effectiveness of intelligent tutoring systems.
9. R. S. Sutton and A. G. Barto. *Reinforcement learning: An Introduction*. MIT Press, Cambridge, MA, 1998.