Sample Efficient Learning to Make Decisions with a Focus on Education

Min Hyung Lee
Machine Learning Department
Carnegie Mellon University
minhyunl@andrew.cmu.edu
Supervised by Emma Brunskill

Abstract

Background Massive open online classes have made education accessible to a huge number of students, but each student in a class receives the same set of materials. However, previous research shows that dynamically adapting content can lead to improvements in the efficiency and efficacy of the courses. Intelligent tutoring systems have been successful at creating adaptive courses, but they require a lot of human expertise in tuning the algorithms and/or large amount of data to train the algorithms.

Aim Framing the problem as a sequential decision making process, we define the number of exploration that the algorithm requires as sample efficiency. In such setting, our work wishes to find a sample efficient decision making algorithm that uses a small number of data points to find a good instructional policy.

Experiment Platform We created an online course on histograms based on an existing statistics curriculum capable of adaptive problem selection.

Methods In the first attempt, we tried to use the Abstraction from Demonstration approach to analyze human demonstrations to extract features that are most crucial in making instructional decisions. However, a pilot study showed that the approach still required a huge amount of exploration. Instead, we moved on to another approach in which we use a student model called Bayesian knowledge tracing to parameterize the policy space. To find the best policy, we used Bayesian optmization to directly search for the policy that yields the most reward. To test the approach, we recruited 60 participants through Amazon Mechanical Turk, and tested different policies to teach the participants.

Results The policy chosen by Bayesian optimization after only 30 data points yielded an objective value similar or higher than the baseline policy, showing potential in finding a good instructional policy. However, the behavior of the policy was not optimal, as it was giving no or very few problems to the subjects.

Conclusions As an educational platform, testing different policies on students to find a good one is very expensive. Thus, ensuring sample efficiency in learning instructional policies is necessary. Through our work, we tried different approaches to tackling this problem, and in the end, found some promising result in using Bayesian optimization to find a good instructional policy.

1 Introduction

Massive open online classes (MOOCs) are revolutionizing accessibility of education with hundreds of thousands of students worldwide taking courses concurrently through Coursera, EdX, etc. However, all the students receive the same set of materials including problem sets and lecture videos regardless of their previous knowledge of the material or their speed of learning [16]. Yet it is well known that dynamically adapting content to a student can lead to substantial improvements. Bloom [2] first showed that small teacher-to-students ratio can lead to substantial learning benefits. More recently, research on intelligent tutoring systems (ITS) have also shown that by using ITS, the time spent learning the same amount of material could be reduced to a third of the typical length [7].

However, ITSs typically require a huge amount of human expertise both to create the course content and also define and refine a statistical model of student learning used to select problems. To select good parameters for statistical student models, which yield a system that enables good learning outcomes, typically requires significant human expertise and/or large amounts of data to fit these parameters. This crucially limits the benefits of ITSs, since in MOOC platforms, new courses are constantly created, and the administrators often lack the resources to hire educational experts or have large amounts of data in advance. Each data point needed to train the algorithm in the MOOC setting is a student trying to learn new material through the course. Each data point is precious and should be treated with care.

The goal of our project is to develop a learning system that can provide personalized instructions to students in the MOOC setting as quickly as possible. We define sample efficiency as the number of student interactions (number of data points) required to obtain a system that makes good decisions.

Section 2 introduces the framework with which we formulate the problem. Section 3 briefly details approaches taken in the RL setting to improve sample-efficiency. Section 4 describes the course platform we created to gather data and run experiments regarding different approaches. Section 5 describes the first approach we took, in which we use human demonstrations to improve the sample efficiency of RL algorithms based on abstraction from demonstration. We executed a pilot study to try out the approach, but the results were not promising. We moved on to the second approach described in Section 6. Section 7 wraps up the work discussed throughout the paper.

2 Background

We formulate the adaptive problem selection process in online courses as sequential decision making under uncertainty. A sequential decision making problem is commonly modeled as a Markov decision process (MDP) which consists of a tuple $< S, A, R, T, \gamma >$. S is the set of possible states, where a state is the belief and observation of where the system is currently at. A is the set of possible actions or the decision a system can make. $R(s, a)$ is a reward function, which governs how much reward or feedback the system will receive for performing action a at state s. $T(s, a, s')$ is the transition function, which is the probability of landing at a certain state s' after performing action a at state s. γ is the discount factor. It is a measure of how the effect of choosing an action diminishes over time.

Based on the underlying MDP, the system chooses what action to perform at each state without knowing the reward or transition functions in advance. The collective decision that the system makes is formulated as a policy. A policy π is defined as a mapping from state to action, i.e. a guideline of what action the system will take at each state. A policy is then evaluated using the following two functions. The value of a policy starting at state s called the value function, $V^\pi(s)$, is the expected sum of future rewards by following the policy. The value of choosing an action a at state s and then following the policy π is defined as the Q function, $Q^\pi(s,a)$. The two functions satisfy the following recurrence relations.

$$Q^\pi(s,a) = R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V^\pi(s')$$

$$V^\pi(s) = Q^\pi(s, \pi(s))$$

If we know the reward and transition functions, we can use dynamic programming to find the value of the policy. However, in practice, we do not know these functions in advance. Reinforcement learning (RL) algorithms finds the policy with high expected sum of future rewards through exploration. Typical RL algorithms test different policies, choosing different actions at each state, to estimate the value function and Q function. The policy can then be generated by selecting the action that leads to the highest value [25].

For instance, in our setting, the state may be how many questions the student got right so far, how long did the student take to get this far, or a statistical measure of how much the student mastered the subject. Note that a state can be represented in many ways. The action is which learning material to give next to the student. The reward may be the score the student receives on a test taken at the end of the course, an objective evaluation of how well the student learned. The goal of the RL algorithm would be to find the best instructional policy that yields the highest average test score. It would have to do exploration of different instructional policies to achieve this goal. In the educational setting, however, each exploration requires a new student testing the policy, which is very expensive. Thus, we aim at finding a sample-efficient RL algorithm, which can find a sufficiently close-to-optimal policy using the least number of exploration.

3 Related Works

Sequential decision making algorithms have been used in education in the past. Chi et al. [4] used model-based RL algorithm to find instructional strategies, and evaluate their empirical performance compared to hand-tuned policies. Lopes et al. [15] used multi-armed bandits for an intelligent tutoring system, in which the RL problem is simplified by having only one state variable. Mandel et al. [17] applied policy evaluation techniques on optimizing educational games. Rafferty et al. [19] used partially observable MDP planning to accelerate teaching. Although the works proved the effectiveness of using RL in the field of education, they did not focus on sample-efficiency, which we focus on in our research.

A lot of research has been done on creating sample-efficient RL algorithms. The first is Probably Approximately Correct (PAC) RL algorithms. An RL algorithm achieves a PAC guarantee when the number of data points necessary to find a policy that is ϵ-close to the optimal policy is less than some polynomial in the relevant quantities including the size of the state and action space [22]. Thus, this approach aims at theoretically formulating sample-efficiency. Some examples of PAC RL algorithms are R_{max} [3], MBIE [23], and delayed Q learning [22]. However, the theoretical bounds that these approaches give are not tight enough for real-world settings. R_{max} and MBIE algorithms require more than 10^6 exploration per distinct state-action pair, which is too large in the practical setting. Thus, we moved on to looking at other directions to solving the problem.

Another approach is to use function approximation techniques to model the reward function. In the work by Ure et al. [24], they used an adaptive function approximator of the value function. Even though the state space was extremely big, they were able to approximate the value function using a small number of data points, and thus find the optimal policy. Some popular RL algorithms using function approximators are least-squares policy iteration [12], which uses a class of linear functions, and fitted Q iteration. In the work by Riedmiller [20], they use neural networks to approximate the function leading to great gains in sample efficiency. However, since this approach approximates the value function using a specific class of functions, many assumptions about the behavior of the value function must be applied for good results, and so we elected not to consider this approach.

In our work, we use two other approaches: leveraging human demonstrations and directly searching for the best policy in the policy space. Details of the two approaches will be discussed in the appropriate sections.

4 Experiment Platform

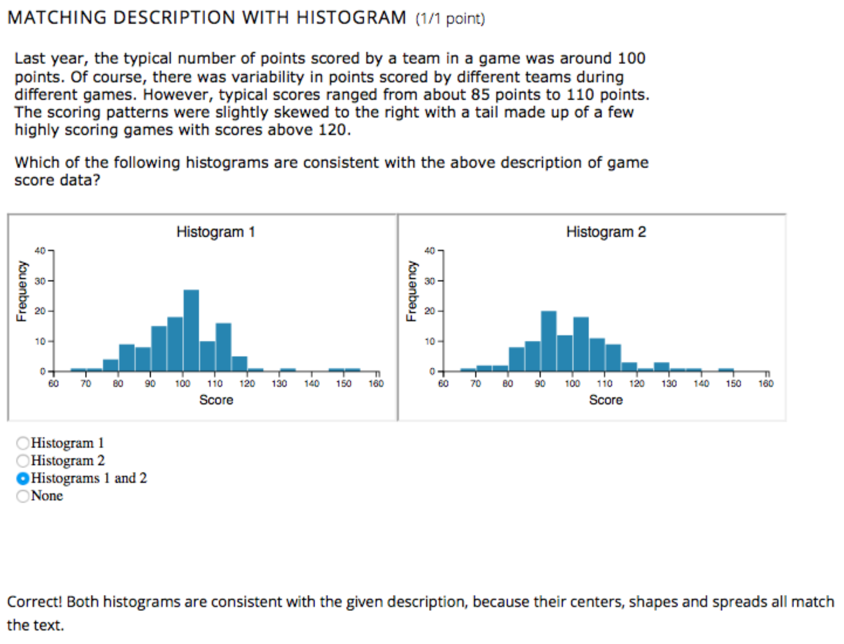
To experiment with our algorithms of personalizing educational platforms, we created a course platform based on the EdX open source platform[1]. The main design goals of the course were to cover a topic with a wide demand and need, to have a reasonable scope of material that we can teach in a couple hours, and to allow adaptive problem selection.

The topic for our course was histograms. As an intuitive tool for analyzing data, histograms are used in a wide variety of places. However, according to [9], many students struggle to try to understand and interpret histograms, including knowing what the axis means, what properties can be derived from the shape of the histogram, etc. Along with the wide demand and need for histograms, we hypothesized that it could be taught in a few hours, allowing the design of short-term experiments.

Referencing the Statway curriculum[2] developed by the Carnegie Foundation for the Advancement of Teaching, we initially created 20 problems with multiple choice, short answer, and matching questions. Since the problems are designed to teach, the students can try multiple times until they get the question correct. Some of the problems give hints to help the student make the next guess. When the student gets the problem correct, we give a brief explanation of the solution. Students are not allowed to move on to the next problem until they get the problem correct. Figure 1 is an example of a descriptive multiple choice problem. It starts with a brief description of where the data is collected from and a description of where the center is, what the spread and shape of the distribution is like. Then the student must choose which of the histograms correspond to the description. Also, note the short explanation of why choice 3 was the correct answer.

Along with the problems designed to teach the students, we created an assessment to test the current state of knowledge. The assessment contains 13 questions based on previous research on testing the student's knowledge on histograms [11]. The same assessment is given at the start of the course, and at the end of the course, evaluating the improvement of students after completing the course. Figure 2 is an example assessment question testing the understanding of what center

[1]https://github.com/edx/
[2]http://www.carnegiefoundation.org/in-action/pathways-improvement-communities/



MATCHING DESCRIPTION WITH HISTOGRAM (1/1 point)

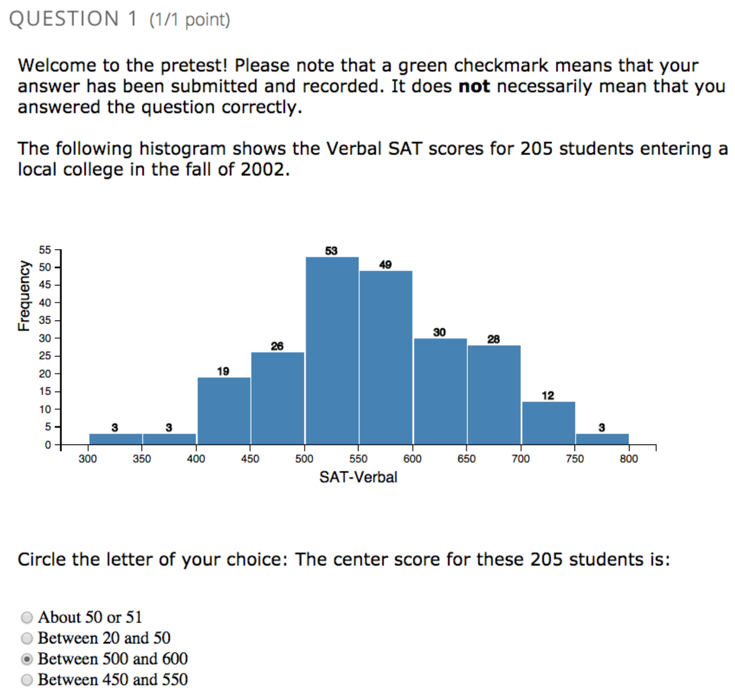
Last year, the typical number of points scored by a team in a game was around 100 points. Of course, there was variability in points scored by different teams during different games. However, typical scores ranged from about 85 points to 110 points. The scoring patterns were slightly skewed to the right with a tail made up of a few highly scoring games with scores above 120.

Which of the following histograms are consistent with the above description of game score data?

- Histogram 1
- Histogram 2
- [✓ Histograms 1 and 2]
- None

Correct! Both histograms are consistent with the given description, because their centers, shapes and spreads all match the text.

Figure 1: Example descriptive problem in the histogram course



QUESTION 1 (1/1 point)

Welcome to the pretest! Please note that a green checkmark means that your answer has been submitted and recorded. It does **not** necessarily mean that you answered the question correctly.

The following histogram shows the Verbal SAT scores for 205 students entering a local college in the fall of 2002.

Circle the letter of your choice: The center score for these 205 students is:

- About 50 or 51
- Between 20 and 50
- [✓ Between 500 and 600]
- Between 450 and 550

Figure 2: Example assessment problem in the histogram course

of a histogram means. Based on the histogram on SAT verbal scores, the student must pick the correct center of the distribution. In contrast to the non-assessment problems, student gets one chance of answering the problem and we give no feedback on whether the attempt was correct or incorrect to reduce the effect of assessment of the knowledge of the student.

We initially used the tools provided by the EdX platform to create the problems, but at the current stage, EdX does not support adaptive reordering of problems, which was crucial for our study. To accommodate, we implemented an external problem server using Python and Javascript, which renders a "next problem" button as soon as the student finishes the problem. By clicking the button, the user moves to the next problem chosen dynamically, while being prohibited to solving any other problems. Such measures allowed us to integrate adaptivity to the EdX platform.

5 Pilot Study on Using Human Demonstration

5.1 Approach

The first approach we took was to leverage expert demonstrations, as in the case of learning from demonstration (LfD). In contrast to typical RL algorithms, which try to estimate the consequences of each action through exploration, LfD uses expert demonstration to learn the optimal state to action mapping using a supervised learning approach [1]. Experts demonstrate the task to be learned while the system tracks what decision the expert made at each state. Such sequences of state-action pairs is compiled into a labeled dataset. The state is represented as a list of features, and the action is the class label. By training machine learning algorithms on the dataset, the system can predict which action to take at each state, thereby learning from demonstration. However, such an approach puts a high reliance on the experts and their actions. The system struggles in making the right decision, when it lands on a state that it never observed in the demonstrations or when the demonstrator made a sub-optimal decision by mistake. Numerous research has been done on generalizing LfD to overcome this problem.

One of the algorithms that try to address this problem is abstraction from demonstration (AfD) developed by Cobo et al. [6]. AfD is suitable for featured stated spaces. AfD builds a classifier to predict what features of the state are used by the demonstrator to make decisions. It then uses only those features to create a new state space, which is then used as the state space for running a RL algorithm on in the future. Intuitively, many features may be irrelevant for near optimal performance, and the idea is that people may not be able to perfectly do a task, but they may still be good at identifying which features are relevant for selecting actions or making decisions.

As with LfD, AfD first compiles the demonstrations by humans into a labeled dataset. Starting from the full set of features, it greedily removes one feature at a time with the least effect on the predictive accuracy of a decision tree classifier. Stopping once the predictive accuracy reaches some threshold, the remaining set of features is input into a RL algorithm. By doing so, the reinforcement learning can be trained significantly faster, due to the reduction of the state space and the policy space [6]. For instance, assume there were 100 features initially, each discretized into 10 classes. Using only 10 of these features, the number of distinct states decreases from 100^{10} to 10^{10}. Assuming there are 8 possible actions, the number of distinct policies, which is a mapping from state to action, decreases from 100^{80} to 10^{80}. Thus, by reducing the number of features, the size of the policy space is greatly reduced, and the number of samples required to explore the policy space is significantly reduced. Pseudo-code for the feature selection process of AfD is in

Algorithm 1.

We believed AfD might be a promising direction for improving the sample efficiency of using RL to learn how to progress through our histogram course. The main motivation was on finding a good representation for how the student is progressing through the course. We could make numerous observations while the student solves each problem like the time spent on solving the problem, number of times the student got similar problems wrong, number of attempts the student made on similar problems, etc. However, if we use all the features to form a state, we would have a large number of different states that the student can be in. We observed above that even with 100 features the size of the state space and the policy space explodes, making sample efficient learning intractable. Thus, we thought by looking at human demonstrations we could pick a smaller set of features that are crucial in making the instructional decisions. As a first step towards this, we wanted to learn how many features are used by teachers instructing students in our histogram course.

A few demonstrations by teachers carry an abundance of information that would take a system huge number of explorations to collect. Teachers can quickly decide whether the participant needs more practice from observing the student's learning progress. If we can find out which features the teachers looked at the most, we will be able to learn a reasonable policy using a fewer number of explorations. Naturally, we could ask the teachers directly for the features they think were the most important. Unfortunately, research from cognitive task analysis [5] suggests that experts frequently fail to verbally specify all the information they are using to make task decisions. A data-driven feature extraction methods like the one used by AfD algorithm could find the features the experts look at consciously and unconsciously.

We did a preliminary pilot study to test the feasibility of this approach. We collected human demonstrations of the task and using the AfD approach, analyzed which features affected the decisions the most.

5.2 Method

5.2.1 Design

To study the effect of using human demonstrations on the sample efficiency of the histogram tutor, we needed to first gather human demonstrations. In our case, it would be teachers instructing the participants to solve a certain sequence of problems according to the participants' progress through the course. There are several ways to make such instructions possible. A teacher could sit behind the student while they solve the problems in the course, observe how they perform, and tell the students which problem to solve next. Although easy to execute, such process will give teachers access to features that our system cannot trivially collect including gestures or facial expressions. As a web-based course, our histogram course cannot film the participants in any ways, and only has access to participants' attempts at solving the problem and the time stamp at which the attempt was made. To control the features that the teachers are exposed to, we used the screen sharing function of Google Hangouts, which shows the student's screen in real-time, but not their faces. There were still features like the mouse and scroll movement that the teacher could observe through screen sharing while the system cannot. For this pilot study, we decided to still take this approach since the teachers would have to only look at logs of student performance while making decisions.

As a reminder about the course platform used in this pilot. Each student's interaction with the course began with a pre-assessment consisted of 13 problems. Afterwards, they were given

Algorithm 1: Greedy Feature Selection Algorithm Used in AfD

```
Data: Human demonstrations $H = \{(s_{1,1}, a_{1,1}), (s_{2,1}, a_{2,1}), \ldots, \ldots\}$ where $s = [f_1, \ldots, f_n]$
      and $a$ is the decision human made, full set of features $F$
Result: Subset of features $F' \subset F$
$D$ = Flatten $H$;
Run decision tree classifier on D with features $F$;
$a_0$ =predictive accuracy of the classifier;
$t := 0, F' = F$;
while $a_t > 0.95 \times \max_i a_i$ do
    $a_{t+1} := \infty$;
    $rm_{t+1} := 1$;
    for $i = 1$ to $|F'|$ do
        $F'' := F' \backslash f_i$;
        Run decision tree classifier on D with features $F''$;
        $a$ :=predictive accuracy of the classifier;
        if $a < a_{t+1}$ then
            $a_{t+1} := a$;
            $rm_{t+1} := i$;
        end
    end
    $F' := F' \backslash f_{rm_{t+1}}$;
end
return $F'$
```

a variable sequence of descriptive problems designed to teach the participant about histograms. Once the teacher decided that the student fully understood the material, the students finished the course with a post-assessment, which are the same set of problems as the pre-assessment. In this study, the problems were classified into 6 different categories hand defined according to what we believed they were aimed at teaching. These skills were 1)interpreting axis, 2)constructing dot plots, 3)center, spread, and skew of histograms, 4)building histograms from data, 5)matching histogram to description and vice versa, and 6)miscellaneous problems.

5.2.2 Participants

We recruited 16 participants for the pilot. We sent recruitment emails to undergraduate students and staff workers from Carnegie Mellon University and University of Pittsburgh who did not have access to formal statistics education in college before. Each participant was given a fixed payment of \$15 at the end of each experiment. The study was approved through CMU's IRB.

Recruiting teachers for demonstration was more challenging. They had to thoroughly understand the topic and remember all the problems that can be given to the students. We decided that the people in our research group understood the course the best, since we were the creator of the content. All the five people in our group understood histograms very well, and reviewed the list of problems multiple times before participating in the experiment to get familiar with the content. To remove the bias of each human demonstrator, each member participated in two to four experiments.

5.2.3 Instructions to Participants

The instructions given to the participants were as follows. First, they were to join a Google Hangouts video call with one of our group members. The time and location of the experiment were agreed upon beforehand. Some came to our lab to use one of our computers for the study, while others joined remotely on their own computers. Regardless of the location of study, they were separated from the teacher leaving only screen sharing and text messaging as means of communication.

After screen sharing was set up, we sent the participants new accounts to join the course platform with, and they began working on the pre-assessment. After the end of the pre-assessment and all other problems, they were told to wait for a message from the teacher. The teacher, looking closely at and taking notes about how the participant was performing, chose the next problem to give the student, and sent the name of the problem to the student through a text message. The student would then navigate to the given problem through the menu, and start working on it. Such process continued until the teacher told the participant to go to the post-assessment page, marking the end of the course.

5.2.4 Procedure

The overall study went as follows. First, we gathered human demonstration data as described in previous sections. In the process, we obtained feedback about our platform and continued to update the course. Six out of sixteen participants worked on the final version of our course, so we only used data generated from these six participants. The data contained the correctness and time stamp of each attempt that the student made on each problem. From the data log, we generated 82 distinct features listed in Table 1 at the end of each problem. (The number of features corresponding to each entry is noted in parentheses) These are the candidate features that the teacher may have looked at while making the decision on which problem to give next. The decision made was added as the class label. Due to the lack of data, we used the category of the problem chosen to be given next as the class label. Thus, the sequences of state and action pairs generated by human teachers were converted into a multi-class classification dataset with 6 distinct classes.

Using the preprocessed dataset, we applied the decision tree classifier implemented in the WEKA package developed by the University of Waikato [10]. At each iteration, we removed one feature at a time and ran the classifier with 6-fold cross-validation to compute the predictive accuracy of the classifier. We used a leave one student out cross validation, where the data points corresponding to each student is left out to be used as a test dataset. After obtaining the classification accuracy when each feature was removed, we chose the one with the highest accuracy, which meant that the feature carried no significance in making the decision. The process continued until the predictive accuracy dropped under a certain threshold: 95% of the maximum accuracy so far. Through the greedy process, we were able to reduce the number of feature significantly. Although AfD then inputs the selected features into a RL algorithm, we wanted to look at the set of features the process would pick before moving on.

Correctness on current/last problem (2)
Number of attempts in solving current/last problem (2)
Time spent on the current/last problem (2)
Total time spent until the current/last problem was solved (2)
Number of problems the student got correct until the current/last problem (2)
Number of problems the student got wrong until the current/last problem (2)
Pre-assessment score (1)
Time spent on pre-assessment (1)
Number of problems finished in skill 1∼6 after solving the current/last problem (12)
Number of problems finished in the current skill (2)
Number of problems student got correct in skill 1∼6 after solving the current/last problem (12)
Number of problems student got correct in the skill of current/last problem (2)
Number of attempts on the last problem in the skill of current/last problem (2)
Time since last time the student got the problem in the skill of current/last problem (2)
Time spent on the last problem in the skill of current/last problem (2)
Pre-assessment score for each skill (6)
Pre-assessment score for current skill (1)
Whether the student finished each seq (1 feature for each seq) (26)
Current skill (1)

Table 1: List of features used in the experiment

5.3 Result

We divided the cases into two. The first was when we included the feature about the skill corresponding to the current problem, and the second was when we did not include the feature. Since the teachers tended to give multiple problems of the same skill in a row, the current skill was a highly informative feature. In other words, by choosing the same skill as the current one, the classifier was able to achieve accuracy of over 50%. We wanted to see which features are picked when this informative feature was not included.

When using the full set of features with the current skill feature, the predictive accuracy was 56.13%. In the process of removing features, it achieved maximum accuracy of 66.98% with 8 features and halted with an accuracy of 64.48% using 4 features. They were the amount of time spent until now, time passed since solving the problem with the same skill, how many problems the student received in the same skill, and the current skill.

When the current skill feature was removed, the predictive accuracy was 47.79%. In the process of removing features, it achieved maximum accuracy of 67.26% with 7 features and halted with an accuracy of 66.37% using 6 features. They were how many questions the student got wrong on first try, pre-assessment score for the skill interpreting axis, number of problems done on skills 1)constructing dot plots, 2)center, spread, and skew of histograms, 3)building histograms, and 4)matching histogram to description and vice versa.

5.4 Discussion

In general, the accuracy of the predictions increased as we removed more features. We believe this is because using fewer features removed over-fitting of the data, which was prevalent since the size of the dataset was very small. But the fact that a comparable accuracy can be achieved with only a small subset of features is encouraging. Also, we note that when the current skill feature is not included, the final accuracy is higher, which seems unlikely since all features included in this case were also included in the first case. We think it is due to the nature of a greedy algorithm. Since current skill is correlated with many other features in the list while carrying much information, in the first case, it picks this feature instead of any other related ones. However, in the second case, it is able to find related features which actually have higher predictive significance.

The experiment also revealed multiple challenges about the problem. First was the issue of low predictive accuracy. Although the predictive accuracy of 67% is not high enough to ensure that the features chosen are enough to make reliable decisions. One possible cause may be that we did not extract all features that were accessed by the teachers. We could not gather features on the cursor movements or the scrolling actions that may give information about the student's progress in the problem. Another cause may come from the greedy approach that we took. The set of features that we chose may not be globally optimal. Such lack of accuracy question the validity of the features chosen through the approach, leading to invalid restrictions on the policy space.

Fundamentally, although we managed to reduce the number of features down to 7, the feature space or the state space of the problem is still too large. 7 features each with 10 discrete values leads to 10^7 distinct states (combination of features). Since we have 6 different actions to choose from, the number of distinct policies are 10^{42}, which is still too many to fully explore. Applying general RL algorithms on such a big policy space would still be intractable, requiring a huge number of explorations.

Through this pilot study, we realized that using a feature-based state representation requires a significant amount of exploration even after reducing the number of features. We decided to pursue an alternate approach which we will now describe.

6 Preliminary Study on Using Bayesian Optimization

6.1 Approach

In the second attempt, we decided to simplify the instructional policy space directly by parameterizing it. We used student models to achieve this. Student models are statistical models that that track and represent the state of a student's learning. Based on the assumption that a repeated practice on problems of the same skill can lead to mastery, these models have been widely studied in the educational data mining setting. Bayesian knowledge tracing (BKT) is a good example. It is based on a hidden Markov model, in which the hidden states correspond to whether the student knows or does not know the material and the observations correspond to whether the student got the problem correct [8]. Figure 3 shows the general BKT model. I, N, K are hidden states referring to the initial state, the state at which the student did not understand the material, and the state at which he/she did understand the material respectively. C, W are the observed state, referring to the state at which the student got the problem wrong and correct respectively. Starting from an initial state, p_i denotes the probability of knowing the material initially. p_t denotes the probability of transitioning from the "not known" to "known" state. p_g denotes the probability of getting the

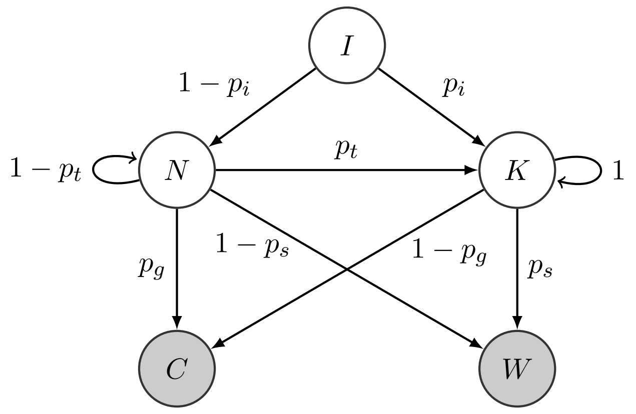


Figure 3: Bayesian Knowledge Tracing model

problem correct even if the student does not know the material. p_s refers to the probability of getting the problem wrong even if the student knows the material.

Another alternative is the performance factor analysis (PFA), which is based on logistic regression [18]. Unlike BKT, PFA does not have an explicit model of mastery, but fits the probability of getting the next problem correct using logistic regression based on how many questions the student got correct so far. In our experiment, we used the following logistic regression model.

$$y = \beta_0 + \beta_c \#(\text{correct}) + \beta_w \#(\text{wrong})$$

$$p = \frac{1}{1 + e^{-y}}$$

where p is the probability of getting the next problem correct.

Both models are widely used in practice to infer student's mastery of the material.

Using the concept of mastery learning, we can formulate our problem into an optimal stopping criterion problem. Instead of choosing the next skill to teach, we choose whether to stop giving the next problem corresponding to the skill or not. By solving the optimal stopping problem for each skill separately, we can obtain the best policy to teach the students. BKT and PFA models can be converted into an optimal stopping policy by defining a threshold such that when the probability of getting the next problem correct exceeds the threshold we stop. Thus, using this approach, the policy space can be parameterized using 5 parameters (p_i, p_t, p_g, p_s, threshold) for the BKT model and 4 parameters (β_0, β_c, β_w, threshold) for the PFA model. Note that there exist numerous intelligent tutoring systems that use mastery learning with a BKT student model, which is identical to the BKT policy class we consider here.

After parameterization, the problem is simplified to finding the optimal set of parameters that lead to maximum reward using the least number of data points. One approach might be to fit the student model based on a few data points, and set a fixed threshold. Intuitively, if we have a perfect statistical model of student learning, then this is sufficient for a good instructional policy. However, since the models can never exactly depict the real-world, it is not clear whether such a policy will work well in practice or whether there exists another policy in the chosen policy space that can achieve higher rewards.

We decided to take a more direct approach to the problem using Bayesian optimization (BO). BO finds the optimum point of a black box function. With an assumption that close points will have close function values, BO fits a Gaussian process on the samples observed so far, inherently modeling the expected value at each point as well as the uncertainty of this estimate. BO aims at reducing the number of evaluations necessary to find the optimum value, which matches well with our problem setting.

BO have been used to find good policies for reinforcement learning by Lizotte et al. [14] to learn gaits under uncertainty. Since then there have been numerous application of BO in the field of robotics, and research showed promising results. In the work by Lindsey et al. [13], they use GPs to efficiently sample instructional policy for their course platform. However, their policies are deterministic and non-adaptive meaning that they are trying to find a best ordering of problems that can work for all students. In our case, the policies are non-deterministic and adaptive, and as such, we utilized student models to parameterize the policy, and used BO to find the best set of parameters that lead to the highest reward.

Before we started the real experiment using BO, We first tested whether the policy identified by BO is better than the policy generated by model fit.

6.1.1 Using Data to Fit Models or Directly Searching for Policy Parameters

To compare the performance of the policy chosen by the two different approaches, we simulated a model mismatch setting, where the real-world situation was simulated using a HMM while the student model was a logistic regression model. The real-world model was used to generate the training and testing datasets, each containing 100 sequences. According to a predefined set of parameters, we simulated the student's progress in the course, expressed as a sequence of binary variables mentioned above. The policy was parameterized using the PFA model, and thus had 4 parameters (β_0, β_c, β_w, threshold). The objective is defined as follows. At the point where the policy decides to stop, we check whether the real probability of getting the next question correct using the real-world model exceeds the real threshold. If it does not, the policy fails and gets a value of 0. If it does, we divide it by the number of problems that the policy gave, so that the policy is biased towards giving fewer problems while teaching the students.

We used 3 different algorithms to find the best set of parameters for the objective function defined as follows.

$$f(\pi) = \frac{\mathbb{I}(p > \text{threshold})}{l}$$

where π is the policy, p is the probability of getting the next problem correct according to the HMM, and l is the number of problems given to the student.

The first algorithm was a brute force search on the policy space to find the best set of parameters. We enumerated all possible combination of the parameter values each discretized to 100 different values. For each set of parameters, we ran simulation on 100 training sequences, computing the objective value. We picked the set of parameters with the best objective value and used it to run simulations on 100 testing sequences to get an average objective value. The whole process was done 10 times for robustness. Although inefficient, such process allowed us to obtain an approximation to the best set of parameters based on the objective function.

The second algorithm used the maximum likelihood estimates of the PFA model parameters and a fixed threshold to form the policy. Using the 100 training sequences, we used Scikit-learn API [3] to

[3]http://scikit-learn.org/

	Mean Objective Value
Brute Force	0.316 ± 0.006
MLE	0.173 ± 0.007
BO Iteration 50	0.299 ± 0.118
BO Iteration 100	0.291 ± 0.125

Table 2: Mean objective value of the policies chosen by the three algorithms (Brute force, MLE, BO) with the real-world parameters $p_t = 0.25$, $p_l = 0.25$, $p_g = 0.3$, $p_s = 0.1$

fit the logistic regression parameters without regularization. Then, using the real threshold as the threshold for the policy, we applied the policy on 100 testing sequences, computing the objective values for each set of parameters. We note that by using a different threshold we may have gotten a different result, but finding such optimal threshold requires additional sampling, so we arbitrarily fixed this parameter. Similar to the first strategy, we ran the whole process 10 times or robustness.

The final algorithm used BO to compute the best set of parameters. We used the Spearmint API [21] to run Bayesian optimization. In each iteration, we made a simulation using the sampled set of parameters and returned the objective value computed. BO then computed the current best set of parameters and the set of parameters to test next. We took the best set of parameters computed at iteration 50 and iteration 100 for evaluation. Similar to the other two strategies, the parameters were used to simulate on 100 testing sequences, and the objective value was averaged. The whole process was repeated 10 times.

The simulation was performed with two sets of model parameters for the underlying HMM model. Note that p_l is the initial probability of knowing, p_t is the probability of transitioning from the unknown to known state, p_s is the probability of getting the question wrong when the student knows the skill, and p_g is the probability of getting the question right when the student does not know the skill. The threshold for both experiments was set to 0.7.

In the first experiment, we used the following set of parameters: $p_t = 0.25$, $p_l = 0.25$, $p_g = 0.3$, $p_s = 0.1$, and the results are in Table 2. In the second experiment, we used the following set of parameters: $p_t = 0.3$, $p_l = 0.1$, $p_g = 0.2$, $p_s = 0.25$, and the results are in Table 3.

Compared to the mean objective value of the brute force method, the policy chosen by model fit (second algorithm) performed significantly worse. This implies that student model obtained simply by fitting the model on the data does not yield a good policy. On the other hand, policy chosen by BO with only 50 data points performed better than the policy chosen by model fit. (p-value obtained through t-test: 0.023, 0.097 respectively for the two cases) Through simulation, we observed that even with a small number of iterations, BO can find policies with reasonable performance. This was a very encouraging preliminary evidence favoring the use of BO to optimize our policy.

6.2 Method

6.2.1 Apparatus and materials

In the second study, each participant interacted with our system only through the course platform. Each participant received a distinct user account to login to our online course platform. At the

	Mean Objective Value
Brute Force	0.182 ± 0.005
MLE	0.118 ± 0.012
BO Iteration 50	0.162 ± 0.047
BO Iteration 100	0.163 ± 0.042

Table 3: Mean objective value of the policies chosen by the three algorithms (Brute force, MLE, BO) with the real-world parameters $p_t = 0.3$, $p_l = 0.1$, $p_g = 0.2$, $p_s = 0.25$

end of the pre-assessment as well as all other problems, the system picks the next problem to give to the student according to some policy. A button appears in the student's page navigating them to the chosen problem. If the student navigates to a wrong problem (by mistakenly clicking on the menu), the problem is not shown, and the student is sent back to the correct problem. Such process is repeated until the student is navigated to the post-assessment, which marks the end of the course.

The course platform on histograms introduced in Section 4 was used again as the material for this study. As a recap, each course began with a pre-assessment consisted of 13 problems. Afterwards, they were given a variable sequence of descriptive problems designed to teach the participant about histograms. Students were provided the post-assessment when either no more problems were available or the underlying policy halted and indicated the student should take the test. In this study, the problems were classified into 8 different categories according to which skill they were aimed at teaching. These skills were 1)interpreting x axis, 2)interpreting y axis, 3)center of histograms, 4)spread of histograms, 5)skew of histograms, 6)building histograms from data, 7)matching histogram to description, 8)matching description to histogram. Compared to the previous study, we removed topics that were too ambiguous and divided the existing categories into smaller categories.

6.2.2 Participants

In the previous study, we realized that recruiting subjects through email was too slow. For each experiment, we required more than 30 participants, but recruiting 16 participants took a month. To solve this issue, we decided to use the Amazon Mechanical Turk (MTurk)[4] to recruit students. As a crowdsourcing platform, MTurk has access to people worldwide looking for paid tasks. To integrate MTurk to our system, we used the Psiturk API[5].

The challenge with using MTurk was screening out workers who are not serious or already know statistics, since the data generated by these workers would be invalid. As an initial step, we accepted workers who has a successful task completion rate of more than 95%. We also did a screening on the pre-assessment, accepting only those who got less than 8 out of 13 correct to screen out those who already know statistics and rejecting those who finished the test in less than 2 minutes. Figure 4 shows the distribution of pre-assessment scores of MTurk workers who were accepted to participated in the study.

[4]https://www.mturk.com/
[5]https://psiturk.org/

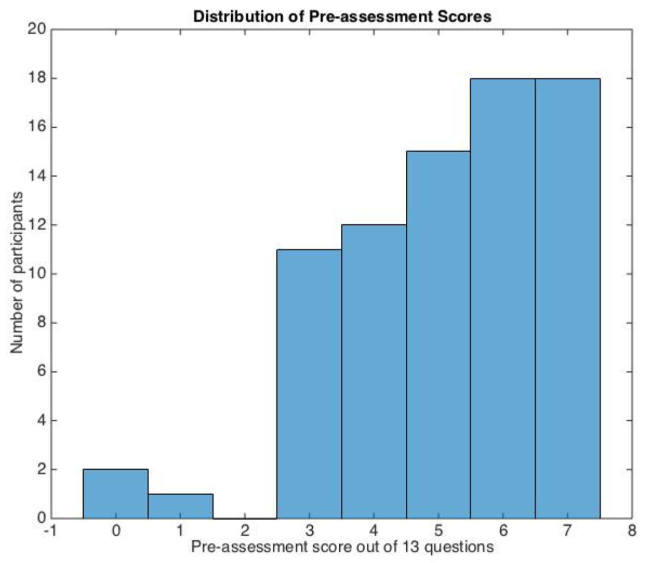


Figure 4: Histogram of the pre-assessment scores of the MTurk workers who got accepted

We gave a base pay of \$1 to all workers who accepted the task, and \$8 for those who successfully finished the course meaning that they solved all 13 problems on the post-assessment. We provided an additional bonus between \$0 and \$9 based on the post-assessment score and time spent on the course as follows.

$$\text{bonus} = 3 \times \mathbb{I}(\text{time spent} > 1 \text{ hour}) + 0.5 \times \#(\text{correct on post-assessment})$$

The additional bonus was designed to incentivize the workers to focus on the course.

6.2.3 Instructions to Participants

Once a participants accepts the task on MTurk, we give an automatically generated ID and password to be used to login to the online course. We tell them about the general structure of the course, and how to navigate to the next problem. We also specify the payment policy for the study, mentioning that there is an additional bonus according to their performance throughout the course. Lastly, we warn the workers that their is a pre-screening step in the first ten minutes, in which they can be stopped from continuing. We do not tell them exactly how the pre-screening is done to prevent gaming of the system. After the set of instructions are delivered, they are navigated to the pre-assessment page, where they start the experiment.

6.2.4 Design

The experiment is divided into three components: 1)run BO to obtain a new set of policy parameters, 2)automatically recruit a new subject to take the course, and 3)lead the subject through the adaptive course. For the BO component, we used the Metric Optimization Engine (MOE) API[6]

[6]https://github.com/Yelp/MOE

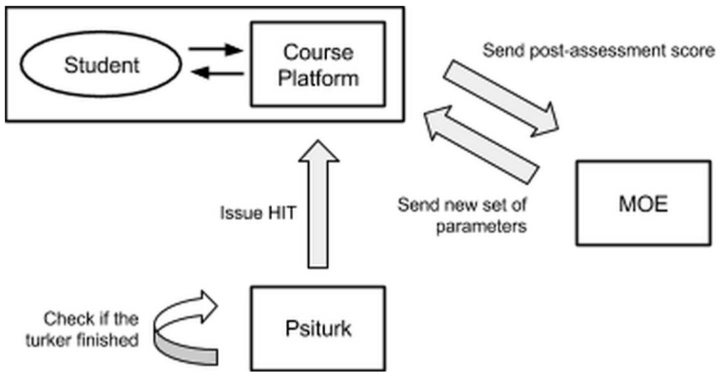

Student
Course Platform
Send post-assessment score
Send new set of parameters
MOE
Issue HIT
Check if the turker finished
Psiturk

Figure 5: Summary of the experiment pipeline where MOE is a BO API, Psiturk is an API for Amazon Mechanical Turk

developed by Yelp because of its ease of integration with the rest of the system and faster computation. In the beginning of the experiment, it generates the first set of parameters to try out, and sends it to the course platform. When a subject finishes the course, the BO component receives the post-assessment scores, and computes the objective value. BO then updates and determines the next set of policy parameters to try. Here, we run 8 separate BO instances for each of the different skills to accommodate the varying difficulty of each skill. In order to control the noise of the post-test score when divided into different skills, the objective function was designed as follows.

$$f(\pi) = \frac{p_s + \mathbb{I}(p > 9)}{\sqrt{l}}$$

where π is the policy, p_s is the normalized post-assessment score for the specific skill, p is the overall post-assessment score, and l is the number of problems given. Note that this function can be designed in any other way according to different priority among the objectives.

The second component recruits new participants through MTurk. To integrate MTurk to our system, we used the Psiturk API[7]. Psiturk deals with the process of creating and advertising HITs, as well as paying for the workers. This component continuously runs in the background throughout the experiment, checking whether the previous worker finished or not. Once the previous worker finishes, it creates a new task to gather the next worker.

The last component is the adaptive course platform. For each of the 8 skills, a separate BKT is constructed using the different set of parameters corresponding to each skill. After each problem, the probability of getting the next problem in the skill correct is computed for each BKT. For those that do not exceed the threshold, we collect the problems corresponding to the skill that is not given yet, merge into one set, and pick the next problem randomly from the set. The process continues until all skills have passed the threshold. At this point, we give the post-assessment and send the post-assessment result to the BO component.

The three components runs on separate processes dynamically interacting with each other to make the pipeline efficient and continuous. A summary of the pipeline is shown in Figure 5.

[7]https://psiturk.org/

6.2.5 Procedure

Data was gathered in three phases. In the first phase, we ran gathered data according to a baseline policy, in which we gave all the questions in random order to the students. Compared to giving only a subset of the material, we would expect the subject's post assessment score to be similar or higher using this baseline policy, since the subject is given more practice. However, we are particularly interested in balancing post assessment performance with the number of problems given, which measures time in a coarse way. Thus, the objective value defined in the previous section would not necessarily be the highest, but a reasonable one that we would hope to be able to find or exceed using BO with a limited number of subjects.

In the second phase, we started running BO. In each iteration, BO picks the policy parameters to test next, and the objective value achieved by using the policy is inputted into the BO server. We ran 30 iterations of BO, so 30 participants were recruited in the second phase.

After 30 iterations, we found the set of parameters with the highest expected reward according to the Gaussian process, then used this set on 10 participants as the final phase of the data gathering process. This allowed us to see which policy BO was converging to and evaluate it objectively.

6.3 Result

Table 4 compares the objective values of the baseline experiment where students are given all the problems, and those of the 10 experiments run using the learned set of parameters. Except the skill for interpreting y-axis, the policy chosen by Bayesian optimization performs as well as the baseline policy in terms of the objective value. For the skill for building histograms, the learned policy does significantly better than the baseline policy with a p-value of 0.04 using t-test. Other comparisons show no statistical significance.

Interestingly, the policy gave no problems to the students for 6 of the skills, and the average improvement was 2 rising from an average pre-assessment score of 4.42 to post-assessment score of 6.42, while in the baseline experiment the average improvement was 4. This can also be observed in Figure 6, which plots the pre-assessment scores and the post-assessment scores of the participants. The blue points refer to the subjects who followed the baseline policy, and the red points refer to those who followed the learned policy. The majority of participants who received all the problems improved by 2 or more points. (Blue dashed line indicates improvement by 2 points) Significant number of participants who followed the learned policy did not improve more than two points.

6.4 Discussion

The experiment result revealed the strengths and drawbacks of the algorithm. First of all, regardless of whether the policy made sense, the policy given by BO after only 30 iterations produced a high objective value that met or exceeded a hand defined decent baseline policy. Thus, in terms of sample efficiency, the algorithm was successful, and showed that BO may be feasible to quickly learn to act well. Nevertheless, BO was learning to not give any problems corresponding to a number of skills because such policy led to a higher objective function value. We started searching for potential causes to this behavior.

First, we investigated the design of the objective function. In our case, the objective function was inversely proportionate to the square root of the length of the sequence. As such, the gain from

Skills	Baseline	Learned Parameters
Matching description to histogram	0.631	0.750
Interpreting y axis	0.566	0.537
Matching histogram to description	0.529	0.625
Interpreting center	0.750	0.875
Interpreting shape	0.350	0.708
Interpreting x axis	0.618	0.640
Building a histogram	0.161*	0.500*
Interpreting spread	0.917	0.917

Table 4: Average objective values of the baseline policy and the policy learned by BO

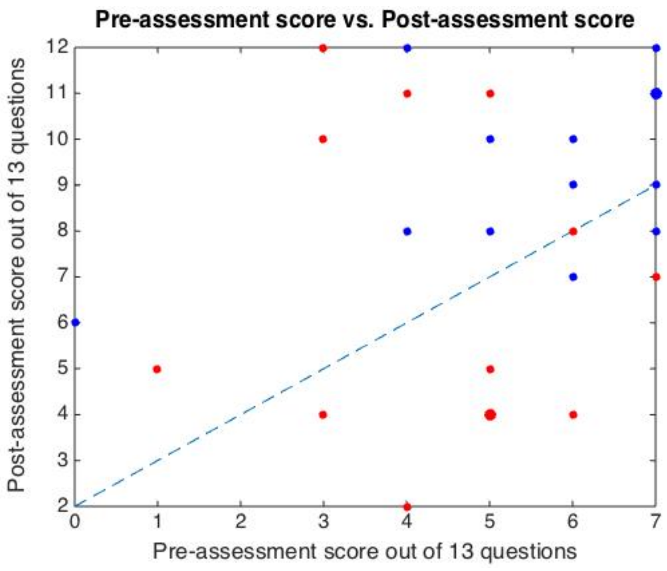


Figure 6: Pre-assessment score vs. Post-assessment score for the subjects following two different policies (Blue: baseline policy, Red: learned policy) Size of the dot was set according to number of students with the same pre and post-assessment scores

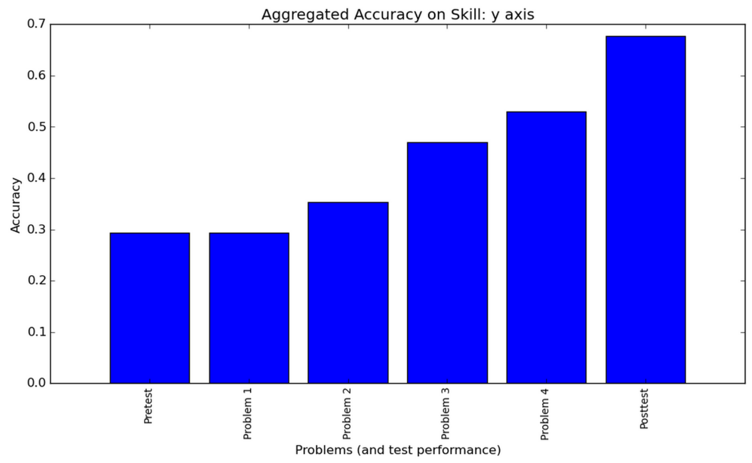


Figure 7: Accuracy on y-axis problems as the course progressed

giving one less problem was very large compared to getting one more question correct on the post-assessment. Thus, the algorithm favored giving no problems, which may lead to less performance on the post-assessment, over giving more problems to get better performance. We can alter the objective function so that the function is for instance linear for both post-assessment score and the number of problems given, changing the priority of the objectives that BO attempts to optimize.

Another potential cause for such behavior may be the design of the course problems and assessments. For the method to work, our problems must actually help the student understand a concept, and the assessment must accurately measure the understanding of the skill tested. In other words, as we give students more problems on the skill we should see an improvement on the assessment. To find out whether our problems satisfied this condition, we analyzed the baseline experiment data for each skill. For some skills there was a positive improvement. An example is y-axis, which teaches what the y-axis variable represents. Figure 7 shows the accuracy of students on the y-axis problems as they progress through the course. However, on some, there was no positive correlation. As shown in Figure 8, for the general histogram skill, there is no positive correlation. One reason may be that the skill is not specific enough. Since the problems corresponding to the general histogram skill may require different sub-skills. In this case, giving more problems in the skill may not necessarily improve the accuracy. Another reason can be due to the different difficulty of the problems. Looking at the fact that BO did not give any problems for the skills with less improvements, such behavior can signal the course designer to create better problems or assessment questions. This can be very useful when improving the quality of the course platform.

7 Conclusion

Through our work, we aimed at creating a sample efficient learning algorithm that can produce good instructional policies for an online educational platform. In the first attempt, we wanted to try to use AfD as a way to go beyond expert performance while reducing the amount of data needed. However, after a pilot study using the approach, we realized that the approach was not reasonable. Instead we moved on to the next attempt of using a student model to parameterize the policy space. We used BO to perform direct search on the chosen policy space using the least

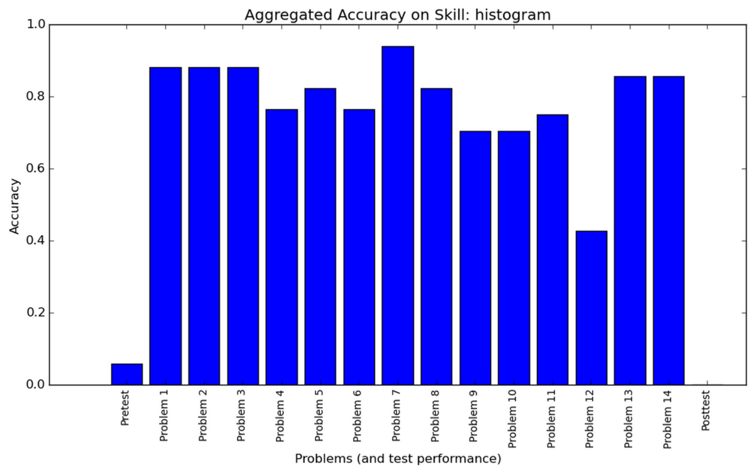


Figure 8: Accuracy on general histogram problems as the course progressed

number of data points. In only 30 iterations, BO managed to find a policy with good objective function values, but it was showing unanticipated behaviors, which we investigated the reason for.

There are a number of interesting directions for future work. First of all, one is to test different objective functions that reduce the effect of the number of problems given, and place a higher weight on the post-assessment score. This will help characterize the sensitivity of the effectiveness of BO to our objective function, in terms of sample efficiency. Second is to explore different approaches to utilizing pre-sampled data points to prevent exploration of meaningless points, which happens especially at the beginning of the training phase. Third is to extend the course platform to provide better content.

References

[1] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

[2] B. S. Bloom. The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational researcher*, pages 4–16, 1984.

[3] R. I. Brafman and M. Tennenholtz. R-max-a general polynomial time algorithm for near-optimal reinforcement learning. *The Journal of Machine Learning Research*, 3:213–231, 2003.

[4] M. Chi, K. VanLehn, D. Litman, and P. Jordan. Empirically evaluating the application of reinforcement learning to the induction of effective and adaptive pedagogical strategies. *User Modeling and User-Adapted Interaction*, 21(1-2):137–180, 2011.

[5] R. E. Clark and F. Estes. Cognitive task analysis for training. *International Journal of Educational Research*, 25(5):403–417, 1996.

[6] L. C. Cobo, P. Zang, C. L. Isbell Jr, and A. L. Thomaz. Automatic state abstraction from demonstration. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, page 1243. Citeseer, 2011.

[7] A. Corbett. Cognitive computer tutors: Solving the two-sigma problem. In *User Modeling 2001*, pages 137–147. Springer, 2001.

[8] A. T. Corbett and J. R. Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User modeling and user-adapted interaction*, 4(4):253–278, 1994.

[9] R. DelMas, J. Garfield, and A. Ooms. Using assessment items to study students difficulty reading and interpreting graphical representations of distributions. In *Proceedings of the Fourth International Research Forum on Statistical Reasoning, Thinking and Literacy. University of Auckland*, 2005.

[10] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18, 2009.

[11] J. J. Kaplan, J. G. Gabrosek, P. Curtiss, and C. Malone. Investigating student understanding of histograms. *Journal of Statistics Education*, 22(2), 2014.

[12] M. G. Lagoudakis and R. Parr. Least-squares policy iteration. *The Journal of Machine Learning Research*, 4:1107–1149, 2003.

[13] R. V. Lindsey, M. C. Mozer, W. J. Huggins, and H. Pashler. Optimizing instructional policies. In *Advances in Neural Information Processing Systems*, pages 2778–2786, 2013.

[14] D. J. Lizotte, T. Wang, M. H. Bowling, and D. Schuurmans. Automatic gait optimization with gaussian process regression. In *IJCAI*, volume 7, pages 944–949, 2007.

[15] M. Lopes, B. Clement, D. Roy, and P.-Y. Oudeyer. Multi-armed bandits for intelligent tutoring systems. *arXiv preprint arXiv:1310.3174*, 2013.

[16] Y. Ma. Exploration of the application of mooc to college education. In *2014 International Conference on e-Education, e-Business and Information Management (ICEEIM 2014)*. Atlantis Press, 2014.

[17] T. Mandel, Y.-E. Liu, S. Levine, E. Brunskill, and Z. Popovic. Offline policy evaluation across representations with applications to educational games. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pages 1077–1084. International Foundation for Autonomous Agents and Multiagent Systems, 2014.

[18] C. H. K. K. Pavlik, P.I. Performance factors analysis - a new alternative to knowledge. In *Proceedings the 14th International Conference on Artificial Intelligence in Education*, pages 531–538, 2009.

[19] A. N. Rafferty, E. Brunskill, T. L. Griffiths, and P. Shafto. Faster teaching via pomdp planning. *Cognitive Science*, 2015.

[20] M. Riedmiller. Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In *Machine Learning: ECML 2005*, pages 317–328. Springer, 2005.

[21] J. Snoek, H. Larochelle, and R. P. Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012.

[22] A. L. Strehl, L. Li, E. Wiewiora, J. Langford, and M. L. Littman. Pac model-free reinforcement learning. In *Proceedings of the 23rd international conference on Machine learning*, pages 881–888. ACM, 2006.

[23] A. L. Strehl and M. L. Littman. A theoretical analysis of model-based interval estimation. In *Proceedings of the 22nd international conference on Machine learning*, pages 856–863. ACM, 2005.

[24] N. K. Ure, A. Geramifard, G. Chowdhary, and J. P. How. Adaptive planning for markov decision processes with uncertain transition models via incremental feature dependency discovery. In *Machine Learning and Knowledge Discovery in Databases*, pages 99–115. Springer, 2012.

[25] M. van Otterlo and M. Wiering. Reinforcement learning and markov decision processes. In M. Wiering and M. van Otterlo, editors, *Reinforcement Learning*. Springer, 2012.