Adaptive Teaching: Learning to Teach

by

Aazim Lakhani
Bachelor of Computer Engineering, Mumbai University, Mumbai, 2009

A Project Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

in the Department of Computer Science

Adaptive Teaching: Learning to Teach

by

Aazim Lakhani
Bachelor of Computer Engineering, Mumbai University, Mumbai, 2009

Supervisory Committee

Dr. Nishant Mehta, Supervisor
(Department of Computer Science)

Dr. George Tzanetakis, Departmental Member
(Department of Computer Science)

Supervisory Committee

Dr. Nishant Mehta, Supervisor
(Department of Computer Science)

Dr. George Tzanetakis, Departmental Member
(Department of Computer Science)

ABSTRACT

Traditional approaches to teaching were not designed to address individual student's needs. We propose a new way of teaching one that personalizes the learning path for each student. We frame this use case as a contextual multi-armed bandit (CMAB) problem a sequential decision-making setting in which the agent must pull an arm based on context to maximize reward's. We customize a contextual bandit algorithm for adaptive teaching to present the best way to teach a topic based on contextual information about the student and the topic the student is trying to learn. To streamline learning, we add an additional feature which would allow our algorithm to skip topic's that a student is unlikely to learn. We evaluate our algorithm over a synthesized unbiased heterogeneous dataset to show that our baseline learning algorithm can maximize reward's to achieve results similar to an omniscient policy.

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

I would like to thank:

Dr. Nishant Mehta for his dedication, commitment and insights, without which I would not have been able to overcome the gaps I could not see.

Almighty One for giving me the courage, belief and strength to pursue my ideas.

Mom for her blessings and prayers.

DEDICATION

I dedicate this project to my family, to whom I owe both the joy and pain of growing up.

Chapter 1

Introduction

The quest for a fully personalized learning experience began with the development of intelligent tutoring system's (ITSs) [6, 14, 29, 31]. However, to date, ITS's are primarily rules-based which requires domain expert's to consider every possible learning scenario that students can encounter and then manually specify the corresponding learning action's in each case. This approach is not scalable since it is both labor-intensive and domain-specific [16].

Machine learning-based personalized learning system's have shown great promise in reaching beyond ITS to scale to large numbers of course's and student's. These system's automatically create personalized learning actions for each individual student to maximize their learning. Examples of action's include reading a textbook section, watching a lecture video, interacting with a simulation or lab, solving a practice question, etc. Instead of domain-specific rule's machine learning algorithm's are used to select action's automatically by analyzing the data student's generate as they interact with learning resource's [16].

The goal of this project is to design a learning algorithm, which could adapt based on student's feedback to help them learn effectively.

1.1 Use Case

There is no universal best way to explain a topic. The best way is subjective to every student. Unless we explore different way's to teach a topic, we cannot find a policy which would help map different student's to explanation's conducive for them. Once, we have such a policy we can use it to teach student's effectively. **This is**

the exploration-exploitation dilemma in which we have to find a trade-off between two competing goal's: maximizing student's satisfaction in the long run, while exploring uncertainties in student's preference's [3]. For example, an adaptive teaching system should present different explanation's knowing student's preferences on learning. However, unless we try different way's of teaching it is not possible to say with certainty whether or not an explanation would help a student learn effectively. We use the term adaptive teaching to avoid confusing it with adaptive learning used in machine learning literature. In the education domain, these term's are used interchangeably.

We represent this use case as a contextual bandit's problem. We use contextual information about the student such as their preference's to learn through *visual, text, demo-based, practical, activity-based, step-by-step, lecture, audio-based explanations, as well as, self-evaluation and pre-assessment of student's*. We also use contextual information about the content's used to teach a topic, by rating them in terms of *ease of understanding, simplicity, intuitiveness, depth in teaching, conciseness, thoroughness, ratings, abstractness, hands-on, experimental*. A **content item or arm's are different action's or ways a topic can be taught**. The reward would be the student's feedback to confirm their understanding of the topic they are trying to learn. The feedback can be through quizzes, interaction's with a content item, task's to name a few. **By pulling an arm, we obtain a reward drawn from some unknown distribution determined by the selected content item and the context. Our goal is to maximize the total cumulative reward**.

Let us make this more concrete by mapping this use case to teaching a class. In any school, a course comprises of multiple topic's. However now instead of a single way to teach everyone, there would be multiple way's to teach. These different way's to teach are called content item's. Student's give their feedback on the presented content. Behind the scenes, our learning algorithm takes information about the student *(also referred to as student context)*, topic, content item's*(also referred to as content context)* to find the best way to teach a student. This project extends the most cited contextual bandit learning algorithm, LinUCB (Linear Upper Confidence Bound) [18] to enhance it for our use case.

1.2 Motivation

The primary and perennial problem in education is the overwhelming challenge of teacher's being responsible for accomplishing learning mastery among a demographically diverse set of student's [23]. In traditional classroom's learning has largely remained a one-size-fits-all experience in which the teacher selects a single learning action for all student's in their class regardless of their diversity in needs, prior knowledge, skill's, learning style's, and background's. It is not feasible for teacher's to ensure their explanation's can cater to all student's. Hence there is a need for a system which could personalize teaching for student's to help them learn effectively as well as increase course engagement and progression.

Such system's would be adaptive, recognize different levels of prior knowledge among student's, as well as course progression based on student's skill and feedback from learning. This could reduce teacher's load to remediation to teaching and facilitating. These would adapt to individual student's learning pattern's instead of student's having to adjust to the way of teaching. They would provide timely and comprehensive data-driven feedback to recognize potential challenges that student's might come across as the course progresses.

1.3 Contribution

We suggest a novel baseline algorithm for our proposed adaptive teaching methodology which learns from student's and content's for each topic to create a personalized learning path for each student. It adapts dynamically based on student's feedback and learning preferences.

We also provide a skip feature which is meant to keep student's engaged to increase student retention as well as provide feedback to teacher's by recognizing the challenges faced by a student early in the course. Our online learning algorithm gives close to optimal result's over a synthesized unbiased heterogeneous dataset.

1.4 Organization

Chapter 1 provided a brief overview of our use case along with the need for an adaptive teaching system and how this project contributes to realizing it. Chapter 2 introduces the technical concept's used to represent our use case along with the algorithm we

customize for adaptive teaching. Chapter 3 describes prior work related to our use case using different approaches and how our work compares to them. Chapter 4 explains the algorithm created for adaptive teaching along with the skip feature. Chapter 5 describes the experimental setup which comprises the dataset synthesized to evaluate our algorithm. It also explains the evaluation strategy followed to examine our result's. Chapter 6 presents the result's of our experiment's and compares it's performance with respect to the best possible policy. Chapter 7 concludes this project by summarizing the contribution's and outline's possible avenues for future work.

Chapter 2

Preliminaries

This chapter briefly explains the key concept's used in this project.

2.1 Multi-armed bandit

Multi-armed bandit is a problem setting where an agent needs to make a sequence of decision's in time $1, 2, ..., T$. At each time t the agent is given a set of K arm's to choose and has to decide which arm to pull. After pulling an arm, it receives a reward for that arm, and the reward's of other arm's are unknown. However, arm pulled does not change the state of the world. This problem setting can be stochastic or adversarial. In a stochastic setting the reward of an arm is sampled from some unknown distribution, and in an adversarial setting the reward of an arm is chosen by an adversary and is not necessarily sampled from any distribution [32]. In this project, we assume the problem setting as stochastic.

Personalized recommender system's recommend item's (e.g., movie's, news article's) to the user's based on their predicted individual interest's on these item's. The users response helps the system improve their prediction [2]. However, the response to any particular item can only be available after these item's are recommended. If an item is never shown to the user's, the recommender system's cannot collect the response on these item's. Such problem's can be naturally modeled as a contextual bandit problem [30].

2.2 Contextual Bandits

In the theory of sequential decision-making, contextual bandit problem's [28] occupy a middle ground between multi-armed bandit problem's [7] and full-blown reinforcement learning (usually modeled using Markov decision processes along with discounted or average reward optimality criteria) [27]. Unlike bandit algorithm's which cannot use any side-information or context, contextual bandit algorithm's can learn to map the context into appropriate action's. However contextual bandits do not consider the impact of action's on the evolution of future context's. Nevertheless, in many practical domain's where the impact of the learners action on future context's is limited contextual bandit algorithm's have shown great promise. Examples include web advertising [1] and news article selection on web portal's [18, 13].

Formally a contextual bandit problem is a repeated interaction which proceeds over T rounds. At each round $t = 1, 2, ...T$ the environment reveals context's $x_t \in X$ which is used by the learner to pick an action $a_t \in A$ which gives a reward r_t revealed by the environment.The goal of the learner is to choose action's which would maximize cumulative reward $\sum_{t=1}^{T} r_t$.

We will now translate this problem setting for our adaptive teaching use case in which an algorithm A which proceeds in discrete rounds $t = 1, 2, 3, ...$ In round t:

1. The algorithm observes the student context x_s and a set A_t of content item's together with their feature vector's x_c for $a \in A_t$. X_t encapsulates x_s and the context x_c of all content item's available in round t.

2. Based on observed reward's in previous rounds, A chooses an arm $a_t \in A_t$. The arm a_t is estimated to have the highest expected reward. In a stochastic setting, the expected reward is given as the inner product of an unknown arm-dependent parameter $\theta_{a,t}$ and the context $x_{t,a}$, that is, $E[r_{t,a}|\mathbf{x_{t,a}}] = x_{t,a}^T \theta_{\mathbf{t,a}}$.

3. The student reveals the received reward r_t for arm a_t whose expectation depends on both the context X_t and the arm a_t.

4. The algorithm then improves it's content item selection strategy with the new observation (x_t, a_t, r_t). It is important to emphasize here that no feedback namely, reward r_t is observed for unchosen arm's $a \neq a_t$ [18].

2.3 Upper Confidence Bound (UCB)

A perpetual challenge in bandit algorithms is to find the right balance between exploration and exploitation (Section 1.1). Upper Confidence Bound (UCB) comprises of a family of algorithm's which try to find the best trade-off between exploration and exploitation. It is based on the principle of *being optimistic* by choosing action's which have the highest potential for reward. The intuitive reason that this works is that when acting optimistically, one of two things happens. Either the optimism was justified, in which case the learner is acting optimally, or the optimism was not justified. In the latter case, the algorithm takes some action that they believed might give a reward when in fact it does not. If this happens sufficiently often, then the algorithm will learn the true reward of this action and not choose it in the future [17]. UCB algorithm's estimate the expected reward for each arm by adding estimated sample mean of an arm with it's upper confidence bound.

We will refer to Figure 2.1 as an example to understand UCB. Let us assume we have three arm's a_1, a_2, a_3. The reward distribution for each arm after several rounds is a Gaussian distribution Q with mean μ and standard deviation σ. The y-axis is the probability of obtaining a certain reward for these arm's.

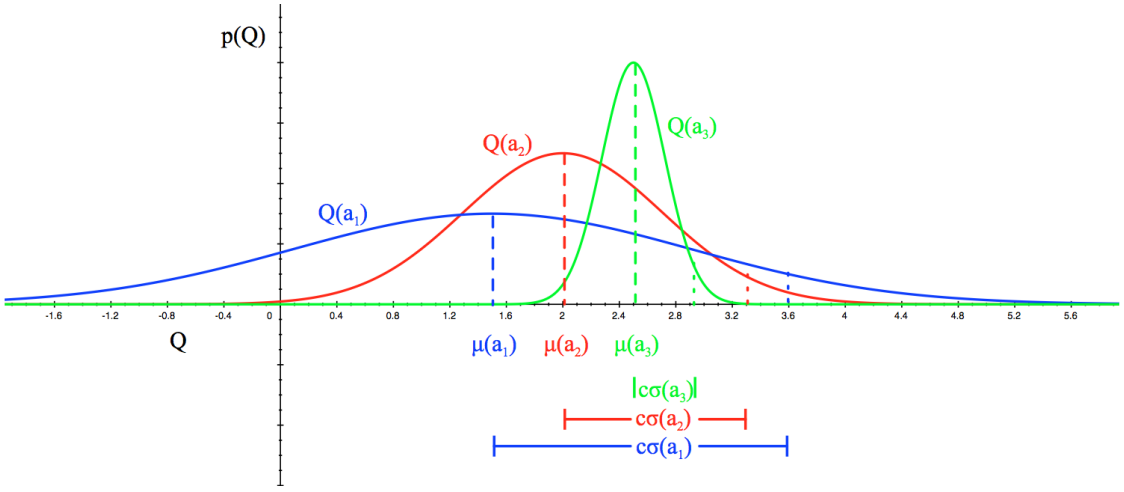

Figure 2.1: An example: UCB
[26]

The upper confidence bound for each arm is given by $c\sigma(a_i)$. The distribution shows that the sum of the expected mean and upper confidence bound is highest for a_1. Hence the UCB algorithm will select a_1. The reward received will reduce uncertainty around a_1. So for the next round, the algorithm once again finds the arm

with the highest sum for the expected mean and upper confidence bound. This is repeated for T rounds [9].

2.4 Linear Upper Confidence Bound (LinUCB)

LinUCB is a way to apply UCB to a more general contextual bandits setting where the UCB of each arm is computed efficiently by assuming the reward is linear, given as $E[r_{t,a}|\mathbf{x_{t,a}}] = x_{t,a}^T \theta_{\mathbf{t,a}}$. The estimated expected mean is parameterized over the context x_a for each arm a. At round t this is given as $\widehat{\theta}_a^T x_{t,a}$. The upper confidence bound around each arm a at round t is given as $\sqrt{x_{t,a}^T A_a^{-1} x_{t,a}}$. Here, A_a is the co-variance over the context data $x_{t,a}$ for each arm a at round t.

LinUCB introduces a hyper-parameter α, which allows us to control exploration over arm's. This is achieved by scaling the upper confidence bound by α. A higher value of α encourages exploration. As a result, the algorithm would need more rounds to explore before it begins exploiting. We can now compute **the expected estimated reward for an arm** a **at round** t **as** $p_{t,a} = \widehat{\theta}_a^T x_{t,a} + \alpha\sqrt{x_{t,a}^T A_a^{-1} x_{t,a}}$ [18].

Chapter 3

Related Work

Our use case could also be formulated using the partially observed Markov decision process (POMDP) framework. POMDPs model the student's latent knowledge states and their transition's to learn an action selection policy that maximizes reward received in the possibly distant future (long-term learning outcome). Previous work applying POMDPs to personalized learning has achieved some degree of success. However, learning a personalized learning schedule using a POMDP is greatly complicated by the curse of dimensionality. The solution quickly becomes intractable as the dimensions of the state and action spaces grow. Consequently, POMDPs have made only a limited impact in large-scale personalized learning application's involving large numbers of student's and learning action's [16].

A more scalable approach to personalized learning is to learn a policy, which maps context's to action's using the multi-armed bandit's (MAB) framework, which is more suitable for optimizing student's success. The simplicity of the MAB framework makes it more practical than the POMDP framework in real-world educational application's [16].

The work in [19] applies a MAB algorithm to educational game's to find trade-off between exploring learning resource's to accurately estimate arm means, while also trying to maximize user's test performance. Their approach is context-free and does not consider diversity among individual user's. The work in [21] collects high-dimensional student - computer interaction features as they play an educational game and uses them to search for a good teaching policy [16].

The work in [16] is focused on adaptive testing to assess a student's performance. They use contextual MAB to find question's to assess a student. The question depends on a student's response to earlier question's. At each round, they have all question's

to assess a student. Contrary to that we only have a restricted set of content item's available at each round. Our use case is focused on adaptive teaching to enable student's to learn.

The work's in [10, 15] both use a form of expert knowledge to learn a teaching policy. The approach of [10], in particular, uses expert knowledge to narrow down the set of possible action's a student can take. Our approach, in contrast, requires no expert knowledge and is fully data-driven [16].

The work in [25] found that various student response model's, including knowledge tracing (KT) [11], Item Response Theory (IRT) model's [20, 24, 5], additive factor model's (AFM) [8], and performance factor model's (PFM) [12] can have similar predictive performance yet lead to very different teaching policies. [16]. While these result's are interesting, we emphasize that the focus of the current work is to develop policy learning algorithm's rather than comparing student model's.

Chapter 4

Algorithm

This chapter presents the algorithm created for the adaptive teaching system. We first present the basic version of the algorithm (Section 4.1). We then explain the skip feature (Section 4.2), which could streamline learning.

The algorithm used is an extension of upper confidence bound (UCB)-based algorithm's [4] (Section 2.3). These algorithm's maintain estimates of the expected reward of each arm together with confidence bound around it. It then pulls the arm with the highest estimated reward which is equal to the sample mean plus the confidence bound. Based on the actual reward it updates the arm parameter's iteratively after each pull to make better decision in upcoming rounds. In this project we are using the most cited contextual bandit algorithm, namely LinUCB (Section 2.4).

Before we dive in it is important to note, that to better understand the algorithm we have divided the explanation into two halves. *The first half explains the overall flow without skipping whereas the second explains in-depth the function calls made in the first half along with skipping.* We are using bandit terminology to explain. **Arm** refers to a content item. **Payoff** is the algorithm's upwardly biased estimate of the expected reward, where the bias is due to the algorithm's use of an upper confidence bound rather than using the sample mean directly. A **round** comprises of computing the expected payoff for each content item; then presenting a content item with maximum expected payoff and getting student feedback for the content item.

Below are the notation's used in the algorithm.

Symbol	Meaning
α	Parameter to scale Confidence bound.
C	Confidence threshold to skip.
$\mathbf{x_s}$	Student context vector.
x_c	Content item's context matrix for a topic.
$\mathbf{x_t}$	Context vector at round t.
X_t / X_t^i	Context at round t. It combines $\mathbf{x_s}$ and all available x_c for topic i.
X_t^{i+1}	Context at round t. It combines $\mathbf{x_s}$ and all available x_c^{i+1} for topic $i+1$.
x_c^{i+1}	Content items context's for topic $i+1$.
a	An arm a for topic i.
a'	An arm a' for topic $i+1$.
A_t	Arm's available at round t.
$A_{t'}^{i+1}$	Arm's available for topic $i+1$ at round t'.
a_t^{i+1}	Arm a for topic $i+1$ at round t.
t	Current round t.
t'	Possible next round t'.
i	Topic being taught.
$i+1$	Next Topic in the sequence.
$p_{t,a}$	Expected payoff from arm a at round t.
$p_{t,a}^i$	Expected payoff from arm a at round t for topic i.
$p_{t',a'}^{i+1}$	Expected payoff from arm a' at round t' for next topic $i+1$.
X	Input features for skip classifier.
Y	Label to train the skip classifier.

Table 4.1: Algorithmic notation's

Note

- We are always on the current topic i, unless we explicitly specify next topic $i+1$.
- All vector's are **bold** faced lower cased.
- All set's are plain faced.

Algorithm 1 Teach with LinUCB

Hyper Parameter's : $\alpha \in \mathbb{R}_+$
C : Confidence threshold to skip
Inputs : Student context $\mathbf{x_s}$ and content context x_c of available arm's $a \in A_t$ for topic i at round t
Prepare context $X_t = \begin{pmatrix} \mathbf{x_s} \\ x_c \end{pmatrix}$
skip-enabled $\leftarrow$ False
while $A_t \neq \emptyset$ **do**
    a_t^i , $p_{t,a}^i \leftarrow$ EXPECTED-PAYOFF(X_t,A_t)
    skip-decision , $p_{t',a'}^{i+1} \leftarrow$ SKIPTOPIC($\mathbf{x_s}$, $p_{t,a}^i$, i)
    if skip-decision and skip-enabled is True **then**
        Move to next topic $i \leftarrow i + 1$
        break
    else
        Pull arm a_t and observe reward r_t
        $A_{a_t} \leftarrow A_{a_t} + \mathbf{x_{t,a_t}} \mathbf{x_{t,a_t}^T}$
        $\boldsymbol{b_{a_t}} \leftarrow \boldsymbol{b_{a_t}} + r_t \mathbf{x_{t,a_t}}$
        label $\leftarrow$ SETLABEL(r_t)
        TRAIN($\mathbf{x_s}$, $p_{t,a}^i$, $p_{t',a'}^{i+1}$,label)
        $t \leftarrow t + 1$
    if $r_t \neq 1$ **then**
        Remove $a_t \in A_t$
        skip-enabled $\leftarrow$ True
    else
        Move to next topic : $i \leftarrow i + 1$
        break

4.1 Basic Version

The basic version is without skipping. It explains the main flow of the algorithm. The next section explains the function's used along with skipping.

The algorithm requires two hyper-parameter's to be configured. The first one is α which scale's the confidence bound (Section 2.4). The second hyper-parameter is the confidence threshold C which decides confidence threshold that must be exceeded to skip a topic. Skipping is a feature to help reduce student's who are unlikely to learn from content item's available for a topic. It is meant to streamline learning. This could also be used by teacher's to recognize topic's that should be addressed in class.

We now explain how LinUCB (Section 2.4) helps the algorithm decide an arm to pull. Before we recommend a content item to a student, we need to prepare context X_t for the round t. It is prepared by combining the student context $\mathbf{x_s}$ with content item's context x_c for the topic i being taught. With the context X_t and arm's A_t, we use LinUCB to compute the expected payoff from each arm and return the arm a_t^i with the maximum expected payoff $p_{t,a}^i$ which must be pulled for topic i at round t.

Assuming the classifier does not recommend skipping, a student is presented with the content item a_t for topic i. After being taught the student sends a reward r_t to complete the round t. Now the round t is complete we update the arm parameter's A_{a_t} , $\mathbf{b_{a_t}}$ of the arm pulled. We then use this reward r_t to train the skip classifier to make better prediction's in upcoming rounds. The feature's for the classifier comprise of student's contextual information $\mathbf{x_s}$, expected payoff $p_{t,a}^i$ from the current topic i and the expected payoff $p_{t',a'}^{i+1}$ for the topic $i+1$.

If no reward r_t was sent by the student $\mathbf{x_s}$, then it implies the student was unable to understand topic i. In which case the algorithm removes the presented arm a_t and remains on the same topic i. However if a reward r_t was sent, then the student is moved to the next topic $i+1$. That completes the first half. The second half explains the function's briefly described above.

4.2 With Skipping

On line 6 (Section 4.1) of the algorithm we get the expected payoff $p_{t,a}^i$ estimated on pulling the arm a_t^i for the current topic i. Now to decide whether or not it should pull the arm or move to the next topic it calls the skip topic function.

The *SKIPTOPIC* function takes the student context $\mathbf{x_s}$, the expected payoff $p_{t,a}^i$

function SkipTopic(x_s, $p^i_{t,a}$, i)
    Get next topic $i+1$ from topic i
    Get arm's $A^{i+1}_{t'}$ and content context x^{i+1}_c for topic $i+1$
    Prepare context vector $X^{i+1}_{t'} = \begin{pmatrix} x_s \\ x^{i+1}_c \end{pmatrix}$
    $a^{i+1}_{t'}$, $p^{i+1}_{t',a'} \leftarrow$ EXPECTED-PAYOFF($X^{i+1}_{t'}$,$A^{i+1}_{t'}$)
    skip-decision $\leftarrow$ PREDICT(x_s, $p^i_{t,a}$, $p^{i+1}_{t',a'}$) to decide on skip
    return skip-decision , $p^{i+1}_{t',a'}$
function EXPECTED-PAYOFF(X_t,A_t)
    for $a \in A_t$ **do**
        Get $x_{t,a} \in X_t$
        if a is new **then**
            $A_a \leftarrow I_d$ (d-dimensional identify matrix)
            $\boldsymbol{b_a} \leftarrow 0_{d\times 1}$ (d-dimensional zero vector)
        $\widehat{\theta}_a \leftarrow A_a^{-1} b_a$
        $p_{t,a} \leftarrow \widehat{\theta}_a^T x_{t,a} + \alpha\sqrt{x_{t,a}^T A_a^{-1} x_{t,a}}$
    Choose arm $a_t = \arg\max_{a\in A_t} p_{t,a}$ with ties broken arbitrarily
    return $a_t, argmaxp_{t,a}$,
function PREDICT(x_s, $p^i_{t,a}$, $p^{i+1}_{t',a'}$)
    X $\leftarrow x_s$,i+1 , $p^i_{t,a}$, $p^{i+1}_{t',a'}$
    Y , confidence-score $\leftarrow$ Prediction from classifier
    if confidence-score <C **then**
        decision $\leftarrow 0$
    return decision , confidence-score
function TRAIN(x_s, $p^i_{t,a}$, $p^{i+1}_{t',a'}$, label)
    $X \leftarrow x_s, p^i_{t,a}, p^{i+1}_{t',a'}$, topic ,
    $Y \leftarrow$ label
    Train online SGD classifier
function SETLABEL(r_t)
    if r_t is 0 **then**
        label $\leftarrow 1$
    else
        label $\leftarrow 0$
    return label

for pulling arm a at round t for topic i and the current topic i. It uses the topic i to get a reference to the next topic $i+1$. Through the topic $i+1$ it gets content item's $A_{t'}^{i+1}$ and context data x_c^{i+1} associated those content item's. After combining the context's to prepare $X_{t'}^{i+1}$ it gets the maximum expected payoff $p_{t',a'}^{i+1}$ and the arm $a_{t'}^{i+1}$ to pull by passing the context vector $X_{t'}^{i+1}$ and arm's available for next topic $A_{t'}^{i+1}$. The expected payoff function returns an arm with the maximum estimated payoff. Skip topic function then calls the skip classifier to predict a skip-decision for the student context $\mathbf{x_s}$, along with the expected payoff from the current and the next topic to make a prediction.

The *EXPECTED-PAY0FF* function takes the context X_t, along with the arm's A_t available at round t. After an arm a_t is initialized with parameters A_a, b_a they are used to calculate the expected mean $\widehat{\theta}_a^T x_{t,a}$ and confidence bound $\sqrt{x_{t,a}^T A_a^{-1} x_{t,a}}$ for the arm. The confidence bound is scaled by α. The expected mean and the scaled confidence bound are added to give the expected payoff $p_{t,a}$ for arm a at round t. It then finds an arm a with maximum expected payoff $p_{t,a}$ and returns the expected payoff along with the arm a to be pulled.

The *PREDICT* function is used to predict whether the student should be moved to the next topic $i+1$ or should remain on the same topic i. It combines student context vector $\mathbf{x_s}$, the expected payoff $p_{t,a}^i$ for the current topic i and the expected payoff $p_{t',a'}^{i+1}$ for the next topic $i+1$ to prepare a feature vector X. It then gets a prediction from the binary supervised online Support Vector classifier with hinge loss to make a prediction Y and a confidence-score for it's prediction. If the confidence-score is less than the confidence threshold, then set the *decision* variable is set to 0 which implies no skipping. This is because a confidence score lower than the threshold implies that the classifier is not sufficiently confident about it's prediction.

The *TRAIN* function is used to train the skip classifier to make better prediction's. Similar to the predict function it combines student context vector $\mathbf{x_s}$, the expected payoff $p_{t,a}^i$ for the current topic i and the expected payoff $p_{t',a'}^{i+1}$ for the next topic $i+1$ to prepare a feature vector X. It set's the *label* to the output Y. Together they train the skip classifier.

The *SETLABEL* function is used to set the *label* to train the skip classifier. If the reward r_t for round t is set to 0 then the *label* is set to 1. This implies that since staying on the same topic did not give any reward, it would have been better to skip. If the reward r_t for round t is set to 1, then the *label* is set to 0. This implies that staying on the same topic was a good decision. The set *label* is then returned.

Chapter 5

Experiments

This chapter explains the dataset (Section 5.1) used to evaluate the learning algorithm. It then describes the environmental setup (Section 5.2) used for these experiment's. The next section explains how we evaluate our algorithm (Section 5.3) in absence of pre-existing benchmark's using an omniscient policy (Section 5.4). This is followed by section's which explain how the learning algorithm (Section 5.5) and the skip feature (Section 5.6) work in these experiment's.

5.1 Dataset

Machine learning algorithm's are data-driven. Due to the novelty of our approach to the best of our knowledge, there is no similar dataset available. Hence **we synthesized datasets to represent data generated by student's taking course's in an adaptive teaching environment.**

An honest attempt is made to synthesize an unbiased dataset representative of the heterogeneous student's and content item's. Biased datasets tend focus on targeted student group's (for instance, having many student's who give positive feedback). This could result in higher reward's. Contrary to this, our dataset is representative of diverse student and content data and is not skewed towards a particular student group or content type.

The contextual data is created from a uniform distribution U(0,1] sampled randomly to simulate the diverse nature of student preference's and content feature's.

5.1.1 Course's

We use the following courses for our experiment's.

1. *Course 1 :* A course which comprises of 10 topic's. It is taken by 50 student's. There are a total of 119 content item's for 10 topic's. So on an average, there are 12 content item's per topic. We use this course to find optimal hyper-parameter's (α and C) for our learning algorithm.

2. *Course 2 :* A course which has 25 topic's. It is taken by 100 student's. There are 329 content item's for 25 topic's. So on an average, there are 13 content item's per topic. We use this course for evaluation.

5.1.2 Context

We will assume there was a survey conducted among student's who were asked how should teaching streamline learning? Student's gave their preferences on a scale of 1 to 10 with 1 being least preferred and 10 being most preferred. These preference's were normalized.

Research has shown that student's prefer to learn a certain way. Though there is no unanimous consensus, there is a fair bit of research and understanding on the needs of a student. The feature's we consider are by no means exhaustive but a representative subset of the main feature's. The tables 5.1 and 5.2 describe the student and content context used for these experiment's.

Student Context	**Description**
Visual (S_V)	How much preference is given to visual explanation's (video, short-film, movie-clip, video blog's)?
Text (S_T)	How much preference is given to written explanation's (book's, article's, blog's, research paper's)?
Demo-based (S_D)	How much preference is given to live experiment's to help understand a concept?
Practical (S_P)	How much preference is given to an explanation, followed by a demo of the topic, and enabling student's to perform it?
Step-by-step (S_S)	How much preference is given to a guide to practice, try and understand a topic in a systematic way?
Activity/Task-based (S_AT)	How much preference is given to content item's which are interactive and require student's to participate?
Lecture (S_L)	How much preference is given to being passive and listen to an expert explain the topic?
Audio (S_A)	How much preference is given to audio explanations (podcast, music)?
Self-evaluation (S_SE)	Student's self-evaluate their readiness, motivation, excitement for the course.
Pre-assessment (S_PA)	Teacher's conduct a pre-assessment of the pre-requisites required for the course.

Table 5.1: Student context

Below (Figure 5.1) is a student context data point which shows a student preference. It tells us that this student prefers visual (S_V), text(S_T), demo-based(S_D) method's of learning, but does not prefer practical (S_P), activity-based(S_AT), and did not fare well in the pre-assessment(S_PA). The student does not mind step-by-step(S_S), lecture's(S_L) , audio's(S_A) method's of learning and believes he/she is ready for the course (S_SE).

	S_V	S_T	S_D	S_P	S_S	S_AT	S_L	S_A	S_SE	S_PA
0	0.87	0.82	0.88	0.36	0.6	0.06	0.66	0.56	0.66	0.07

Figure 5.1: Student context template

Content Context	**Description**
Ease of understanding (C_E)	How relatively easy is it to understand the content?
Simple/Intuition (C_I)	Does it provide a surface level or deep understanding of the topic?
Surface/In-depth (C_ID)	How much preference is given to live experiment's to help understand a concept?
Brief/Concise (C_C)	Is it short, to the point or descriptive, verbose and elaborative, keeping in mind that learner's have different levels of concentration and capacity to remember?
Thorough (C_T)	How well does the content item cover the topic?
Preference/Well reviewed/Well rated(C_R)	How well rated is the explanation?
Theoretical/Abstract (C_A)	How theoretical or abstract is the content item?
Practical/Hands on (C_P)	Is it something that can be tried or experienced?
Experimental/Task based (C_ETB)	Does it require a task to be completed to fully understand it, like collaboration with other student's or some research/findings?

Table 5.2: Content context

Below (Figure 5.2) is a content context data point prepared for the course. This content item is thorough(C_T), practical(C_P), and experimentally sound(C_ETB), but not in-depth(C_ID),concise(C_C), and abstract(C_A). It is moderate in term's of understanding(C_E), intuitiveness(C_I) and has positive reviews(C_R).

	C_E	C_I	C_ID	C_C	C_T	C_R	C_A	C_P	C_ETB
C_1_1	0.45	0.72	0.31	0.05	0.91	0.75	0.06	0.88	0.97

Figure 5.2: Content context template

Apart from the above contextual data, there is a course which is taught. For our

experiment's, we consider a typical course which comprises of topic's to be taught. These topic's are labeled as *T_1, T_2 ... T_25*. For e.g: T_1 refers to the first topic of the course. Each topic has between 5 to 20 different content item's. Each content is labeled in the format *C_topic-id_content-number*. For e.g: C_1_2 refers to the second content item for topic T_1.

We now have the required contextual information. Topic's in the course are taught in a sequence outlined by the teacher. This allows them to control the course sequence. Let us take an example to understand the data.

5.2 Environment

We run a simulation of a course being taken by student's with the omniscient policy and the learning algorithm deciding the content item to be presented for each student. It is an environment where several student's are taking the course at the same time. Both the omniscient policy and the learning algorithm work in online mode. The learning algorithm updates it's parameter's in each round to give better predictions.

5.3 Evaluation Strategy

Since there are no readily available benchmark's to compare our algorithm, we assume there exist an omniscient policy. This policy has optimal parameter's to recommend the best arm to pull.

We run the same course with an omniscient policy and the learning algorithm to evaluate our learning algorithm relative to the omniscient policy. The evaluation is conducted with and without skipping. Due to the stochastic nature of student's feedback both the omniscient policy and the learning algorithm will run for a different number of rounds. However, the total cumulative reward available is the same for both of them. Hence we evaluate them based on cumulative reward accumulated over all rounds.

We simulate the student feedback as a Bernoulli distribution. Here, the probability of success is the maximum expected reward computed by the omniscient policy. This reward for an arm a with optimal parameter's $\theta^*_{t,a}$ and with context vector $x_{t,a}$ at round t is given by $E[r_{t,a}|\mathbf{x_{t,a}}] = x^T_{t,a}\theta^*_{\mathbf{t,a}}$. It is passed to the Bernoulli distribution as the probability of reward for the presented content item. Based on the reward received by the learning algorithm the arm parameter's are updated to make better

decision's in the upcoming rounds. This experiment aims to find how well does our algorithm optimize the arm parameter's to match the omniscient policy.

5.4 Omniscient Policy

This policy knows all the probability distribution's. At every step of makes the way the best decision as it knows the true distribution's. It does not have to learn anything. It has optimal parameter's θ^* for each arm. Hence it is expected to maximize the cumulative reward.

This policy calculates the expected payoff for each arm a available for a topic. It then selects the arm which has maximum expected payoff.

5.5 Learning Algorithm

The learning algorithm can adapt to several student's at the same time to present a content item personalized for each student. For every topic, a student is trying to learn it gets the expected payoff for all available content item's. It check's whether it should skip to next topic or remain on the current topic. Skipping is activated only if the student gave no reward for a content item presented for the topic.

When a student is on a topic, the algorithm presents a content item that could maximize reward's. After working through the content item, the student shares feedback on the content item. If a reward is sent, then this implies that the student understood the concept and can be taken to the next topic. If no reward was sent, then the student may be presented with the next best content item for the same topic or could be moved to the next topic in the course sequence.

Once the student has shared feedback on the content item, the data is sent to train the skip classifier to make a better prediction in forthcoming rounds.

5.6 Skip Topic's

The learning algorithm checks with skip topic's to decide whether or not the content item should be presented for the current topic. Skip topic predicts this by using the student context along with the estimated payoff for the current topic and the estimated payoff for the next topic in the course sequence.

It makes this decision using an online supervised learning stochastic gradient descent classifier with student context along with the estimated payoff of the current and next topic to make a decision. The label for the classifier depends on the reward received for the topic. If a reward was sent then the label is set to 0 or else it is set to 1. Thus the classifier makes use of the feedback sent by all student's to recognize common topic's and content item's that student's find difficult so it could make a confident decision.

The aim to create skip topic's feature is to streamline learning for student's. If a student has been taught a topic once and was not satisfied with it, then there is the option to skip to the next topic or explain the same topic with a different content item.

The skip classifier is a linear Support Vector Machine estimator with hinge loss. The estimator is a regularized linear model with stochastic gradient descent (SGD) learning. The gradient of the loss is estimated, each sample at a time and the model is updated along the way with a decreasing learning rate. The regularizer is a penalty added to the loss function that shrinks model parameter's towards the zero vector using squared Euclidean norm [22].

Chapter 6

Results and Evaluation

This chapter presents results using the experimental set-up given in the previous chapter (Chapter 5). We evaluate the learning algorithm with respect to the omniscient policy. Before evaluation we need to first find optimal value's for hyper-parameter's α (Section 6.1) and confidence threshold C (Section 6.2). We then proceed to use these optimal value's to evaluate the learning algorithm with and without the skip feature. (Section 6.3).

6.1 Confidence Bound α

Finding an optimal value for α is important to learn faster as it scale's the confidence bound of each content item. An optimal value would find the right balance between exploration and exploitation. A higher value of α would imply the learning algorithm takes more rounds exploring which can lead to sub-optimal result's.

This parameter is configured for the learning algorithm and not the omniscient policy. We empirically evaluated an optimal value for α using course 1 (Section: 5.1.1). The graph (Figure 6.1) shows the cumulative reward for different value's of α.

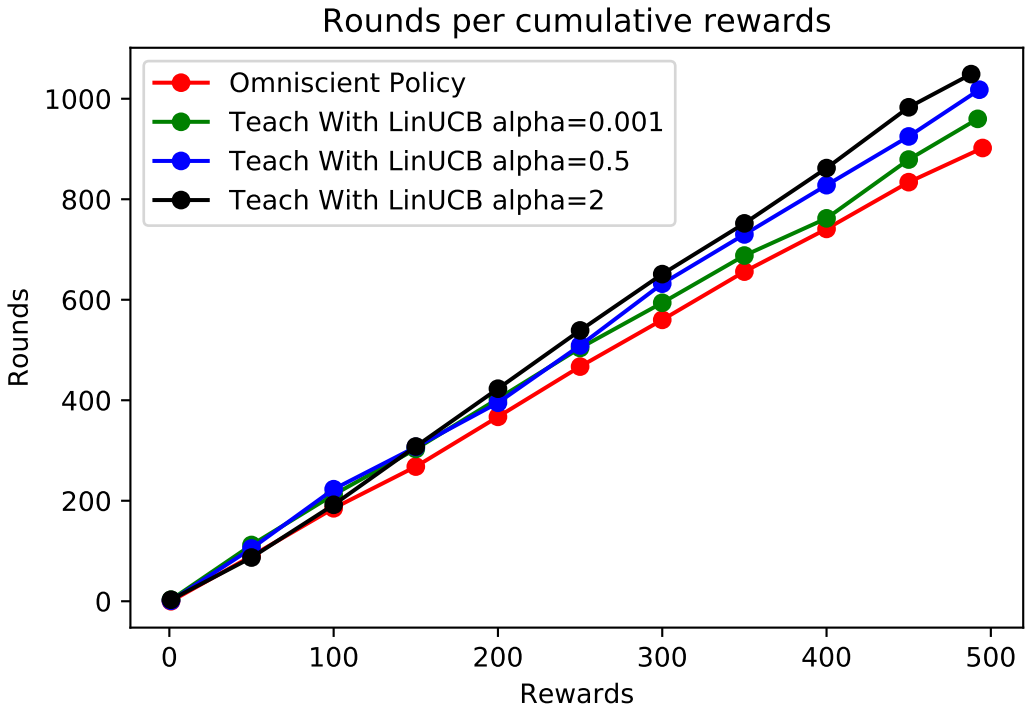

Figure 6.1: Rounds per cumulative rewards for α.

The graph (Figure 6.1) compares the omniscient policy with the learning algorithm for different values of α. It shows that the learning algorithm **took 960 rounds to maximize reward's when** $\alpha = 0.001$ compared to 1018 rounds required by $\alpha = 0.5$ and 1049 rounds required by $\alpha = 2$. On running repeated experiment's, we found that a value of α between 0 to 0.5 gives better result's.**We would be using** $\alpha = 0.001$ **to evaluate the learning algorithm.**

The graph (Figure 6.2) presents a different view of the above graph. It shows the number of rounds to reward's ratio at different interval's of reward's for different value's of α and compares it to the omniscient policy. It clearly shows that $\alpha = 0.001$ required the fewest rounds to maximize reward's.

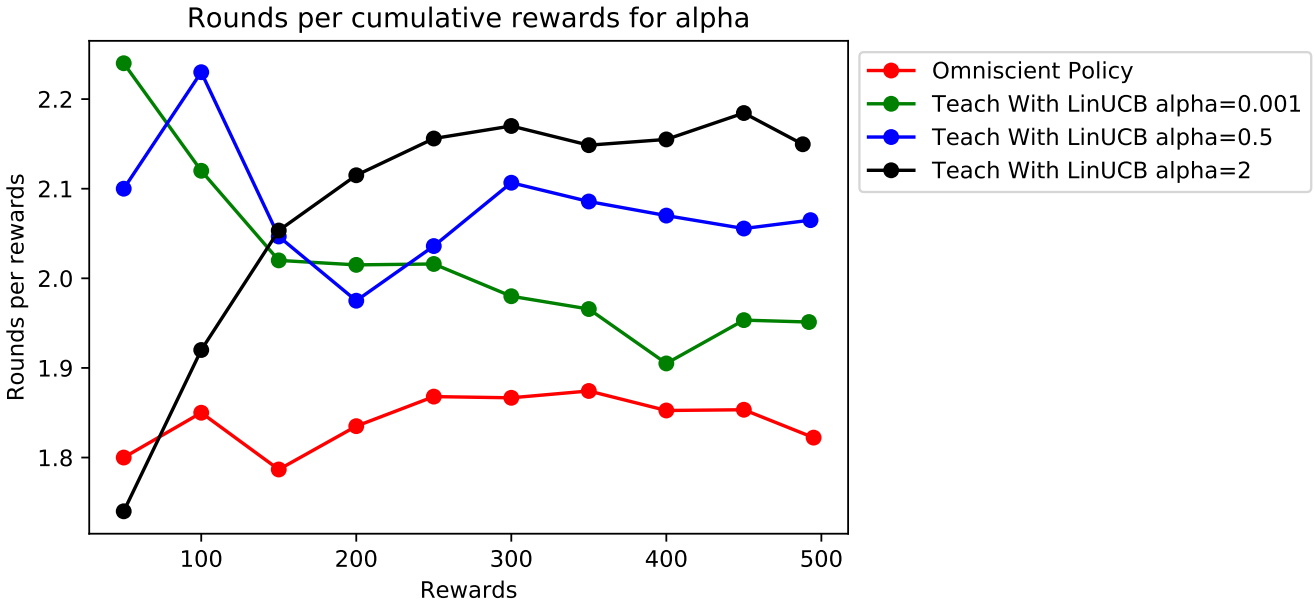

Figure 6.2: Rounds per cumulative rewards ratio for α.

The below table shows the number of content item's presented for different value's of α. As α increases more content item's are presented for student's to learn.

Values of α	**Content Items Explored**
0.001	64
0.5	79
2.0	81

Table 6.1: Content item's explored per α

.

6.2 Confidence Threshold (C)

This is a threshold on the confidence score the skip classifier should exceed for its prediction to be accepted. Skipping is enabled for a topic only after a student gives no reward to a content item. The threshold helps:

- To keep student's engaged by skipping topic's they are unable to understand.
- Give teacher's control on their preference to skipping.

- Allow the learning algorithm to skip content item's that are less likely to give reward's.

We do not want the confidence threshold to be too high as student's might have to go through each content item nor do we want it to be too low such that student's are taken to the next topic on the first occurrence of not understanding a topic. Hence finding an optimal value for the confidence threshold is important to have a good learning experience.

We evaluate the performance for different value's of the confidence threshold over course 1 (Section: 5.1.1). Below are the result's.

6.2.1 Without confidence threshold

We evaluated the skip classifier with no confidence threshold. Below is a table that shows the result's.

Reward's per prediction type (in %).

		Stay (0)	Skip (1)	Total
Reward	0	25.10	18.28	43.38
	1	32.25	24.37	56.62

Table 6.2: Prediction's without confidence threshold

The classifier is evaluated on how well it helps the learning algorithm maximize reward's. This shows us that by **56.62%** it's decision helped increase reward's.

6.2.2 With confidence threshold

We evaluate the skip classifier with confidence threshold. We will only consider data point's where the skip classifier's decision was overruled as its confidence score was below the threshold. This would be when the skip classifier had predicted skipping to the next topic, but since the confidence score was below the threshold, the prediction was ignored. This gives us the true measure of effectiveness for the confidence threshold.

We evaluated the classifier for different value's of confidence threshold. For different threshold value's performance ranged consistently between 56 - 60 %. We found the skip classifier performed most optimally when the confidence threshold is 30. The below table 6.3 shows the result's.

Reward's per prediction type (in %).

Reward	Stay (0)	Skip (1)	Total
0	18.3	24.18	42.48
1	18.3	39.22	57.52

Table 6.3: Prediction's with confidence threshold of 10

The above table shows us that by **57.52%** its decision helped increase reward's. As the value of the confidence threshold was increased the number of skips decreased. Table 6.4 shows the result's for confidence threshold of 30.

Reward's per prediction type (in %).

Reward	Stay (0)	Skip (1)	Total
0	36.82	4.09	40.91
1	50.45	8.64	59.09

Table 6.4: Prediction's with confidence threshold of 30

The above table shows us that by **59.09%** it's decision helped increase reward's.

6.3 Learning Algorithm

We now evaluate the learning algorithm with and without the skip feature.

6.3.1 No Skipping

With skipping disabled the only way a student can move to the next topic is by understanding it or until all content item's have failed to explain the student. This could increase the number of rounds required by a student to complete a course.

The graph (Figure 6.3) shows the cumulative reward of the learning algorithm with respect to the omniscient policy. The reward for the omniscient policy increases linearly, whereas that of the learning algorithm is similar to the optimal policy. This is expected as it does not have optimal arm parameter's pre-configured and learns them in each round.

The omniscient policy required 4410 rounds to get a cumulative reward of 2490. This implies it needs 1.77 rounds for a reward (of 1). The learning algorithm required 4688 rounds to get a reward of 2491. This implies it needs 1.88 rounds for a reward

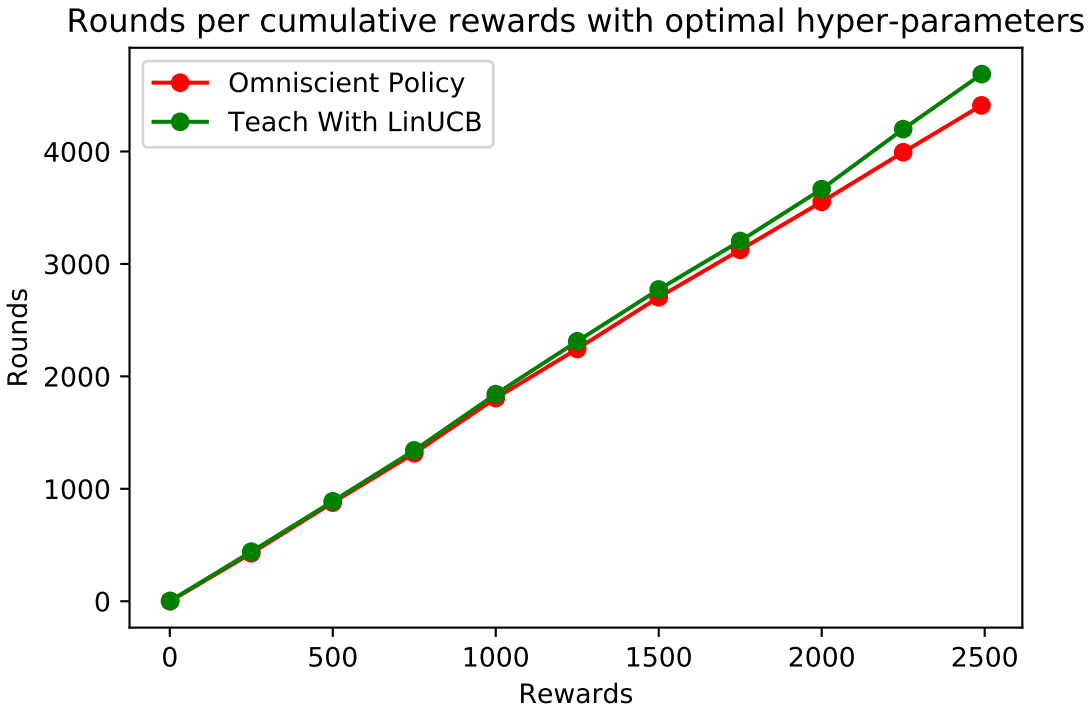

Figure 6.3: Rounds per cumulative reward's without skipping.

(of 1). The graph (Figure 6.4) shows the number of rounds required by the algorithm to obtain rewards at different interval's for optimal value's of hyper-parameters and compares it with the omniscient policy. It shows that our learning algorithm is close to the optimal policy.

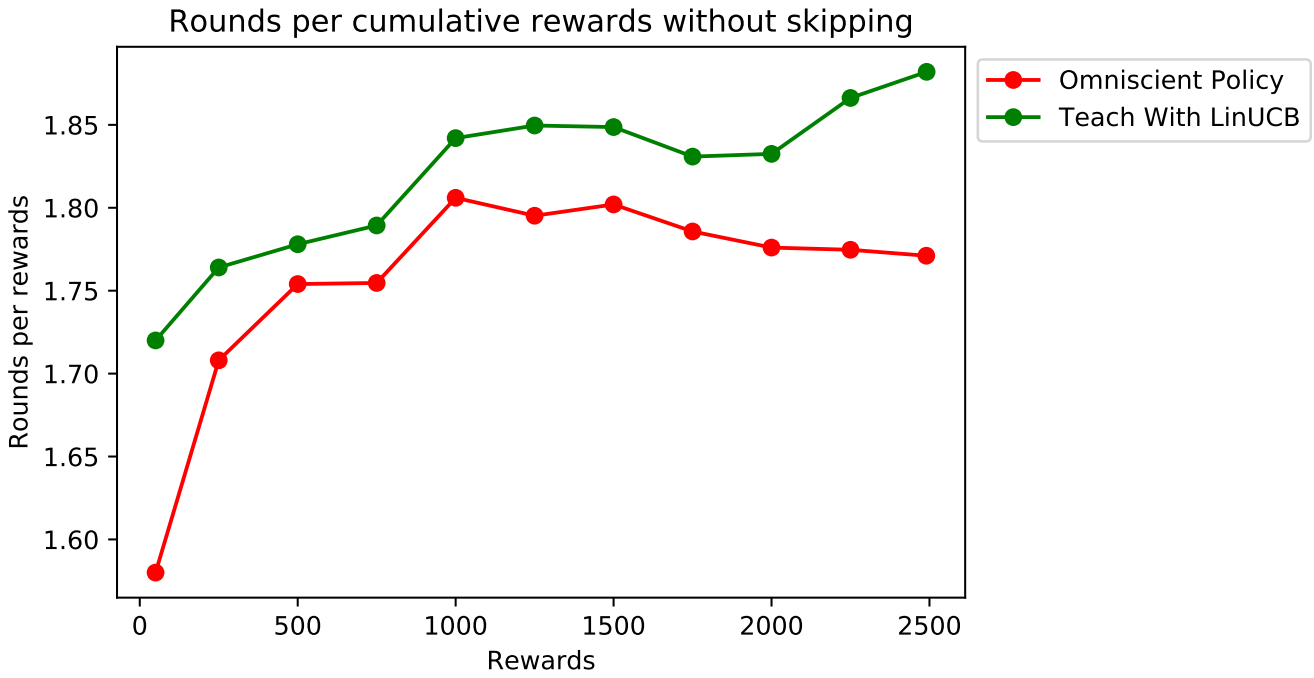

Figure 6.4: Rounds per cumulative reward's ratio without skipping.

6.3.2 With Skipping

If a topic is not understood by a student then skipping is enabled. This does not directly imply the student would be taken to the next topic. For it to happen, the skip classifier should be confident beyond the confidence threshold to predict that it would be better to take the student to the next topic.

Skipping tells the learning algorithm to skip sub-optimal content item's and instead move to content item's that have a higher estimated reward. This ensures that we do not present content item's which are unlikely to help a student understand the topic. The graph (Figure 6.5) shows result's of the learning algorithm with optimal confidence threshold $C = 30$ and $\alpha = 0.001$.

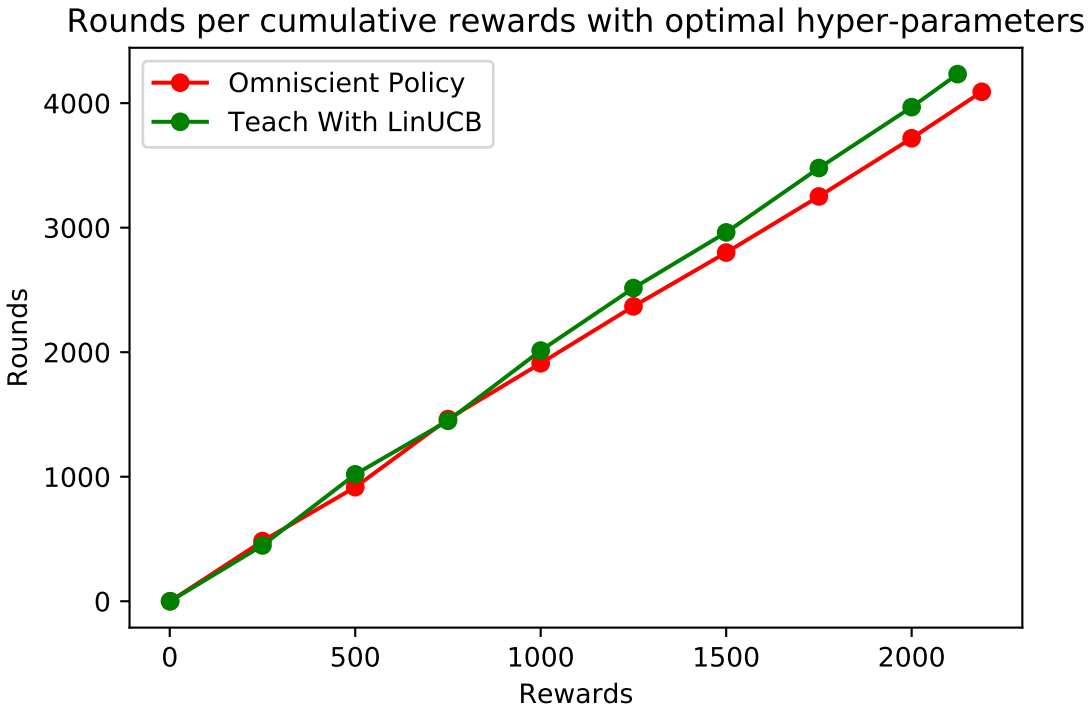

Figure 6.5: Rounds per cumulative reward's with skipping

The graph (Figure 6.5) shows the performance of the learning algorithm with respect to the omniscient policy. The number of rounds and the cumulative reward's reduces with skipping enabled. The cumulative reward reduces for some topic's that the student did not understand the skip classifier predicted with high confidence that it would be better to move to the next topic.

The omniscient policy required 4019 rounds to get a cumulative reward of 2128. This implies it needs 1.89 rounds for a reward (of 1). The learning algorithm required 4185 rounds to get a reward of 2062. This implies it needs 2.03 rounds for a reward (of 1). The graph (Figure 6.6) shows the number of rounds required by the algorithm to obtain rewards at different interval's for optimal value's of hyper-parameters and compares it with the omniscient policy. It shows that even with skipping our learning algorithm is close to the optimal policy.

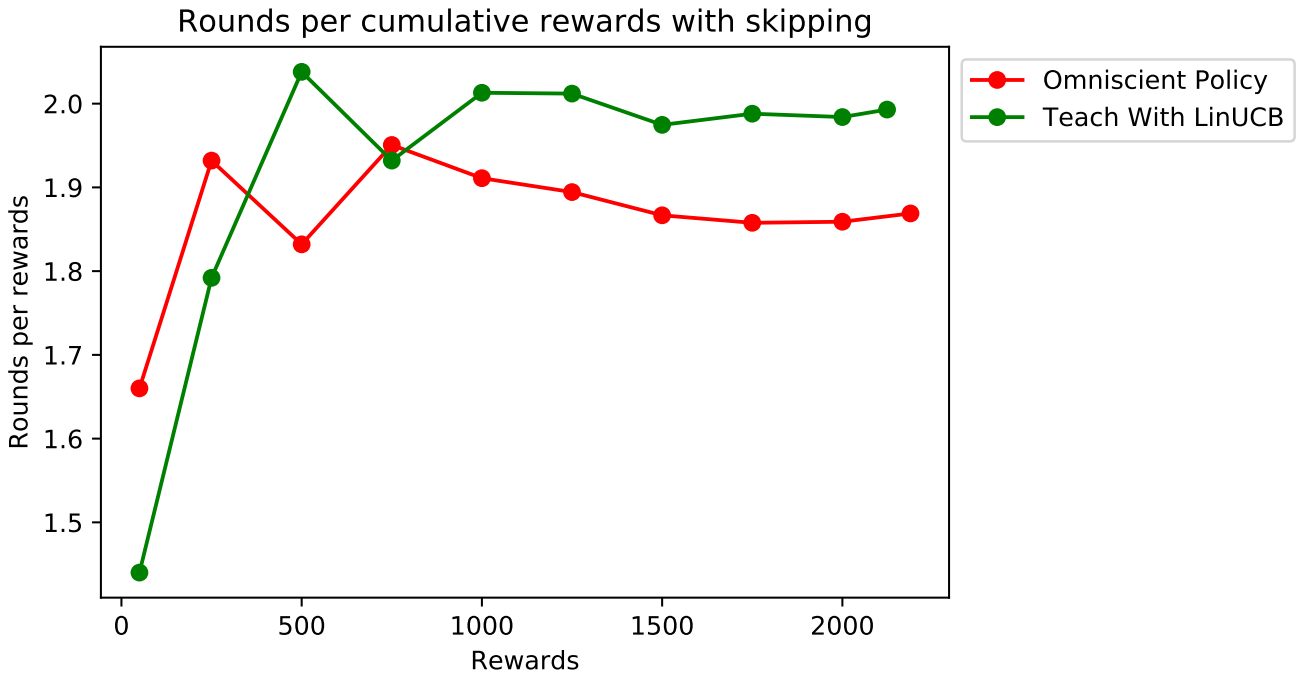

Figure 6.6: Rounds per cumulative reward's ratio without skipping.

Comparing the cumulative reward graph with and without skipping shows us that our learning algorithm performs better without skipping than with skipping. However, without skipping it needs more rounds which could affect student experience.

Chapter 7

Conclusions

This project presents a student-centric approach to teaching. An approach which could make classroom's more interactive by providing a personalized learning experience for student's. We synthesized an unbiased dataset to represent heterogeneous student and content data to evaluate our learning algorithm. Since there were no benchmark's available, we created an omniscient policy which has optimal parameter's pre-configured. The algorithm learns these parameter's to find an optimal content item for each student.

We then present a feature which would be useful when there are several different content item's for a topic to avoid student's from getting frustrated by being unable to understand a topic. This not only helps student's but also helps teacher's recognize topic's student's are less likely to understand. We evaluated the learning algorithm to set a baseline for this new teaching methodology.

Our future work would involve creating an actual course that follows the teaching method's outlined in this project. This would give real-world student data to evaluate the algorithm. We would also like to design other algorithm's to evaluate their performance against our baseline algorithm. An additional optimization would be to find an optimal strategy to introduce skipping such that it does not restrict exploration and still provides a good student experience.

Bibliography

[1] Naoki Abe and Atsuyoshi Nakamura. Learning to optimally schedule internet banner advertisements. In *ICML*, volume 99, pages 12–21, 1999.

[2] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge & Data Engineering*, (6):734–749, 2005.

[3] Deepak Agarwal, Bee-Chung Chen, Pradheep Elango, Nitin Motgi, Seung-Taek Park, Raghu Ramakrishnan, Scott Roy, and Joe Zachariah. Online models for content optimization. In *Advances in Neural Information Processing Systems*, pages 17–24, 2009.

[4] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.

[5] Yoav Bergner, Stefan Droschler, Gerd Kortemeyer, Saif Rayyan, Daniel Seaton, and David E Pritchard. Model-based collaborative filtering analysis of student response data: Machine-learning item response theory. *International Educational Data Mining Society*, 2012.

[6] Peter Brusilovsky and Christoph Peylo. Adaptive and intelligent web-based educational systems. *International Journal of Artificial Intelligence in Education (IJAIED)*, 13:159–172, 2003.

[7] Sébastien Bubeck, Nicolo Cesa-Bianchi, et al. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning*, 5(1):1–122, 2012.

[8] Hao Cen, Kenneth Koedinger, and Brian Junker. Learning factors analysis–a general method for cognitive model evaluation and improvement. In *International Conference on Intelligent Tutoring Systems*, pages 164–175. Springer, 2006.

[9] Ankit Choudhary. Reinforcement learning guide: Solving the multi-armed bandit problem from scratch in python, September 24, 2018.

[10] Benjamin Clement, Didier Roy, Pierre-Yves Oudeyer, and Manuel Lopes. Multi-armed bandits for intelligent tutoring systems. *arXiv preprint arXiv:1310.3174*, 2013.

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

[12] Yue Gong, Joseph E Beck, and Neil T Heffernan. Comparing knowledge tracing and performance factor analysis by using multiple model fitting procedures. In *International conference on intelligent tutoring systems*, pages 35–44. Springer, 2010.

[13] Kristjan Greenewald, Ambuj Tewari, Susan Murphy, and Predag Klasnja. Action centered contextual bandits. In *Advances in neural information processing systems*, pages 5977–5985, 2017.

[14] Kenneth R Koedinger, John R Anderson, William H Hadley, and Mary A Mark. Intelligent tutoring goes to school in the big city. *International Journal of Artificial Intelligence in Education (IJAIED)*, 8:30–43, 1997.

[15] Kenneth R Koedinger, Emma Brunskill, Ryan SJd Baker, Elizabeth A McLaughlin, and John Stamper. New potentials for data-driven intelligent tutoring system development and optimization. *AI Magazine*, 34(3):27–41, 2013.

[16] Andrew S Lan and Richard G Baraniuk. A contextual bandits framework for personalized learning action selection. In *EDM*, pages 424–429, 2016.

[17] Tor Lattimore. The upper confidence bound algorithm, September 18, 2016.

[18] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670. ACM, 2010.

[19] Yun-En Liu, Travis Mandel, Emma Brunskill, and Zoran Popovic. Trading off scientific knowledge and user learning with multi-armed bandits. In *EDM*, pages 161–168, 2014.

[20] Frederic M Lord. *Applications of item response theory to practical testing problems*. Routledge, 2012.

[21] Travis Mandel, Yun-En Liu, Sergey Levine, Emma Brunskill, and Zoran 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.

[22] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[23] Lou Pugliese. Adaptive learning systems: Surviving the storm, October 17, 2016.

[24] Mark D Reckase. Multidimensional item response theory models. In *Multidimensional Item Response Theory*, pages 79–112. Springer, 2009.

[25] Joseph Rollinson and Emma Brunskill. From predictive models to instructional policies. *International Educational Data Mining Society*, 2015.

[26] David Silver. Exploration and exploitation.

[27] Richard S Sutton and Andrew G Barto. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

[28] Ambuj Tewari and Susan A Murphy. From ads to interventions: Contextual bandits in mobile health. In *Mobile Health*, pages 495–517. Springer, 2017.

[29] Kurt Vanlehn, Collin Lynch, Kay Schulze, Joel A Shapiro, Robert Shelby, Linwood Taylor, Don Treacy, Anders Weinstein, and Mary Wintersgill. The andes physics tutoring system: Lessons learned. *International Journal of Artificial Intelligence in Education*, 15(3):147–204, 2005.

[30] Joannes Vermorel and Mehryar Mohri. Multi-armed bandit algorithms and empirical evaluation. In *European conference on machine learning*, pages 437–448. Springer, 2005.

[31] Beverly Park Woolf. *Building intelligent interactive tutors: Student-centered strategies for revolutionizing e-learning*. Morgan Kaufmann, 2010.

[32] Li Zhou. A survey on contextual multi-armed bandits. *arXiv preprint arXiv:1508.03326*, 2015.