# Adaptive Teaching: Learning to Teach

by

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

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 rewards. 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 allows our algorithm to skip a topic 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 rewards 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 development of a personalized learning system began with the creation of an intelligent tutoring system (ITS) [7, 16, 34, 36]. ITSs teach students in different ways based on the pedagogical rules defined for each student. Each student is categorized into groups and taught based on rules defined by the instructor. Such systems would use pre-existing knowledge through assessments, recognize different learning styles, and the track student's progress to behave intelligently.

Each ITS typically comprises of the following components which interact with each other to teach a student. The content component maps concepts to be taught with the prerequisites and dependencies required to understand the concepts. The student model categorizes students into groups to track and categorize each student into groups. The pedagogy component defines instructions to be delivered to a student. This component diagnoses responses to provide assistance to students. Such diagnoses are defined as rules within the system. The pedagogical component presents content based on the state of a student.

There are a few reasons why ITSs are not widely used in the real world. ITSs are primarily rules-based and needs domain experts to manually specify every possibility that a system might face so it could present appropriate learning actions. They are non-adaptive, that is, they provide the same response to a problem independent of a student's past experience with the system. Examples are such system include German Tutor [14] and SQL-Tutor [24]. ITSs assume the student's behavior to be stationary, which is not true in the real world. Managing pedagogical rules, content to deliver and categorizing students is labor-intensive and time-consuming.

In recent years, machine learning has shown that it has the potential to personalize learning and scale for several courses and students. Machine learning based systems

[33] use data to personalize learning actions for each student without the need to explicitly specify learning actions for each individual student. Example of actions could be reading a chapter from a book or article, listening to a podcast, watching a video, or interacting with the system by answering quizzes. These systems are continuously learning from the data generated through students interactions with the system. Thus it has the potential to eliminate the challenges one would face with a traditional ITS [18].

**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 ways to teach a topic, we cannot find the best ways to teach different students. Every student is unique, so we need to use different teaching methods to find a way that is conducive for a student. Once, we have found it we can use this knowledge to teach other similar students effectively. **This is the exploration-exploitation dilemma where there is a trade-off between exploration (exploring non-stationarity in a student's preferences) and exploitation (maximize a student's satisfaction over a period of time)** [3]. For example, an adaptive teaching system should present different explanations knowing a student's preference for learning. However, unless we try different ways 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 terms are used interchangeably.

We represent this use case as a contextual bandit problem. We use contextual information about the student such as their preferences 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 students*. We also use contextual information about the content used to teach a topic, by rating it in terms of *ease of understanding, simplicity, intuitiveness, depth in teaching, conciseness, thoroughness, ratings, abstractness, hands-on, experimental*. A **content item or arms are different actions 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, interactions with a content item, tasks 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 topics. However now instead of a single way to teach everyone, there would be multiple ways to teach. These different ways to teach are called content items. 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 items*(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) [20] to enhance it for our use case. LinUCB was created by Yahoo! research to personalize news recommendation where the user's contextual data like IP address, history of prior visits, location, etc. was used to personalize news recommendation. The feedback was measured in terms of click-through-rate (CTR). When a user clicks a news recommendation the learning algorithm receives a reward, or else they receive no reward. LinUCB increased CTR over Yahoo! Front Page Today Module dataset by 12.5% [20].

In some aspects, our use case is different from the news recommendation. Firstly, they have to recommend news from the vast pool of news articles. However, we represent adaptive teaching in a way that the learning algorithm has to select from content items relevant to the topic. Such a representation helps the algorithm learn quickly, which is necessary for adaptive teaching to keep students engaged with a system that understands their needs. News recommendation is a feature presented along with a host of other features. They can explore much longer. However, exploring excessively in adaptive teaching use case could leave students disillusioned with the system.

There are similarities between them. Both systems suffer from a cold-start situation and non-stationarity of user behavior. Cold start arises because a significant number of users are likely to be new with no historical information. Both use cases suffer from non-stationarity in user's behaviour as preferences change with time. Hence, balancing exploration (to find content that matches the user's interest) and exploitation (present content that interest users) becomes critical for user satisfaction.

## 1.2 Motivation

The main problem with traditional education, which has been perpetual, is the enormous challenge teachers face for being responsible to ensure every student is able to acquire expertise in their subject even though students may come from diverse backgrounds and interests [27]. In such classrooms learning has largely remained a one-size-fits-all experience in which the teacher selects a learning resource for all students in their class regardless of their diversity in needs, understanding, ability, preferred learning style, and prior knowledge. It is not feasible for teachers to ensure their explanations can cater to all students. Hence there is a need for a system which could personalize teaching for students to help them learn effectively as well as increase course engagement and progression.

Such systems would be adaptive, recognize different levels of prior knowledge among students, as well as course progression based on a student's skill and feedback from learning. This could change teachers responsibility from a provider to a remediator and facilitator in teaching. These would adapt to individual student's learning patterns instead of a student having to adjust to the way of teaching. They would provide timely and comprehensive data-driven feedback to recognize potential challenges that students might come across as the course progresses.

## 1.3 Contribution

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

We also provide a skip feature which is meant to keep a student engaged to increase their retention as well as provide feedback to teachers by recognizing the challenges faced by a student early in the course. Our online learning algorithm gives close to optimal results 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 concepts 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 along with the dataset synthesized to evaluate our algorithm. It also explains the evaluation strategy followed to examine our results. Chapter 6 presents the results of our experiments and compares our learning algorithm with respect to the best possible policy. Chapter 7 concludes this project by summarizing the contributions and outlines possible avenues for future work.

# Chapter 2

# Preliminaries

This chapter briefly explains the key concepts used in this project.

## 2.1 Multi-armed bandit

Multi-armed bandit is a problem setting where an agent makes a sequence of decisions in time $1, 2, ..., T$. At each time $t$ the agent is given a set of $K$ arms to choose and has to pull an arm. As a feedback for pulling an arm, it receives a reward for that arm, while the rewards of other arms cannot be determined. Problems represented as multi-armed bandits are stateless. They can be stochastic or adversarial. In a stochastic environment the reward of an arm is sampled from some unknown distribution, and in an adversarial setting the reward of an arm is picked by an adversary and is may be sampled from a distribution [37]. In this project, we assume the problem setting as stochastic.

Personalized recommender systems recommend items (e.g., movies, news articles, web advertisements) to their users based on their predicted preference for these items. The feedback received from users response helps the system improve their prediction [2]. However, to improve the quality of recommendations an item has to first be recommended. If an item is never recommended to users, the system cannot evaluate the feedback to improve it's quality of predictions on these items. Such problems where there is some information about the users which can be used to improve the quality of predictions, can be represented as a contextual bandit problem [35].

## 2.2 Contextual Bandit

In the theory of sequential decision-making, contextual bandit problems [32] are placed between multi-armed bandit problems [8] and full-blown reinforcement learning (usually modeled using Markov decision processes with discounted or average reward to take optimal actions) [30]. Traditional bandit algorithms do not use any side-information or context. However, contextual bandit algorithms use context to learn and map contexts to appropriate actions. Since bandit algorithms only have a single state they do not have to consider the impact of their actions over future states. Nevertheless, in many practical domains, such a problem setting is useful. This is true when the learner's action have limited impact on future contexts. In such a problem setting contextual bandit algorithms have shown great promise. Examples include web advertising [1] and personalized news article recommendation on web portals [20, 13].

Formally a contextual bandit problem is a repeated interaction which takes place over $T$ rounds. At each round $t = 1, 2, ...T$ the environment reveals contexts $x_t \in X$ about the user and the available actions which are 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 an action 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 items together with their feature vectors $x_c$ for $a \in A_t$. $X_t$ encapsulates $x_s$ and the context $x_c$ of all content items available in round $t$.

2. Based on observed rewards 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 arms $a \neq a_t$. This is valid for similar contextual bandit settings. [20].

## 2.3 Upper Confidence Bound (UCB)

An unavoidable 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 algorithms which try to find the best trade-off between exploration and exploitation. It is based on the principle of *being optimistic* by choosing actions 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 well justified, in which case the learner is already 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 [19]. UCB algorithms 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 arms $a_1, a_2, a_3$. The reward distribution for each arm after several rounds is a Gaussian distribution $Q$ with mean $\mu$ and standard deviation $\sigma$. The y-axis is the probability of obtaining a certain reward for these arms.

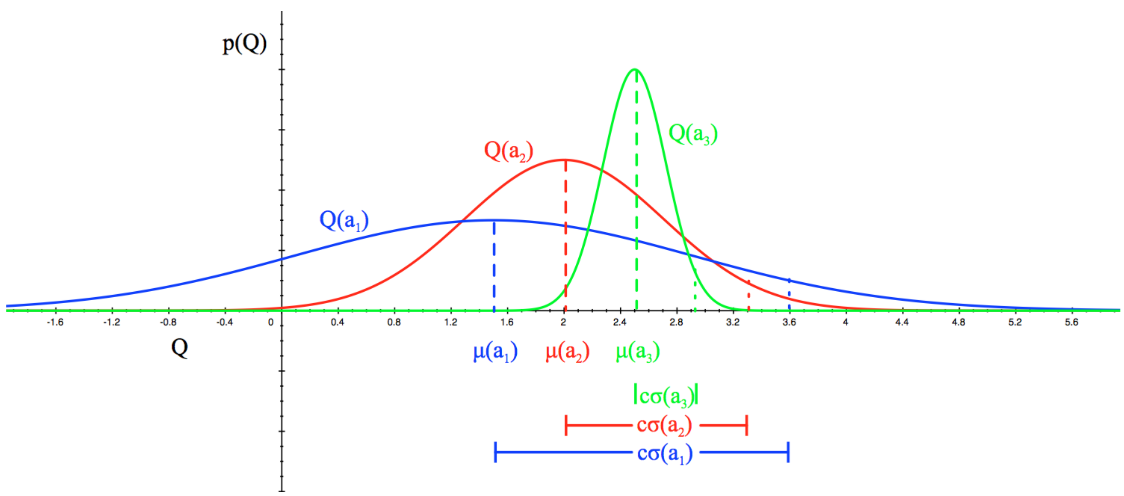


Figure 2.1: An example: UCB
[29]

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 bandit setting where the UCB of each arm is computed efficiently by assuming the reward is linear, given as $E[r_{t,a}|\mathbf{x_{t,a}}] = x_{t,a}^T \boldsymbol{\theta_{t,a}}$. 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 $\alpha$, which allows us to control exploration over arms. This is achieved by scaling the upper confidence bound by $\alpha$. A higher value of $\alpha$ 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** $p_{t,a} = \widehat{\theta}_a^T x_{t,a} + \alpha\sqrt{x_{t,a}^T A_a^{-1} x_{t,a}}$ [20].

# Chapter 3

# Related Work

Our use case could also be represented using a partially observed Markov decision process (POMDP) framework. POMDPs model the student's latent knowledge states and their transitions to learn a policy that will present an action that could maximize reward received over the long run (long-term learning outcome). Previous work applying POMDPs to personalized learning has had limited success. However, to create a personalized learning schedule using a POMDP can get complicated and intractable as the number of dimensions representing the states and actions grows. As a consequence of this curse of dimensionality, POMDPs have had a limited impact to personalize learning in large-scale applications which has a large number of students and learning actions [18].

Reinforcement learning with Markov decision processes was used to define the learning path for each student. This represents the knowledge states a student can transition to [15]. At each state, a topic is presented to a student, and their knowledge is tested. The feedback received is used to update the usefulness of executing an action before the student is transitioned to the next state. A similar approach was applied in another system named ADVISOR [5]. However, it could not scale with the number of features. As the number of states increases, it becomes difficult for such a system to learn.

A more practical and tractable approach to personalized learning is to learn a policy, which maps contexts to actions using the multi-armed bandit (MAB) framework, which is more suitable for our use case. This makes it more practical than the MDP framework in large-scale educational applications [18].

eTutor [31] developed an online learning algorithm for personalized education. It uses the context of a student to decide the content item to present. It learns the sequence to present content items to maximize the final score and reduce the time to teach. This system is targeted to those who would like to refresh their knowledge. Our goal is to teach students who have little knowledge of the subject.

Multi-armed bandits have also been used to recommend courses to learners and assess their knowledge [25]. The works in [10, 17] both use expert knowledge to learn a teaching policy. Similar tasks are studies in [10], which uses domain expertise to reduce the set of possible actions a student can take. It is focused on assessments to test a students knowledge, using psychometrics paradigms such as Item Response theory and Zone of Proximal Development. Item response theory decides the content item to present to a student based on their response to previous 'n' content items. Zone of proximal development defines the content that should be presented to enhance a student's knowledge. It presents a content item that is just hard enough to challenge a student to keep them engaged. These studies do not focus on providing a adaptive learning path to navigate students to learn and understand new concepts. Our focus is on adaptive teaching, rather than adaptive testing.

The work in [21] applies a MAB algorithm to educational games to find a trade-off between exploring learning resources to accurately estimate arm means, while also trying to maximize users test performance. Their approach is context-free and does not consider diversity among individual users. The work in [23] collects data to find how students interact with the system to extract features as they play an educational game. It uses this knowledge to find a good teaching policy [18].

The work in [18] is focused on adaptive testing to assess students performance. They use contextual MAB to find questions to assess a student. The question depends on a student's response to earlier questions. At each round, they have all questions to assess a student. Contrary to that we only have a restricted set of content items available at each round. Our use case is focused on adaptive teaching to enable students to learn.

Other works typically create a model for each component, namely student, knowledge, domain and use knowledge tracing [11], item response theory and zone of proximal development [22, 28, 6] to make better decisions. These different methods have similar predictive performance. However, they could have very different teaching policies. [18]. While these results are different approaches to make the best prediction, none of them use machine learning to develop a policy learning algorithm.

# 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 algorithms [4] (Section 2.3). These algorithms 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 arms parameters iteratively after each pull to make a 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 algorithms upwardly biased estimate of the expected reward, where the bias is due to the algorithms 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 notations used in the algorithm.

| Symbol | Meaning |
| --- | --- |
| $\alpha$ | Parameter to scale Confidence bound. |
| $C$ | Confidence threshold to skip. |
| $\mathbf{x_s}$ | Student context vector. |
| $x_c$ | Content items 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 contexts for topic $i+1$. |
| $a$ | An arm $a$ for topic $i$. |
| $a'$ | An arm $a'$ for topic $i+1$. |
| $A_t$ | Arms available at round $t$. |
| $A_{t'}^{i+1}$ | Arms 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 notations

**Note**

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

**Algorithm 1** Teach with LinUCB

**Hyper Parameters :** $\alpha \in \mathbb{R}_+$

$C$ : Confidence threshold to skip

**Inputs :** Student context $\mathbf{x_s}$ and content context $x_c$ of available arm $a \in A_t$ 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 functions used along with skipping.

The algorithm requires two hyper-parameters to be configured. The first one is $\alpha$ which scales 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 students who are unlikely to learn from content items available for a topic. It is meant to streamline learning. This could also be used by teachers to recognize topics 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 items context $x_c$ for the topic $i$ being taught. With the context $X_t$ and arms $A_t$, 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 parameters $A_{a_t}$ , $\mathbf{b_{a_t}}$ of the arm pulled. We then use this reward $r_t$ to train the skip classifier to make better predictions in upcoming rounds. The features for the classifier comprise of a 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 $i+1$. This completes the first half. The second half explains the functions 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 arms $A^{i+1}_{t'}$ 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** $a_t, argmax p_{t,a},$
**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 items $A_{t'}^{i+1}$ and context data $x_c^{i+1}$ associated those content items. After combining the contexts 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 arms 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 arms $A_t$ 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 $\alpha$. 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 predictions. 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 sets 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 experiments. The next section explains how we evaluate our algorithm (Section 5.3) in absence of pre-existing benchmarks using an omniscient policy (Section 5.4). This is followed by sections which explain how the learning algorithm (Section 5.5) and the skip feature (Section 5.6) work in these experiments.

## 5.1 Dataset

Machine learning algorithms are data-driven. Due to the novelty of our approach to the best of our knowledge, there is no similar dataset available. Hence **we synthesize datasets to represent data generated by students taking courses in an adaptive teaching environment.**

An honest attempt is made to synthesize an unbiased dataset representative of the heterogeneous students and content items. Biased datasets tend to focus on targeted student groups (for instance, having many students who give positive feedback). This could result in higher rewards. 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 preferences and content features.

### 5.1.1 Course

We use the following courses for our experiments.

1. *Course 1 :* A course which comprises of 10 topics. It is taken by 50 students. There are a total of 119 content items for 10 topics. So on an average, there are 12 content items per topic. We use this course to find optimal hyper-parameters ($\alpha$ and $C$) for our learning algorithm.

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

3. *Course 3 :* A course which has 100 topics. It is taken by 400 students. There are 720 content items for 100 topics. So on an average, there are 7 content items per topic. We use this course to evaluate our algorithm's scalability.

### 5.1.2 Context

Information about students preferences could be collected through scientific experiments in which students are explained topics in different ways. These different ways represent diverse teaching methods and strategies used to recognize how students learn. Students are then assessed to evaluate their understanding. Such an experiment would help us better understand different learning preferences and evaluate features of teaching material which helped them learn effectively.

Research has shown that students prefer to learn in various ways [12]. Though there is no unanimous consensus, there is a fair bit of research and understanding to support the different needs of a student. These needs are debatable with different opinions published targeting students from different streams, such as medical, law, management students. Generalizing features of teaching material, methods and strategies is a different field of research. The features we consider are by no means exhaustive but a representative subset of the main features. The tables 5.1 and 5.2 describe the student and content context used for these experiments.

We will assume there was a survey conducted among students 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. Although some features are correlated, to avoid being biased I did not consider these correlations.

More importantly, these correlations are subjective without strong scientific backing. The extent of correlations depends on the dataset. Correlations would simplify the dataset by reducing dimensionality and help the algorithm learn quickly. Without correlations, we got to evaluate the algorithm under the worst-case assumptions.

These preferences were normalized between 0 to 1.

| **Student Context** | **Description** |
|---|---|
| Visual (S_V) | How much preference is given to visual explanations (video, short-film, movie-clip, video blog's)? |
| Text (S_T) | How much preference is given to written explanations (books, articles, blogs, research papers)? |
| Demo-based (S_D) | How much preference is given to live experiments 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 students 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 items which are interactive and require students 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) | Students self-evaluate their readiness, motivation, excitement for the course. |
| Pre-assessment (S_PA) | Teachers 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($S\_D$) methods 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($S\_S$), lectures($S\_L$) , audios($S\_A$) to learn 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 experiments to help understand a concept? |
| Brief/Concise (C_C) | Is it short, to the point or descriptive, verbose and elaborative, keeping in mind that learners 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 students 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 terms 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 experiments, we consider a typical course which comprises of topics to be taught. These topics 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 items. 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. Topics 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 students with the omniscient policy and the learning algorithm deciding the content item to be presented for each student. It is an environment where several students 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 parameters in each round to give better predictions.

## 5.3 Evaluation Strategy

Since there are no readily available benchmarks to compare our algorithm, we assume there exist an omniscient policy. This policy has optimal parameters 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 a 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 parameters $\theta^*_{t,a}$ 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 parameters are updated to make better decisions in the upcoming rounds. This experiment aims to find how well does our algorithm optimize an arm's parameters to match the omniscient policy.

## 5.4 Omniscient Policy

This policy knows all the probability distributions. At every step of makes the way the best decision as it knows the true distributions. It does not have to learn anything. It has optimal parameters $\theta^*$ 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 students 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 items. It checks 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 rewards. 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

The learning algorithm checks with skip topic feature to decide whether or not the content item should be presented for the current topic. Skip topic predicts this by using a student's 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 students to recognize common topics and content items that students find difficult so it could make a confident decision.

The aim to create skip topic feature is to streamline learning for a student. 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 parameters towards the zero vector using squared Euclidean norm [26].

# 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 values for hyper-parameters $\alpha$ (Section 6.1) and confidence threshold $C$ (Section 6.2). We then proceed to use these optimal values to evaluate the learning algorithm with and without the skip feature. (Section 6.3).

## 6.1 Confidence Bound $\alpha$

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

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

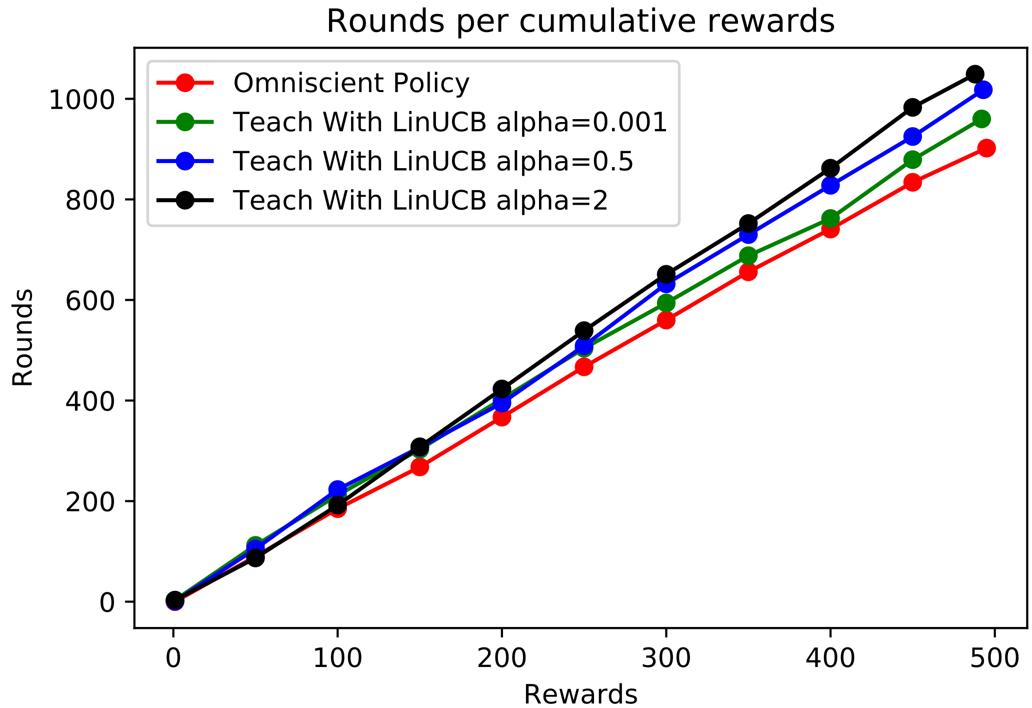


Figure 6.1: Rounds per cumulative reward for $\alpha$.

The graph (Figure 6.1) compares the omniscient policy with the learning algorithm for different values of $\alpha$. It shows that the learning algorithm **took 960 rounds to maximize reward when** $\alpha = \mathbf{0.001}$ compared to 1018 rounds required by $\alpha = 0.5$ and 1049 rounds required by $\alpha = 2$. On repeated run of the same experiment, we found that a value of $\alpha$ between 0 to 0.5 gives better results.**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 ratio at different intervals for different values of $\alpha$ and compares it to the omniscient policy. It clearly shows that $\alpha = 0.001$ required the fewest rounds to maximize reward.

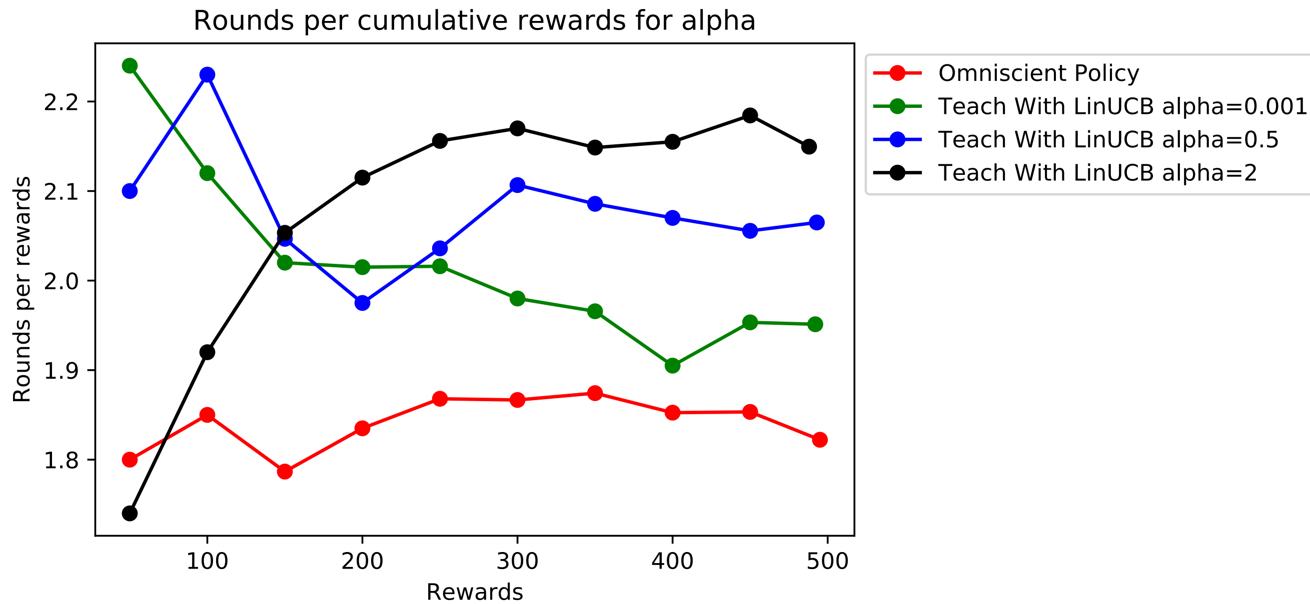


Figure 6.2: Rounds per cumulative reward ratio for $\alpha$.

The below table shows the number of content items presented for different values of $\alpha$. As $\alpha$ increases more content items are presented for a student to learn.

| **Values of $\alpha$** | **Content Items Explored** |
|---|---|
| 0.001 | 64 |
| 0.5 | 79 |
| 2.0 | 81 |

Table 6.1: Content items explored per $\alpha$

.

## 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 a student engaged by skipping topics they are unable to understand.
- Give teachers control on their preference to skipping.

- Allow the learning algorithm to skip content items that are less likely to give rewards.

We do not want the confidence threshold to be too high as students might have to go through each content item nor do we want it to be too low such that students 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 values of the confidence threshold over course 1 (Section: 5.1.1). Below are the results.

### 6.2.1 Without confidence threshold

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

Reward 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: Predictions without confidence threshold

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

### 6.2.2 With confidence threshold

We evaluate the skip classifier with confidence threshold. We will only consider data points where the classifier's decision was overruled as its confidence score was below the threshold. This would be when the 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 values of confidence threshold. For different threshold values 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 results.

Reward per prediction type (in %).

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

Table 6.3: Predictions with confidence threshold of 10

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

Reward per prediction type (in %).

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

Table 6.4: Predictions with confidence threshold of 30

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

## 6.3 Learning Algorithm

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

### 6.3.1 Without Skipping

With skipping disabled the only way a student can move to the next topic is by understanding it or until all content items 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 parameters 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

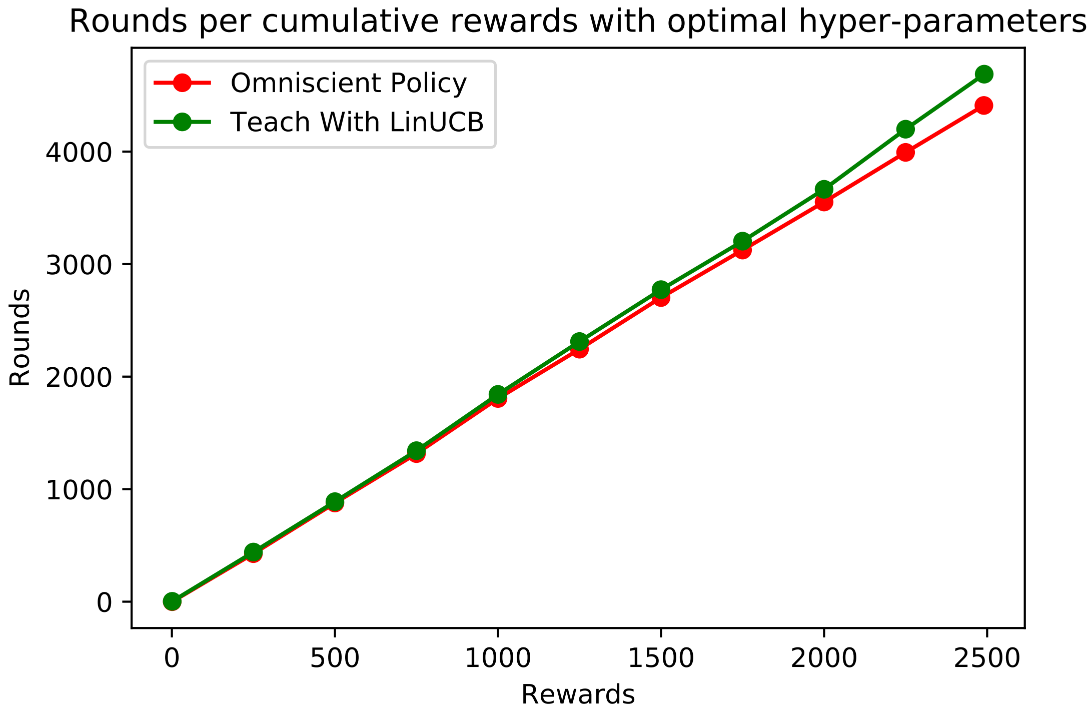


Figure 6.3: Rounds per cumulative reward without skipping.

(of 1). The graph (Figure 6.4) shows the number of rounds per reward required by the algorithm at different intervals with optimal values of hyper-parameters and compares it to the omniscient policy. It shows that our learning algorithm is close to the optimal policy.

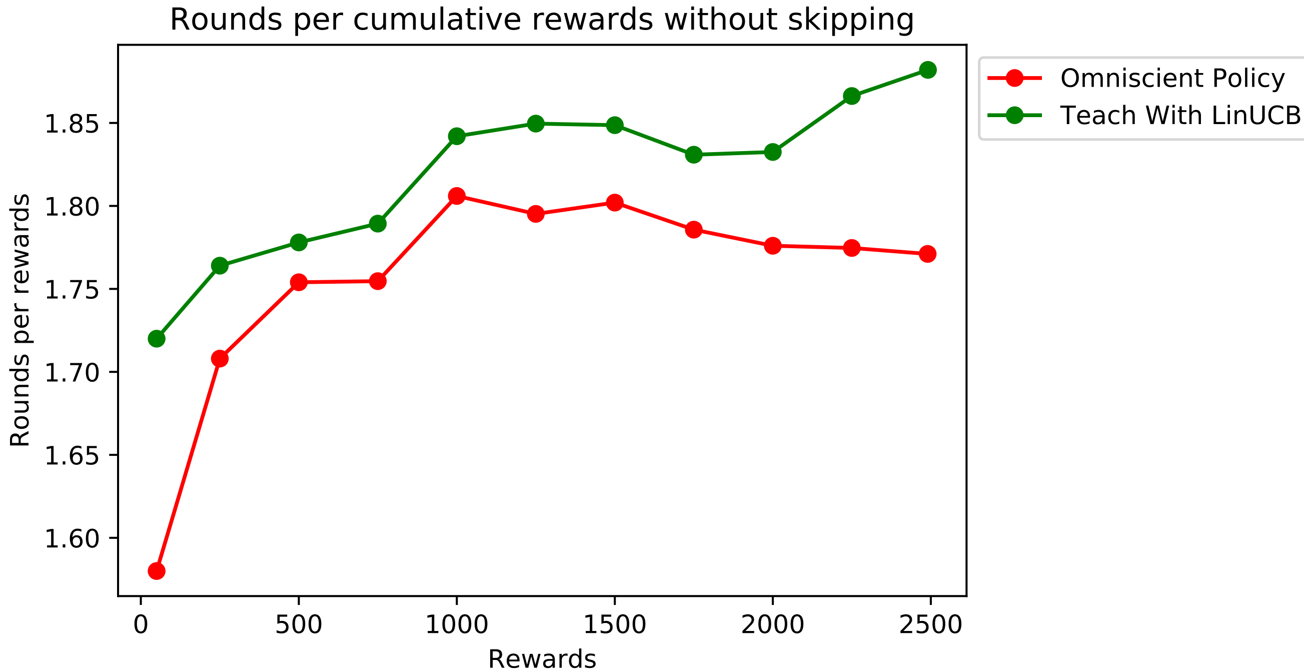


Figure 6.4: Rounds per cumulative reward ratio without skipping.

### 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 items and instead move to content items that have a higher estimated reward. This ensures that we do not present content items which are unlikely to help a student understand the topic. The graph (Figure 6.5) shows results of the learning algorithm with optimal confidence threshold $C = 30$ and $\alpha = 0.001$.

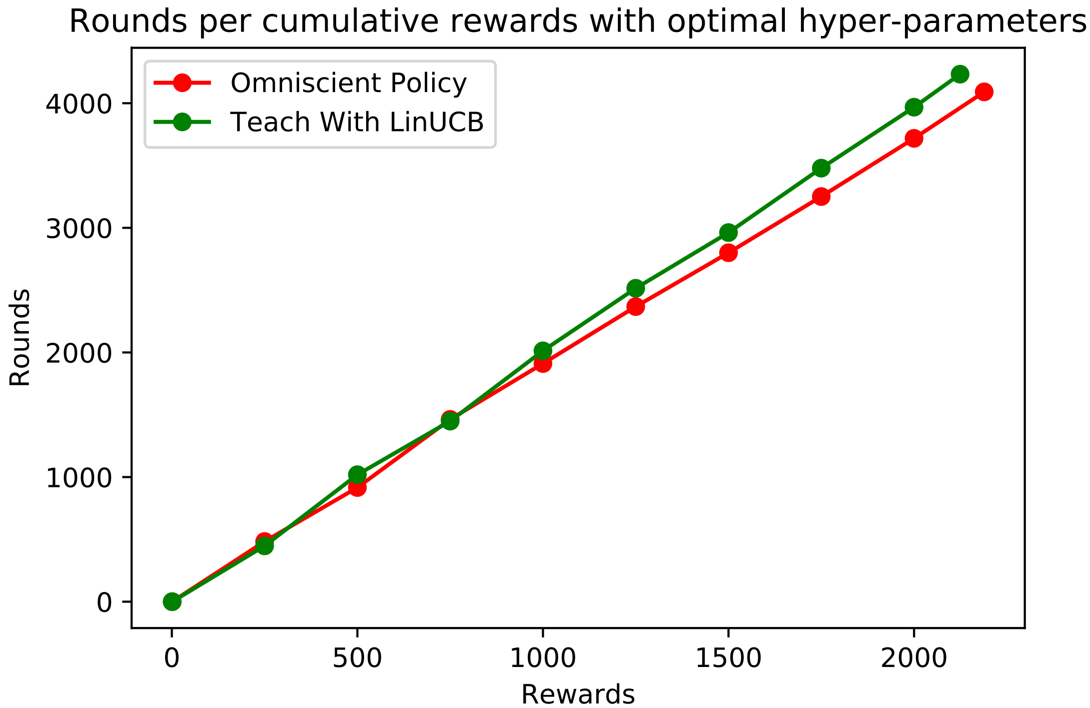


Figure 6.5: Rounds per cumulative reward with skipping

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 reduces with skipping enabled. The cumulative reward reduces as for topics that a 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 per reward required by the algorithm at different intervals for optimal values of hyper-parameters and compares it to the omniscient policy. It shows that even with skipping our learning algorithm is close to the optimal policy.

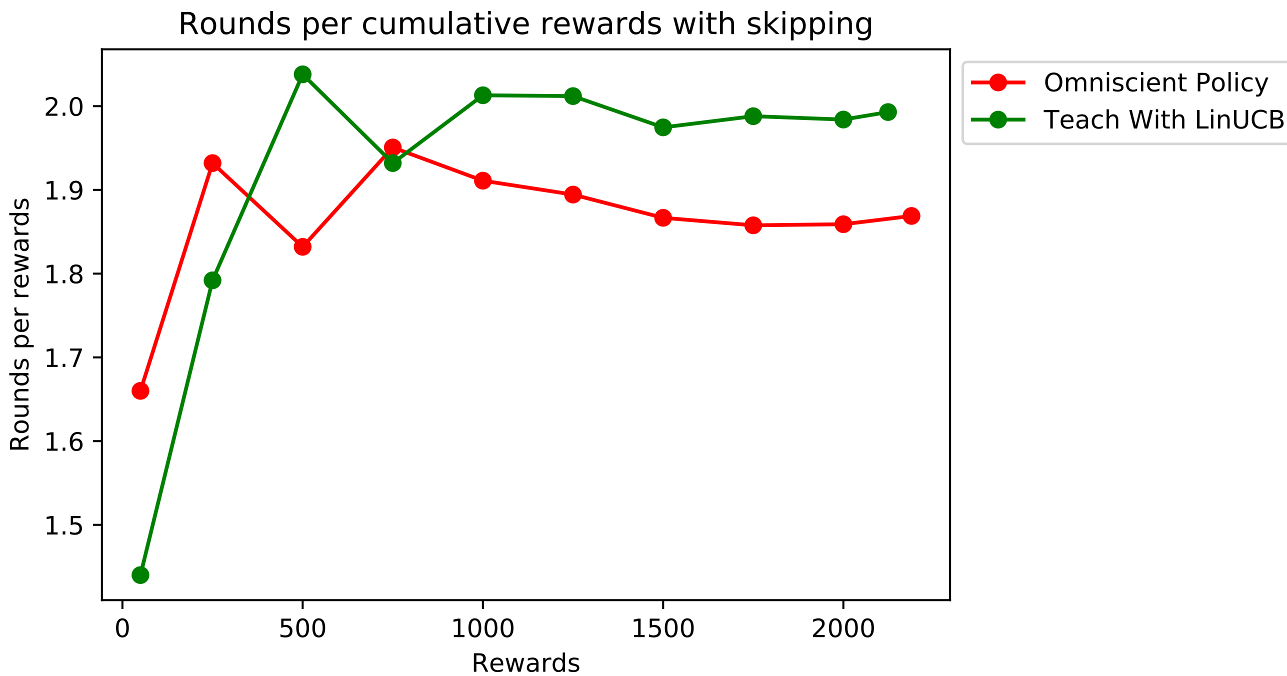


Figure 6.6: Rounds per cumulative reward ratio with skipping.

To evaluate the scalability of our learning algorithm we ran a simulation over course 3 (Section 5.1.1). We found that the omniscient policy required 71,449 rounds to get a cumulative reward of 38,891. This implies it needs 1.84 rounds for a reward (of 1). The learning algorithm required 77,179 rounds to get a reward of 37,350. This implies it needs 2.07 rounds for a reward (of 1). The graph (Figure 6.7) shows the number of rounds per reward required by the algorithm at different reward intervals and compares it to the omniscient policy. It shows that our learning algorithm scales over a larger course.

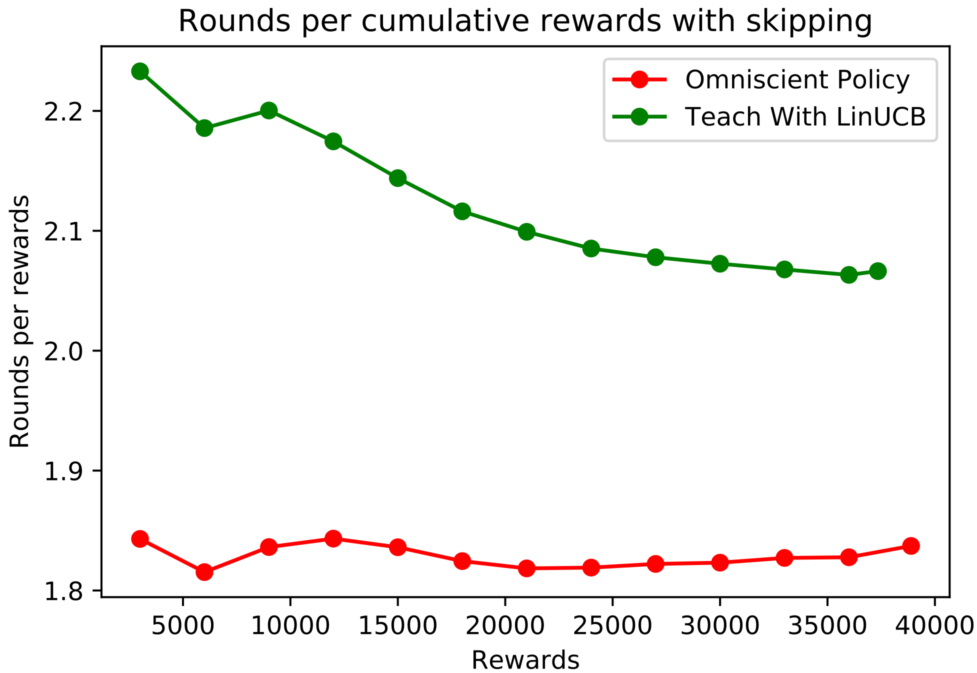


Figure 6.7: Scalability over larger courses.

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 a classroom more interactive by providing a personalized learning experience for students. We synthesized an unbiased dataset to represent heterogeneous student and content data to evaluate our learning algorithm. Since there is no benchmark available, we created an omniscient policy which has optimal parameters preconfigured. The algorithm learns these parameters to find an optimal content item for each student.

We then present a feature which would be useful when there are several different content items for a topic to avoid a student from getting frustrated by being unable to understand a topic. This not only helps students but also helps teachers recognize topics students 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 methods outlined in this project. This would give real-world student data to evaluate the algorithm. We would also like to design other algorithms 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] Joseph Beck, Beverly Park Woolf, and Carole R Beal. Advisor: A machine learning architecture for intelligent tutor construction. *AAAI/IAAI*, 2000(552-557):1–2, 2000.

[6] 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.

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

[8] 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.

[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] Richard M Felder. Matters of style. *ASEE prism*, 6(4):18–23, 1996.

[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] Trude Heift and Devlan Nicholson. Web delivery of adaptive and interactive language tutoring. *International Journal of Artificial Intelligence in Education*, 12(4):310–325, 2001.

[15] Ana Iglesias, Paloma Martínez, Ricardo Aler, and Fernando Fernández. Learning teaching strategies in an adaptive and intelligent educational system through reinforcement learning. *Applied Intelligence*, 31(1):89–106, 2009.

[16] 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.

[17] 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.

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

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

[20] 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.

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

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

[23] 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.

[24] Antonija Mitrovic. An intelligent sql tutor on the web. *International Journal of Artificial Intelligence in Education*, 13(2-4):173–197, 2003.

[25] Minh-Quan Nguyen and Paul Bourgine. Multi-armed bandit problem and its applications in intelligent tutoring systems. *Master's thesis. École Polytechnique*, 2014.

[26] 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.

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

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

[29] David Silver. Exploration and exploitation.

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

[31] Cem Tekin, Jonas Braun, and Mihaela van der Schaar. etutor: Online learning for personalized education. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 5545–5549. IEEE, 2015.

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

[33] Rice University. Openstax tutor, 2016.

[34] 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.

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

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

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