Can Q-Learning Learn You-Learning? Reinforcement Learning for Adaptive Quizzing

Nathanael Cadicamo
Stanford University Department of Computer Science
`cadicamo@stanford.edu`

Abstract

US student reading and math scores have fallen to a multi-decade low, motivating an exploration into supplemental educational technology. In the domain of online learning, quizzes are often served to both evaluate and stimulate student learning. The present paper investigates how best to sequentially serve the types of questions in a quiz to optimize student learning. Specifically, I model the quiz environment as a Markov Decision Process with states representing the domain and quiz history of a given student, actions representing the type of question to serve, and rewards a function of the student history, question difficulty, and correctness of student response. I use Q-Learning to find an optimal policy for sequential question serving and find a 14% increase in expected reward over a random baseline. This finding serves as a foundation for further research into the application of ML in education technology.

1 Introduction

According to the National Assessment of Educational Progress, reading and math scores in the United States have dropped to their lowest point in decades [2]. Though it is beyond the scope of the current paper to investigate why this is the case, I speculate that this is in part due to the rise in technology access and declining attention spans. The flip side of this increase in technology is that educational content, often in the form of online learning platforms, is also more accessible than ever. One thread of inquiry relevant to this domain is the enhancement of online learning platform features, such as quizzes.

Online quizzes in the evaluative context have long been used to simply measure the knowledge level of a student. Beyond evaluation, numerous applications, like the language-learning platform Duolingo, also use quizzes in a game-like setting which is designed not purely to evaluate but also to stimulate and encourage student learning. Platforms which use these quizzes are well-advised to adapt their question sequences to the status of the student learner so as to maximize their engagement and learning; outside of the standardized testing setting, it is not obvious that precisely the same question sequence should be served to each and every student, and it is rather likely that adaptive quizzing increases student learning. The relevant question then becomes how to best model a sequence of questions in relation to a student, and how to optimally choose the next question given their quiz history.

The present paper therefore investigates this question, modeling the quiz environment as a Markov Decision Process and applying Q-Learning to identify an optimal policy for sequential question recommendation.

2 Related Work

There is a rich body of literature to suggest that "RL in education allows for personalized and adaptive learning," and as such, this paper will build upon this literature by aiming to find a policy to identify the next-best-action in a quiz environment to maximize student learning [5].

A 2022 paper investigates intelligent quiz question recommendation with a two-sided recommendation system [4]. This paper models the state space as the set of all possible question sequences, which differs slightly from our model of state space. It also uses a reward function that is dependent on user feedback beyond the score of the question, including a like/dislike feature present in their question data. As such, their reward function does not explicitly take into account the progress or quiz history of a given student, nor does it explicitly factor in the difficulty of the question, which the present paper does attempt to do. The fact that this paper finds a slight increase over random in cumulative reward suggests that reinforcement learning in this setting is likely sound.

A 2020 paper implements Q-Learning specifically to learn an optimal policy over a small state space representative of a learning curriculum, where actions correspond to moving between states representative of levels within the curriculum [1]. This paper finds Q-Learning to effectively increase reward as episodes increase, but does so in a relatively simplistic fashion that does not account for real-world quiz history data.

This research sets the stage for the present paper, which contributes novel research by using a more robust and detailed reward function and significantly larger state space while retaining the Q-Learning algorithm as the core policy-learning model.

3 Methodology

3.1 Data

This project makes use of the large and highly detailed Riiid dataset. This dataset contains the quiz history of over a million students on the TOEIC test. In our `train.csv`, each entry is an event, most of which are question events, that contain features including but not limited to `content_type_id` which indicates whether it is a quiz question or lecture event, `content_id`, which is a foreign key to the question ID in the `questions.csv` if the type is in fact a question, `anwered_correctly`, which is a boolean indicator for whether or not the student answered correctly, and `user_id`, which indicates the student at this event. The `questions.csv` also contains a feature `part`, which indicates which section of the test that question is from. This feature approximates the knowledge domain or subject of the question.

3.2 Modeling

A Markov Decision Process involves a state space, an action space, a reward function, and transitions. I choose to represent the state space as a proxy for student knowledge. Specifically, I construct tuples of the form [`part`, `num_correct_so_far`, `num_incorrect_so_far`] that represent, at any given question event, the approximate knowledge level of the given student in terms of their quiz history in this part of the test. This representation allows for unambiguous extraction of states from the dataset, and transitions between states are implicit in the progress throughout the dataset of question event t to question event $t + 1$, resetting each time there is a new part encountered or a new user.

For our downstream goal of learning optimal sequential question recommendations, the action space consists of types of questions to ask. It is a given that, within any part of a test (within a subject area), we should confine the type of question to that part. Our freedom in choosing an action arises from how difficult a question actually is. In order to model this, I choose to represent actions as 1 of 5 possible question types, ranging from level 1 questions as "easiest" to level 5 as "hardest." The question difficulty levels are empirically determined from performance on the dataset such that the total performance on any given question is given by the total instances it was answered correctly divided by the total instances it was served. This is calculated with padding to ensure non-zero difficulty scores for rare and hard questions. Then the `question_ids` are mapped to one of the five levels such that the empirically easiest fifth are level 1 and the empirically hardest fifth are level 5 question actions.

The reward function involves more engineering, and further research into methods to model reward in this context is needed. I choose to model reward here as intuitively indicative of student learning such that a correct answer yields a higher reward than an incorrect answer. To discourage the outright avoidance of harder questions, the reward is scaled by the difficulty of the question. Further, since getting a question incorrect is still a valuable learning experience, incorrect answers are not penalized with a negative reward, but are significantly smaller. It is additionally relevant that not all questions are of the same degree of difficulty for all students. To model this, the reward is also scaled by a progress factor, which serves to scale reward up (whether incorrect or not) if a student is doing poorly and scale a reward down if they are doing well, encouraging the policy to, in principle, learn to serve harder (higher reward) questions to students doing well without penalizing students doing poorly. The code for this function is reproduced below:

```
def get_reward(correct: bool, question_data: dict,
                num_correct_so_far: int, num_incorrect_so_far: int):

    # calculate empirical difficulty score for given question in range
         (0, 1]
    eps = 0.01
    score = (question_data[0] + eps) / (question_data[1] + eps)

    # make difficulties pronounced and flip with -log, and normalize
        with sigmoid
    # harder questions have higher difficulty score
    norm_difficulty = sigmoid(-math.log(score))

    # calculate student progress to scale reward up/down if student is
         doing too poorly/well
    progress_factor = 1 - (num_correct_so_far / (num_correct_so_far +
        num_incorrect_so_far + 1))

    # calculate reward from correctness, difficulty and student
        progress
    base_correct = 100
    base_incorrect = 1
    if correct:
        reward = base_correct * norm_difficulty * progress_factor
    else:
        reward = base_incorrect * norm_difficulty * progress_factor

    # round to 3 decimal places for space efficiency
    return round(reward, 3)
```

This modeling choices enable a representation of the original dataset in a [state, action, reward, next-state] space which serves as the foundation for the training of our learning model.

3.3 Learning

In order to learn an optimal policy from this space, I use Q-learning, an off-policy reinforcement learning algorithm which here iteratively updates a q-table representing the q-values associated with taking an action from a given state:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\Big(r + \gamma \max_{a'} Q(s',a') - Q(s,a)\Big)$$

[3] This algorithm updates the q-table values for s, a by adding to that value a learning rate α times the sum of the immediate reward r plus the discount factor γ times the max q-value associated with the next possible actions from the next state, minus the value of the current q-value. This fundamentally represents the values of a in s as a function of the future values given by actions available in the next state, where a higher γ represents weighting the future more and vice versa.

4 Experiments

I first preprocessed with `preprocess.py` the original dataset into the [state, action, reward, next state] tuples as describe above. Experimentation was done with the hyperparameters of the reward function specifically to tune it to a degree sufficient to pronounce the effect of correct and incorrect questions. The final hyperparameters and transformations on parameters are expressed in the above code segment.

For Q-Learning, I implement a Q-Learning algorithm in `qlearning.py` which builds and iteratively updates a q-table according to our algorithm above, and after terminating, selects the action associated with the maximum q-value for each state to produce a deterministic policy of state to action for all states.

I use a decaying learning rate such that earlier updates are given by larger values of α and later updates are finetuned with smaller values. Specifically, at iteration i,

$$\alpha_i = \alpha_0(d^i)$$

where $\alpha_0 = 0.5$, $d = 0.9$, and the total number of iterations is 35. Ideally, we would use more iterations, but the computational cost associated with this dataset is such that 35 iterations takes approximately 12 hours to terminate on a personal computer.

I also use a discount factor of $\gamma = 0.99$, which is relatively high, indicating that we want to weight future reward significantly because the nature of a quiz is to increase the total learning occurring therein rather than near-sighted in the scope of merely the present question.

To evaluate, I use expected reward for our learned policy as compared to a random policy. I run 1000 iterations of randomly initialized 20 question episodes from the dataset, evaluating the reward of taking action a at s, resetting s to be randomly one of the next valid corresponding states of taking a from s or, if no such state exists, a new random state, and summing each step reward with a discount factor to get the total episode reward. I average the reward per episode to get discounted sums of rewards for both the learned policy and the random policy, yielding a value for expected reward. This is given by an `evaluate_policy` function, available in the source code.

5 Results

Evaluation as specified above yields that the Q-Learning policy has a 14% increase in expected reward over baseline random policy.

Policy	**Expected Reward per 20 Step Episode**
Learned Policy	246.132
Random Policy	215.678

Table 1: Comparison of Expected Rewards

Previous iterations of the process, including a different model of the action space as simply one of three categories (easy, medium, and hard), showed a smaller increase in performance.

The distribution of question levels in the final policy learned by the Q-Learning algorithm is given by Table 2:

Question Level	**Number of Instances**
1	93,475
2	108,361
3	139,095
4	173,098
5	218,823

Table 2: Number of Learned Question Recommendations by Question Level

This shows a bias toward harder questions, which have a larger reward if correct. This suggests potential need of refinement of the reward function, especially considering that incorrect answers in the current model are not penalized.

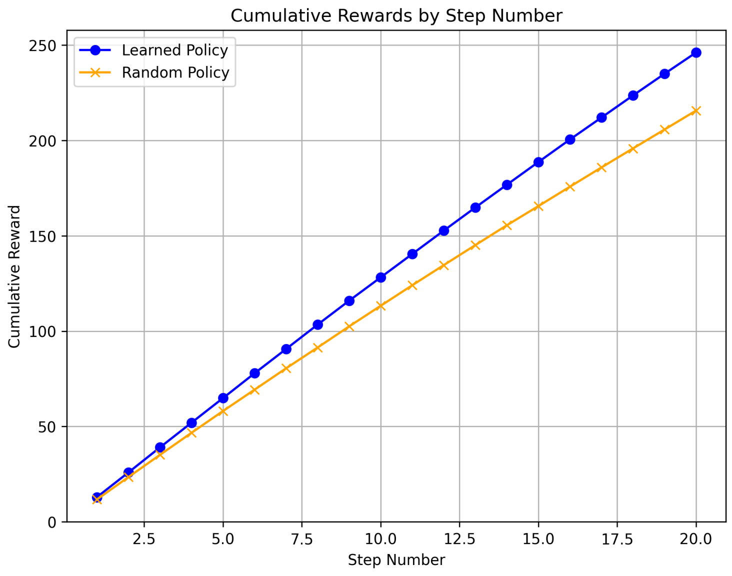

Figure 1: Cumulative expected reward per step on the 20 step evaluation.

6 Discussion

The 14% increase in expected reward for the learned policy over the random baseline indicates that the current model specifications do allow for Q-Learning to successfully learn a better-than-random policy. 14% is not a huge increase, but it is non-trivial, and suggests that Q-Learning can potentially aid in adaptive quizzing.

There are several limitations on this result. To begin, the reward function used here is by no means definite. It is unclear and an open field of inquiry as to how we best ought to model reward associated with quizzing events. The choice to not penalize incorrect questions means that all questions, regardless of correctness, do receive a positive reward, and since harder questions have a higher reward, we can see from Table 2 that the model learns to recommend hard questions at more than twice the rate of easy questions. This bias is somewhat unintuitive and suggests that the reward function may push a signal that does not align well with real education experience. Further, while the student progress factor and state space modeling choices attempt to make quiz events well-personalized, there is a sufficient lack of granularity in the data and model that it is not obvious that these results would translate well to an actual quizzing environment.

Further experimentation in modeling reward is needed. More, further detail in the state space would likely also contribute to a more robust model, where more granular student and question data would lead to more fine-grained understanding of the knowledge level of the given student at a given timepoint, which could also inform a better reward function.

7 Conclusion

Recent decreased student performance motivates an exploration of new methods in adaptive quizzing to increase student learning. The present paper takes a real-world dataset of student quiz performance and transforms it into a [state, action, reward, next state] space to apply Q-Learning in order to learn an optimal policy for next-question recommendation, where the action space consists of five different difficulty levels of questions. Experimentation finds a 14% increase in expected reward over a random baseline, suggesting that Q-Learning indeed can improve adaptive quizzing, at least in this context. Further research is needed to determine the extent to which this finding applies in real-world quizzing environments, where increased quiz, question, and student detail would likely produce a more robust policy.

References

[1] Mohamed Boussakssou, Bader Hssina, and Mohammed Erittali. Towards an adaptive e-learning system based on q-learning algorithm. *Procedia Computer Science*, 2020.

[2] Sequoia Carrillo. U.s. reading and math scores drop to lowest level in decades. *NPR*, 2023.

[3] Mykel J. Kochenderfer, Tim A. Wheeler, and Kyle H. Wray. *Algorithms for Decision Making*. MIT Press, 2022.

[4] Kejie Mao, Qiwen Dong, Ye Wang, and Daocheng Honga. An exploratory approach to intelligent quiz question recommendation. *Procedia Computer Science*, 2022.

[5] Bisni Fahad Mon, Asma Wasfi, Mohammad Hayajneh, Ahmad Slim, and Najah Abu Ali. Reinforcement learning in education: A literature review. *Informatics*, 2023.

Code

The code for this project is available at this GitHub repository.