How to “do” school: Optimizing within injustice

Jiahui Chen[*]
Stanford University
jchen23@stanford.edu

Parth Sarin[*]
Stanford University
psarin@cs.stanford.edu

Regina Ta[*]
Stanford University
rta@stanford.edu

Abstract

Simulated educational data is being used more in educational research to address some of the practical and legal challenges with data collection in classrooms. In that tradition, we model a classroom with students and teachers as POMDPs and use various planning algorithms to learn policy. We analyze the optimal policies and what they say about how students optimize within unjust systems. Finally, we reflect on the efficacy of simulated data in education and its history.[†]

1 Introduction

Especially in America, education is often perceived as a social good: an enterprise where teachers help students discover themselves, learn interesting subjects, and enrich their lives. In reality, education also involves socialization, violence, suppression, and—for many—learning is not always the priority [5, 4]. This project explores the decisions that students and teachers must make about how to spend their time after school lets out. One goal is to make visible how, as the result of structural features, everyone might find themselves simply trying to stay above water, even if they are interested in learning.

It’s very challenging to get access to real data in this context for a few reasons: the subjectivity of the observations that students and teachers make (about their mental health, for example) raises reliability concerns [3]; systematically collecting this kind of data is expensive, and the relative availability of other methodologies (like ethnography) has led to many interesting studies that don’t involve quantitative data collection [9, 3]; and, there are overlapping legal protections that limit the collection and dissemination of educational data involving minors [19].

As a result, we rely on simulated data for this project. We do so acknowledging that we made several modeling decisions that are not reflective of reality and, when they are, are based on *our* experiences as students which are not representative of the student population in the United States. Moreover, this project is meant to critique what we see as an increasing trend where quantitative education research is being performed on simulated data [8, 16]. Especially with the growing popularity of large language models, more researchers are trying to simulate data to evaluate teachers and curricula and predict how students will behave [6]. In our writing and analysis, we try to center a more critical approach to data based on the application of Critical Race Theory to statistics called QuantCrit [7]. This upsets the central principle that data should be used for planning in the way that we did for the project because “social relationships are not readily amenable to quantification” [7] — while it might be theoretically possible, quantifying the relationships in a classroom in a way that centers race as an organizing principle (and then learning policy in that context) is inaccessible, especially for this project. And yet, researchers are trying to do just this. What follows is as much a final project for CS 238 as it is an investigation of this scholarly trend.

[*]Denotes equal contribution
[†]Source code available at `https://github.com/parthsarin/cs238-final-project`

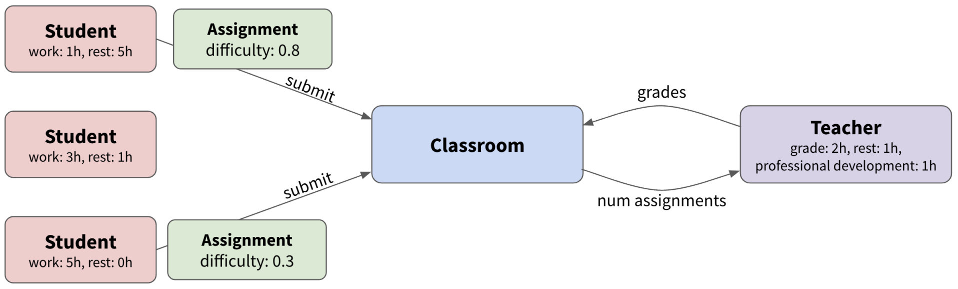

Figure 1: The agents and environments that interact in our model of the classroom

1.1 Environment

Figure 1 depicts a typical interaction in the classroom model that we developed. Each classroom consists of 35 students and one teacher, who are facing different decision-making problems and interact with each other through the submission of assignments and return of grades.

Observation: At the end of each day, students and teachers observe the amount of `free_time` they have when they get home (uniformly chosen between 0 and 7 hours) as well as their workload (number of assignments to complete/grade). If the student has had a returned assignment, they also observe their grade.

Action: Students and teachers need to decide how to allocate their free time between rest and work or rest, grading, and professional development, respectively. Every 10 days, students are assigned homework (they also have a `submit` action). Teachers grade submitted assignments, and their efficiency in grading improves as they complete professional development.

State: Internally, students and teachers have a mental health score (between -1 and 1), a productivity score (between 0 and 1) and a vector of competencies (each competency g_i is between 0 and 1). The psychometric technique of quantifying intelligence was promoted by Galton in the search for a "law of inheritance" to measure how intelligence deviated among offspring [15, 2, Chapter 1]. Figure 2 shows a visualization from his notes that he used to model how "positive deviations" in a generation could be amplified in the next. Those ideas were central to eugenics and, though they've largely been disavowed, underlie much of modern quantitative education research and standardized testing [12, 13]. Because of that history, we believe our approach is not an inaccurate rendition of the way intelligence has been modeled in the literature.

Interaction: The quality of an assignment is determined based on the student's productivity, the number of hours they work on the assignment, and their competencies. The grade they receive is based on the teacher's productivity and competency.

State transitions: As one might expect, rest improves mental health and work improves productivity. Teachers can improve their competency by completing professional development and, for students, receiving feedback from their teacher improves their competencies.

2 Literature Review

2.1 Simulated data in educational research

The application of simulated data in educational research has gained attention as a solution to some of the challenges in data collection. Ndukwe (2018) and Vemuri (2020) explore the potential of simulated data to provide consistent and controlled variables for complex educational studies [8, 16].

In contrast, Gillborn (2018) critically examines the limitations of simulated data, arguing that it may not accurately reflect the complexities of real-world educational environments [7]. They provided a critical examination of the field, exposing and challenging hidden assumptions that frequently encode racist perspectives beneath the facade of supposed quantitative objectivity. The authors highlight the complexities and biases inherent in data collection and statistical analysis within the education sector, particularly concerning critical race theory and the implications for education policy and 'Big Data' utilization.

2.2 Quantitative student-teacher interaction modeling

Quantitative student-teacher interaction modeling aims to understand and quantify the dynamics of interactions between students and teachers. Several studies have explored the impact of teacher-student interaction on various educational aspects. For instance, Chen (2022) demonstrated quantitatively that teacher-student interaction significantly affects the development of teachers' reflective practice skills [1]. Similarly, Wang (2022) proposed a two-stage algorithm for maximizing teacher-student interaction based on learning input responses, enabling comprehensive quantification of interaction [17]. Furthermore, Xie et al. (2022) designed a conversational model for teacher-student interaction, determining dialogue opportunities and characteristics of active dialogue development [18].

In addition, researchers have applied POMDPs to model student-teacher interactions. Rafferty (2011) used POMDPs to represent how teachers infer their students' knowledge base and plan future actions to maximize their students' learning [10]. Shenfeld (2023) used POMDPs to find optimal policies for students, as they navigate the balance between following teacher guidance and learning to solve tasks on their own [11]. Engaging with this previous work, we hope to apply POMDPs to model student-teacher interactions within a school system for an extended period of time.

3 Methodology

3.1 Dealing with infinite spaces

Much to our chagrin, the observation space $\mathcal{O}$, the state space $\mathcal{S}$, and the action space $\mathcal{A}$ are all continuous, infinite spaces.

The first two are standard challenges: We need to train on a finite collection of examples and then extend our policy to observations which weren't part of training. Parameterized policies (like neural networks) do this automatically — otherwise, we defined a distance metric on our observation space by embedding $\mathcal{O} \to \mathbb{R}^2$ where each observation o is represented as the tuple $(\texttt{free\_time}, \texttt{num\_assignments})$ (or $\mathbb{R}^3$ with `assignment_grade` for students). Then, we learned a memoryless policy π at some collection of observations $O^* \subset \mathcal{O}$ and generalized with a vote

$$\pi(o) = \operatorname{mode}\left(\left\{\pi(o^*) : o^* \in \mathrm{KNN}_{O^*}(o, k)\right\}\right)$$

where $\mathrm{KNN}_{O^*}(o, k)$ are the nearest k points in O^* to o under the L^2 metric in $\mathbb{R}^n$. This approach generalizes to histories rather than single observations (a history of length k lies in $(\mathbb{R}^2)^k$).

An infinite action space poses a more significant challenge as it renders many exploration strategies ineffective. Consider exploring using UCB1, where we select the action that maximizes

$$Q(h, a) + c\sqrt{\frac{\log N(h)}{N(h, a)}}.$$

The issue is highlighted in red: Since $\mathcal{A}$ is infinite, there are infinitely many actions where $N(h, a) = 0$ so they have an infinite score. This means we will always be (randomly) exploring.

We address this by discretizing the action space. Students have to choose between resting and working, so we pick a discretization with 12 options:

$$\begin{aligned}\mathcal{A}_s &= \{\texttt{submit}\} \cup \{(a_1, a_2) : a_1, a_2 \in \{0.0, 0.1, \ldots, 1.0\} \text{ and } a_1 + a_2 = 1\} \\ &= \{\texttt{submit}, (0.0, 1.0), (0.1, 0.9), \ldots, (1.0, 0.0)\}.\end{aligned}$$

Teachers also have the option of professional development, so their discretization has 62 options:

$$\begin{aligned}\mathcal{A}_t &= \{(a_1, a_2, a_3) : a_1, a_2, a_3 \in \{0.0, 0.1, \ldots, 1.0\} \text{ and } a_1 + a_2 + a_3 = 1\} \\ &= \{(0.0, 0.0, 1.0), (0.0, 0.1, 0.9), \ldots, (1.0, 0.0, 0.0)\}.\end{aligned}$$

3.2 Learning algorithms

Q-learning is an on-policy reinforcement learning algorithm. It is designed to learn the value of the policy being executed, including the action taken and the subsequent one. This characteristic makes Q-learning a robust choice for environments where the policy's exploration strategy impacts learning outcomes significantly. The algorithm updates the value function based on the reward and the estimated value of the next state-action pair, adapting iteratively to the environment. We've implemented Q-learning using the nearest neighbors approximation and linear interpolation.

The **Epsilon-Greedy** strategy is a simple yet effective method for balancing exploration and exploitation in reinforcement learning. This strategy selects actions by occasionally choosing a random action (exploration) with probability epsilon, and otherwise choosing the best-known action (exploitation). By adjusting the value of epsilon, one can control the balance between exploring new actions and exploiting known rewards. This approach is particularly useful in environments where the optimal policy is not known a priori, allowing the agent to discover new actions that might lead to higher rewards.

The **UCB1 exploration strategy** in reinforcement learning is designed to balance the trade-off between exploration and exploitation. This approach is utilized to select actions based on both their historical rewards and the uncertainty or variance associated with them. The UCB1 formula adjusts the action-selection process by considering the total number of times the current state has been visited and the number of times each action has been chosen. This mechanism allows the algorithm to preferentially explore actions that either have a high reward estimate or have been less frequently chosen, thereby encouraging exploration of less known actions. (For more details on the implementation of the Q-learning algorithms above, see Appendix B for our pseudocode.)

Finally, we used two variants on **Deep Q-Learning**. In this algorithm, the Q function is modeled using a multi-layer perceptron. For a memoryless DQN, we used the aforementioned embedding $\mathcal{O} \to \mathbb{R}^2$ (or $\mathbb{R}^3$ for students) to generate the inputs into the neural network. The output is a vector in $\mathbb{R}^{|\mathcal{A}|}$ where the ith component is $Q(o, a_i)$. During training, we predicted the Q values for each action and took the best action a corresponding to the UCB1 heuristic. Then, the actor received a reward r and a new observation o' and we updated the network according to the loss

$$\mathcal{L} = \left| Q(o,a) - \left(r + \gamma \max_{a'} Q(o', a') \right) \right|.$$

For the history-aware variant on this algorithm, we used the same loss function but allowed the Q network to see the entire history $h = (o_1, a_1, \ldots, o_{k-1}, a_{k-1}, o_k)$ and generate the action values based on that history: We stacked each observation and action tuple (o_i, a_i) into a single vector in $\mathbb{R}^5$ (or $\mathbb{R}^6$ for students) then ran it through an RNN and used the final hidden state to represent the entire history. We stacked that vector on top of the final observation vector o_k and fed that into an MLP.

4 Results

In this section, we'll compare the performance of each attempted learning algorithm to maximize rewards for students and teachers, as shown in Figure 3.

For students, the only algorithm that consistently outperformed the random policy was Deep Q-learning (both memoryless and history-aware variants). The learned policy is depicted in Figure 4. Since the random policy steadily reduced rewards over time, Q-learning with linear interpolation outperformed the random policy after around 150 days. The basic Q-learning algorithm, Q-learning with UCB1, and Q-learning with Epsilon-Greedy under-performed with respect to the random policy.

For teachers, we observed the inverse: the basic Q-learning algorithm, Q-learning with Epsilon-Greedy, and Q-learning with UCB1 consistently outperformed the random policy. All other algorithms (including Deep Q-learning and Q-learning with linear interpolation) under-performed with respect to the random policy.

Note: Every time we trained for one agent, the other agent was operating on a random policy. In other words, when we trained our algorithms for teachers, students were acting on a random policy, and vice versa. For the reported figures, we matched the student policy with the teacher policy and ran a simulation.

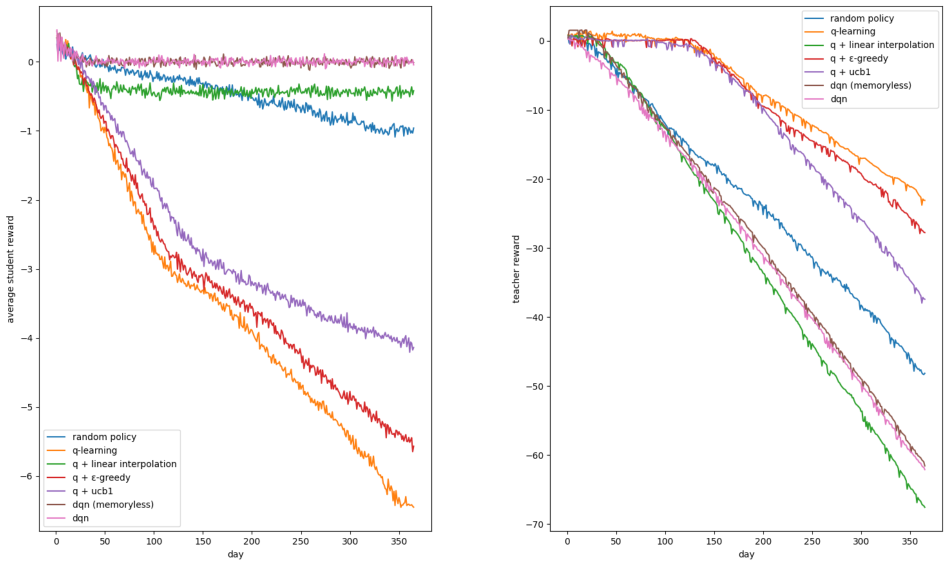

Figure 3: The rewards, over time, for each of the policies we trained in this environment. The figure on the left shows student reward and the figure on the right shows teacher reward.

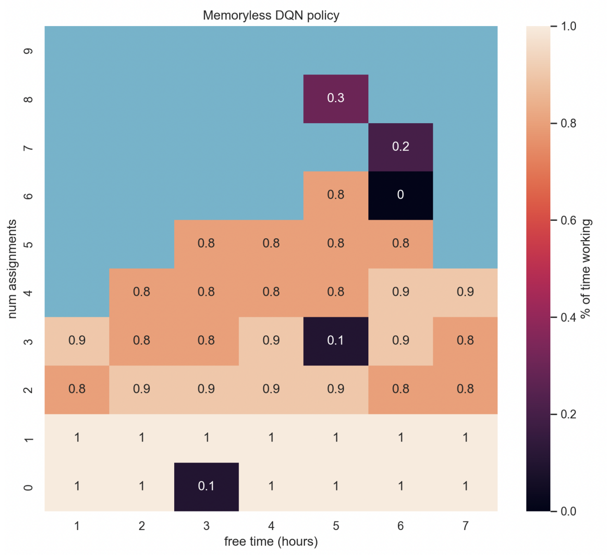

Figure 4: The optimal student policy, which came from the memoryless deep Q network. The color scale indicates the percentage of time allocated towards working and light blue indicates `submit`.

5 Discussion

Basic Q-learning, Q-learning with UCB1, and Q-learning with Epsilon-Greedy generated the most effective policies for teachers, while a random policy proved to be better at maximizing rewards than DQN and Q-learning with linear interpolation. On the other hand, DQN generated the most effective policy for students, while Q-learning with linear interpolation improved over time to eventually outperform the random policy. Our other attempted Q-learning variants (basic, UCB1, and Epsilon-Greedy) struggled to learn optimal policies for students. Note: our implementation of Q-learning with UCB1 involved applying the UCB1 exploration strategy directly during testing, not training.

One possible reason why Q-learning extended with exploration strategies like UCB1 and Epsilon-Greedy performed better for teachers than for students, is because teachers receive more immediate positive feedback through exploration, while students can receive short-term negative feedback through exploration. For instance, teachers may see quick improvements in grading speed through professional development, which would reinforce exploration strategies like UCB1 and Epsilon-Greedy that could help teachers discover more efficient practices.

However, when students explore new strategies to balance working and resting, they might receive an increased workload without an immediate improvement to their mental health or an academic advantage (e.g., a higher grade or feedback from their teacher to increase their competencies). These delayed rewards dis-incentivize students from further exploration, which could yield long-term success albeit with short-term disadvantages. Given these conditions, Q-learning with two exploration strategies might fail to serve students more effectively than a random policy. Randomness, in fact, might better enable students to adapt over time to a constantly changing environment with delayed rewards.

Basic Q-learning also performed better than the random policy for teachers, but not for students. This discrepancy could similarly be due to the fact that teacher rewards have a more immediate correlation with their corresponding actions. In other words, certain teacher actions can consistently be expected to yield higher rewards, enabling basic Q-learning to converge more efficiently toward optimal policies for teachers.

Q-learning with linear interpolation showed improvement for students but not for teachers. One possible reason could be that interpolating between different actions might help to students to "explore" the action space without incurring short-term negative feedback from exploration strategies like UCB1 or Epsilon-Greedy. Since students might have a less direct correlation between their rewards and actions compared to teachers, interpolating between different actions could allow our algorithm to explore more nuanced combinations of actions, which could lead to better learning and policy optimization for students.

Deep Q-learning generated a much better policy for students than for teachers. There are two possible reasons behind why introducing DQN led to a significant improvement for the student policy. First, our implementation of DQN applied UCB1 during training (not just during testing like Q-learning with UCB1), which meant that our algorithm was encouraged to explore a diverse set of student actions throughout the learning process, potentially discovering actions that lead to higher rewards for students. Applying UCB1 also prevented premature convergence to suboptimal policies. Second, since DQN leveraged a neural network, we can learn possible relationships between different observations. Related observations would be treated as completely distinct by a Q-dictionary or Q-table, but would be taken in as parameters multiplied by similar weights in a neural network.

6 Conclusion

Our experiments with various reinforcement learning algorithms in the context of simulated classrooms with students and teachers emphasized the importance of reward functions for each agent, which represent incentive structures in the American school system. The learning process for how each agent maximized their rewards affected the policy that each agent adopted, which represented the decision-making process for students and teachers as they approached school. In particular, the challenge of learning optimal policies for students (even when extending Q-learning with exploration strategies) suggested further investigation into how the immediacy of feedback can affect the balance between exploration and exploitation. We also hope to conduct future work on how these learned policies affect students and teachers' mental health and productivity. (See Appendix A for some starting experiments in this direction.)

6.1 Contributions

All group members collaborated on the code and write-up. **Parth** created our custom environment, including the `POMDP` class that we used to model the environment and helper functions for planning (like `lookahead`); implemented the vanilla Q-learning and deep Q-learning algorithms; wrote the Introduction, Methodology, and created the figures. **Jiahui** initialized `Teachers` and `Students`; implemented Q-learning with UCB1 and Epsilon-Greedy; wrote the Abstract, Literature Review, and Methodology. **Regina** initialized `Classroom`; implemented Q-learning with linear interpolation; wrote the Literature Review, Results, Discussion, and Conclusion.

References

[1] Z. Chen and R. Chen. Exploring the key influencing factors on teachers' reflective practice skill for sustainable learning: a mixed methods study. *International Journal of Environmental Research and Public Health*, 19(18):11630, 2022.

[2] Wendy Hui Kyong Chun. *Discriminating data: Correlation, neighborhoods, and the new politics of recognition*. MIT press, 2021.

[3] Paul Connolly, Ciara Keenan, and Karolina Urbanska. The trials of evidence-based practice in education: A systematic review of randomised controlled trials in education research 1980–2016. *Educational Research*, 60(3):276–291, 2018.

[4] Lee H Ehman. The American school in the political socialization process. *Review of educational Research*, 50(1):99–119, 1980.

[5] Paulo Freire. *Pedagogy of the oppressed*. The Continuum Pushing Company, 1970.

[6] Wensheng Gan, Zhenlian Qi, Jiayang Wu, and Jerry Chun-Wei Lin. Large language models in education: Vision and opportunities. *arXiv preprint arXiv:2311.13160*, 2023.

[7] David Gillborn, Paul Warmington, and Sean Demack. QuantCrit: Education, policy, 'Big Data' and principles for a critical race theory of statistics. *Race ethnicity and education*, 21(2):158–179, 2018.

[8] Ifeanyi G Ndukwe, Ben K Daniel, and Russell J Butson. Data science approach for simulating educational data: Towards the development of teaching outcome model (TOM). *Big Data and Cognitive Computing*, 2(3):24, 2018.

[9] Denise Clark Pope. *Doing school: How we are creating a generation of stressed out, materialistic, and miseducated students*. Yale University Press, 2008.

[10] Anna Rafferty, Emma Brunskill, Thomas Griffiths, and Patrick Shafto. Faster Teaching via POMDP Planning. 6738:280–287, 06 2011.

[11] Idan Shenfeld, Zhang-Wei Hong, Aviv Tamar, and Pulkit Agrawal. TGRL: Teacher guided reinforcement learning algorithm for POMDPs. In *Workshop on Reincarnating Reinforcement Learning at ICLR 2023*, 2023.

[12] Alan Stoskopf. The forgotten history of eugenics. *Rethinking schools*, 13(3):12–13, 1999.

[13] Alan Stoskopf. Echoes of a forgotten past: Eugenics, testing, and education reform. In *The Educational Forum*, volume 66, pages 126–133. Taylor & Francis, 2002.

[14] Frank J. Swetz. Mathematical Treasure: Francis Galton's Typical Laws of Heredity. *Convergence*, 2023.

[15] James M Tanner. Galtonian eugenics and the study of growth: the relation of body size, intelligence test score, and social circumstances in children and adults. *The Eugenics Review*, 58(3):122, 1966.

[16] Sidharth Vemuri, Jenny Hynson, Lynn Gillam, and Katrina Williams. Simulation-based research: a scoping review. *Qualitative Health Research*, 30(14):2351–2360, 2020.

[17] D. Wang. The mechanism of teacher influence on the learning engagement of students. *International Journal of Emerging Technologies in Learning (Ijet)*, 17(21):135–149, 2022.

[18] Y. Xie, Y. Huang, W. Luo, Y. Bai, Y. Qiu, and Z. Ouyang. Design and effects of the teacher-student interaction model in the online learning spaces. *Journal of Computing in Higher Education*, 35(1):69–90, 2022.

[19] Elise Young. Educational privacy in the online classroom: FERPA, MOOCs, and the big data conundrum. *Harv. JL & Tech.*, 28:549, 2014.

A Appendix A

As a bonus aside, by solving our POMDP with the above techniques, we can further analyze the effect of our represented school system on students and teachers. For instance, given the best policy for students generated by Deep Q-learning, we can see how their average mental health and productivity evolve over time (i.e., over 90 days which represents a quarter or a semester) in a classroom of 35 students.

Notice how students' mental health drops to negative values early in the quarter or semester, and only begins to climb afterwards the initial dip. This behavior might suggest how students struggle to balance their workload at the beginning of the quarter, before learning more optimal time-management strategies through trial and error.

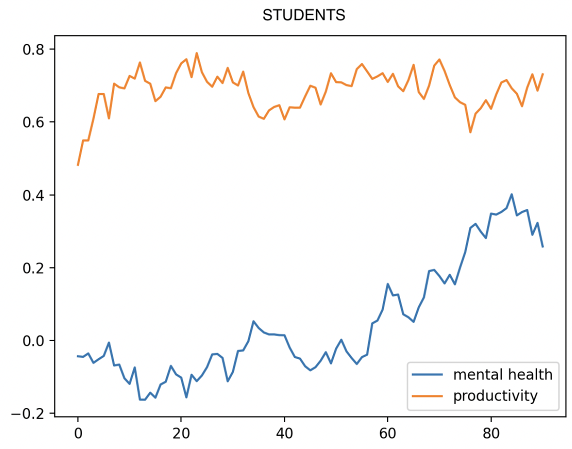

Figure 5: Students' Mental Health and Productivity over Time

Likewise, given the best policy for teachers generated by Q-learning, we can also see how the mental health and productivity evolve over the course of a quarter or a semester for a single teacher in a 35-student classroom.

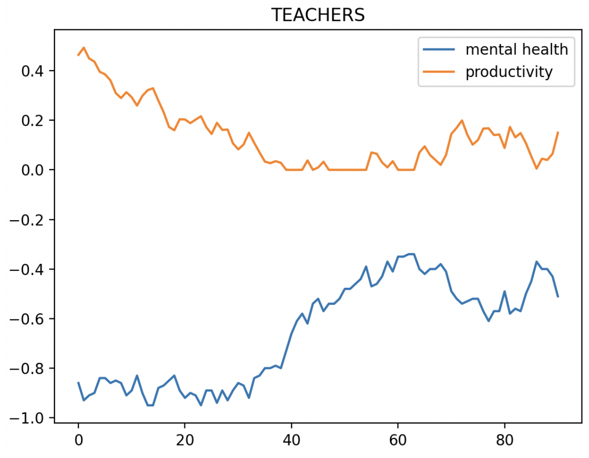

Figure 6: Teachers' Mental Health and Productivity over Time

B Appendix B

Algorithm 1 Q-Learning Policy

```
Require: $Q$ (Q-values), $obsOrdering$ (ordering of observation attributes), $actOrdering$ (ordering
    of action attributes), $actClass$ (action class), $qThresh$ (threshold for nearest neighbor), $isValid$
    (function to validate actions)
Ensure: Updated Q-values ($Q$)
 1: Initialize $Q$, $obsOrdering$, $actOrdering$, $actClass$, $qThresh$, $isValid$
 2: Extract states and actions from $Q$ table
 3: Build a KDTree for the states
 4: function FromQ($observation$)
 5:     Convert $observation$ to a normalized tuple
 6:     Retrieve nearest neighbor using KDTree
 7:     if distance to nearest neighbor $<$ $qThresh$ then
 8:         Retrieve possible actions for the nearest neighbor
 9:         Sort actions by Q-value in descending order
10:         for each action $a$ in sorted actions do
11:             if $a$ is valid then
12:                 return action $a$
13:             end if
14:         end for
15:     end if
16:     return a random action
17: end function
```

Algorithm 2 Q-Learning w/ Linear Interpolation

```
Require: $Q$ (Q-values), $obsOrdering$ (ordering of observation attributes), $actOrdering$ (ordering
    of action attributes), $actClass$ (action class), $qThresh$ (threshold for nearest neighbor), $isValid$
    (function to validate actions)
Ensure: Updated Q-values ($Q$)
 1: Initialize $Q$, $obsOrdering$, $actOrdering$, $actClass$, $qThresh$, $isValid$
 2: Extract states and actions from $Q$ table
 3: Build a KDTree for the states
 4: function FromQ($observation$)
 5:     Convert $observation$ to a normalized tuple
 6:     Load $Q$-dictionary
 7:     $Qsa$ = []
 8:     for each action $a$ do
 9:         if $o$ in $Q$.keys() then
10:             $Qsa$.append([o,a])
11:         else
12:             $X$ = np.array(Q.keys())
13:             $Y$ = np.array(Q.values())
14:             $lr\_model$ = LinearRegression()
15:             $lr\_model$.fit($X, Y$)
16:             $interpolated\_val$ = $lr\_model$.predict([o,a])
17:             $Q$[o, a] += lr * ($interpolated\_val$ - $Q$[o, a])
18:             $Qsa$.append($interpolated\_val$([o,a])
19:         end if
20:     end for
21:     return action np.argmax($Qsa$)
22: end function
```

Algorithm 3 Epsilon Greedy Algorithm

```
Require: $Q$ (Q-values), $obsOrdering$ (ordering of observation attributes), $actOrdering$ (ordering of action attributes), $actClass$ (action class), $qThresh$ (threshold for nearest neighbor), $isValid$ (function to validate actions), $\epsilon$ (the greedy parameter)
Ensure: Updated Q-values ($Q$)
 1: Initialize $Q$, $obsOrdering$, $actOrdering$, $actClass$, $qThresh$, $isValid$, $\epsilon$
 2: Extract states and actions from $Q$ table
 3: Build a KDTree for the states
 4: function FROMEPSILONGREEDY($observation$)
 5:     Convert $observation$ to a normalized tuple
 6:     if random value $< \epsilon$ then
 7:         Return a random action
 8:     end if
 9:     Retrieve nearest neighbor using KDTree
10:     if distance to nearest neighbor $< qThresh$ then
11:         Retrieve possible actions for the nearest neighbor
12:         Sort actions by Q-value in descending order
13:         for each action $a$ in sorted actions do
14:             if $a$ is valid then
15:                 return action $a$
16:             end if
17:         end for
18:     end if
19:     return a random action
20: end function
```

Algorithm 4 UCB1 Exploration Strategy

```
Require: $Q$ (Q-values), $c$ (exploration parameter), $OCounter$ (counter for observations), $OACounter$ (counter for observation-action pairs)
Ensure: Updated Q-values ($Q$)
 1: function UCB1($Q$, $c$, $oCounter$, $oACounter$, $actionSpace$, $currentObservation$)
 2:     Initialize $nextAction$ as an empty set
 3:     for each action $a$ in $actionSpace$ do
 4:         if $OACounter[currentObservation, a] = 0$ then
 5:             $bonus \leftarrow \infty$
 6:         else
 7:             $bonus \leftarrow c \cdot \sqrt{\frac{\log(OCounter[currentObservation])}{OACounter[currentObservation, a]}}$
 8:         end if
 9:         $nextAction[a] \leftarrow Q[currentObservation, a] + bonus$
10:     end for
11:     if $nextAction$ is empty then
12:         return a random action
13:     end if
14:     Choose the action $a_{max}$ with the maximum value in $nextAction$
15:     return $a_{max}$
16: end function
```

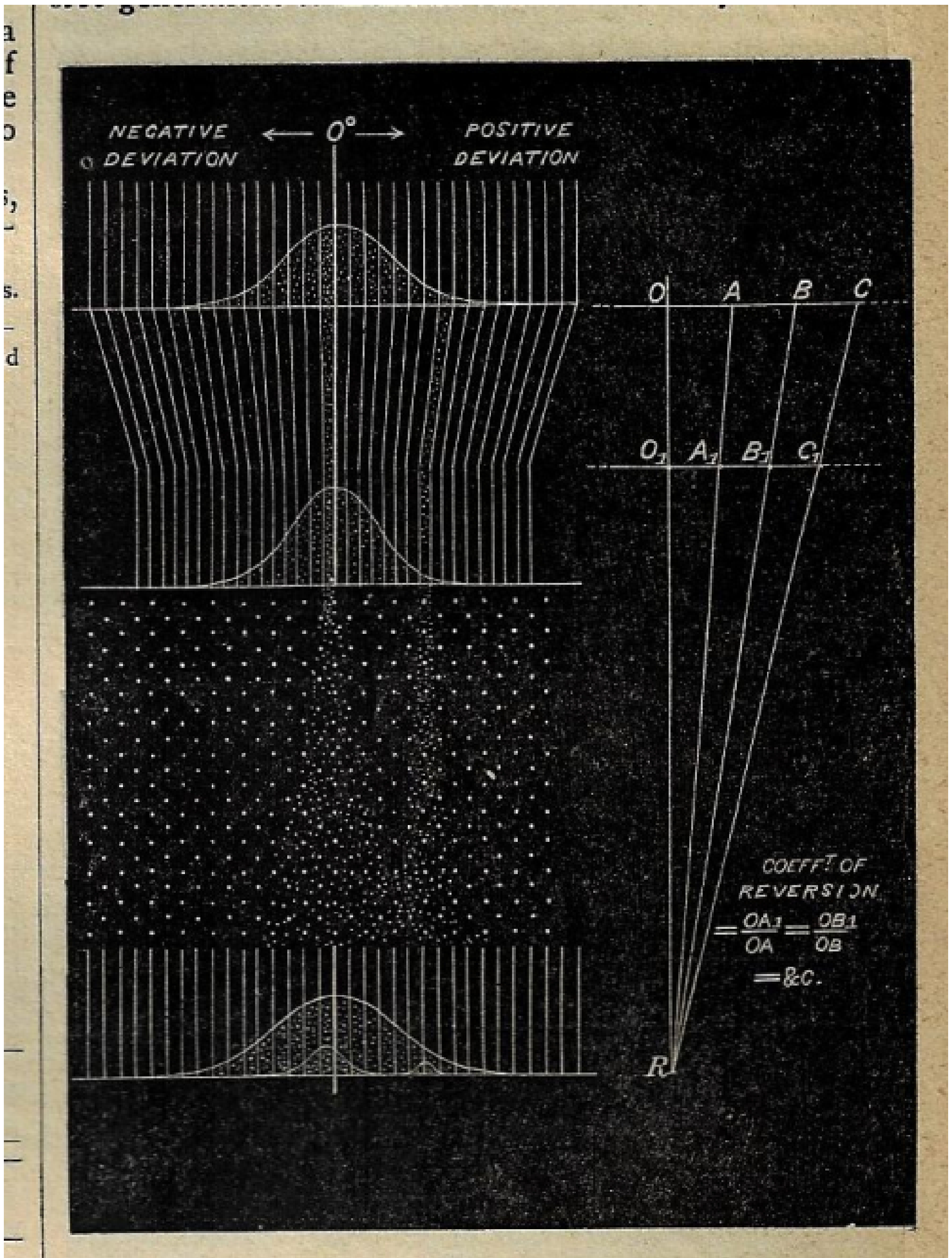

FIG. I.

Figure 2: Francis Galton, who developed a simulation called the “Galton board” (often used to visualize the central limit theorem) and first produced what would become linear regression, was centrally concerned with preserving and amplifying “good” deviations among offspring. This image is taken from his notes [14].