Accelerating Human Learning with Deep Reinforcement Learning

Siddharth Reddy, Sergey Levine, Anca Dragan
Department of Electrical Engineering and Computer Science
University of California, Berkeley
{sgr, svlevine, anca}@berkeley.edu

Abstract

Guiding a student through a sequence of lessons and helping them retain knowledge is one of the central challenges in education. Online learning platforms like Khan Academy and Duolingo tackle this problem in part by using interaction data to estimate student proficiency and recommend content. While the literature proposes a variety of algorithms for modeling student learning, there is relatively little work on principled methods for sequentially choosing items for the student to review in order to maximize learning. We study this decision problem as an instance of reinforcement learning, and draw on recent advances in training deep neural networks to learn flexible and scalable teaching policies that select the next item to review. Our primary contribution is an analysis of a model-free review scheduling algorithm for spaced repetition systems that does not explicitly model the student, and instead learns a policy that directly operates on raw observations of the study history. As a preliminary study, we train and evaluate this method using a student simulator based on cognitive models of human memory. Results show that model-free scheduling is competitive against widely-used heuristics like SuperMemo and the Leitner system on various learning objectives and student models.

1 Introduction

The ability to learn and retain a large number of new pieces of information is an essential component of human education. Scientific theories of human memory, dating back to 1885 and the pioneering work of Ebbinghaus [10], identify two critical variables that determine the probability of recalling an item: *reinforcement*, or repeated exposure to the item; and *delay*, or time since the item was last reviewed. More recent work characterizes the functional form of recall by combining psychological theories with modern machine learning techniques. These memory models can be used to understand the effects of study history – the sequence of pass-fail outcomes and intervals between reviews for a student-item pair – on learning. For example, many of them formalize the *spacing effect* [8]: the phenomenon in which periodic, spaced review of content improves long-term retention.

In recent years, a growing body of work has attempted to leverage our scientific understanding of human memory to engineer the process of human learning using educational technology. Flashcards are one such technology that use the idea of *spaced repetition* to overcome human forgetting. Though they have been around for a while in the physical form, a new generation of electronic flashcard software including SuperMemo [37] and Anki [12] enables a much greater degree of control and monitoring of the review process. These applications are growing in popularity [2], but formal mathematical models for reasoning about and optimizing these systems are lacking.

The core function of spaced repetition software is *review scheduling*: deciding which content to present to the student at any given time. This problem is challenging because schedules must balance competing priorities of introducing new items and reviewing old items in order to maximize learning. Most systems use heuristics to make this trade-off, and there are only a few theoretical frameworks

[19, 27] that present a principled understanding of the tension between novelty and reinforcement. Furthermore, existing systems are tailored to specific learning objectives and student models, which limits their flexibility. Modeling student learning is difficult – potentially harder than learning a good scheduling policy.

The motivation for this work is that faithfully modeling the student and accurately predicting outcomes is merely a surrogate for helping the student achieve their learning objective, so training a policy that directly optimizes for the learning objective without going through the indirection of first learning a student model that optimizes for predictive accuracy may yield more effective policies that require less data to train. We hypothesize that model-free reinforcement learning can learn effective content-selection strategies without explicitly modeling the student, and perform at least as well as complex heuristic policies.

Deep reinforcement learning (DRL) has recently achieved remarkable success in training agents to play video games, defeat human world champions at Go, and control robots [3]. We observe that spaced repetition can be formulated as a task in which the agent receives observations of the student answering questions, takes actions that influence what the student learns and reviews, and earns rewards when the student achieves their educational objectives. Our primary contribution is a review scheduling algorithm that uses model-free reinforcement learning with neural network function approximation to learn a teaching policy that (1) directly operates on raw observations of outcomes and intervals between reviews, (2) scales to large numbers of items, and (3) easily adapts to different learning objectives and student models.

One of the drawbacks of DRL is its sample complexity: learning a policy requires a large number of interactions with the environment. Training an autonomous tutor from scratch using a large number of interactions with real students is infeasible. In this preliminary study, we tackle the problem by using generative models of study histories based on cognitive models of human memory to simulate students, providing training environments for the teaching agent. Our ultimate vision is to interactively train with real students – Section 6.1 outlines ideas for overcoming the sample complexity of DRL in this setting.

2 Related Work

[25] formulates teaching as a partially-observable Markov Decision Process (POMDP), and uses a particle-based solver to compute an approximately optimal policy. [22] uses expectimax search to select teaching actions. [14] formulates personalized action selection as a contextual bandit problem. [34] uses a policy gradient algorithm to train a teacher that uses a particle-based belief update mechanism to track the state of a Bayesian student.

The idea of applying DRL to autonomous tutoring is briefly mentioned in [23]. We flesh out this early seed of an idea with modern DRL algorithms and synthetic experiments in the context of spaced repetition.

[17] develops a reinforcement learning agent for an adaptive tutoring task in a challenging real-world educational game application, using an offline importance sampling-based method to select representations and hyperpameter settings without expensive online experiments on real students.

Existing spaced repetition systems rely on heuristics for review scheduling. The Leitner system [16, 27] uses a network of first-in-first-out queues to coarsely prioritize items by novelty and difficulty. Pimsleur [24], SuperMemo [37], Anki [12], and Mnemosyne [1] use layers of handcrafted rules to decide when to next review an item and to prioritize items within a session. [18] use a threshold-based policy that selects the item with predicted recall likelihood closest to some fixed threshold $\theta \in [0, 1]$.

This work is broadly related to *knowledge tracing*, the problem of estimating how a student's knowledge changes over time as they interact with content [7, 15, 11, 22, 35], and *machine teaching*, the problem of finding the optimal training set for a given learning algorithm [38, 20, 30].

3 Background

In this section, we briefly introduce three probabilistic models of human memory that have previously been proposed in the psychology and educational data mining literature, and recap the reinforcement learning problem statement.

3.1 Models of Human Memory

Exponential forgetting curve. Ebbinghaus' exponential forgetting curve [10] is one of the simplest and oldest models of human memory. In this paper, we use the parameterization proposed in [27], where recall is binary (i.e., a user either completely recalls or forgets an item) and the probability of recalling an item has the following functional form:

$$Z \sim \text{Bernoulli}\left(\exp\left(-\theta \cdot \frac{D}{S}\right)\right)$$

where $\theta \in \mathbb{R}_+$ is the item difficulty, $D \in \mathbb{R}_+$ is the time elapsed since the item was last reviewed by the student, and $S \in \mathbb{R}_+$ is the student's memory strength for the item. In our experiments, we assume a simple model of memory strength evaluated in [27] that sets S to be the number of attempts. Henceforth, we refer to this model as EFC.

Half-life regression. HLR [29] extends the exponential forgetting curve by dropping the item difficulty θ and adding a log-linear model of memory strength:

$$S = \exp\left(\vec{\theta} \cdot \vec{x}\right)$$

where $\vec{x} \in \mathcal{X}$ is a vector of features describing the study history for the student-item pair and $\vec{\theta} \in \Theta$ are the model parameters. In our experiments, we adopt a setup similar to [29] and set $\mathcal{X} = \mathbb{N}^3 \times \{0,1\}^n$ to encode the number of attempts, correct answers, incorrect answers, and identity of the item (one of n items). Note that the item difficulty from EFC is absorbed into the memory strength via the coefficients of the item identity indicator features.

Generalized power-law. Drawing on ideas from the Rasch model [26], additive-factors models [21], and power-law forgetting curves [36], [18] proposes the following recall likelihood:

$$Z \sim \text{Bernoulli}(m(1 + r \cdot D)^{-f}) \tag{1}$$

where $D \in \mathbb{R}_+$ is the time elapsed since the item was last reviewed by the student, $r \in \mathbb{R}_+$ is a constant that controls the decay rate,

$$m = \sigma(a - d + h_\theta(\mathbf{t}_{1:k}, \mathbf{z}_{1:k-1}))) \tag{2}$$

, σ is the logistic function, $\mathbf{t}_{1:k}$ are the times at which reviews 1 through k occurred, $\mathbf{z}_{1:k-1}$ are the outcomes on reviews 1 through $k-1$, h_θ summarizes the effect of study history on recall probability, θ are the parameters that govern h,

$$h_\theta(\mathbf{t}_{1:k}, \mathbf{z}_{1:k-1}) = \sum_{w=1}^{W} \theta_{2w-1} \log\left(1 + c_w\right) + \theta_{2w} \log\left(1 + n_w\right)$$

, $w \in \{1, ..., W\}$ is an index over time windows (smaller w means more recent), c_w is the number of times the student correctly recalled the item in window w out of n_w attempts, θ are window-specific weights,

$$f = \exp\left(\tilde{a} - \tilde{d}\right) \tag{3}$$

, and $\tilde{a}$ and $\tilde{d}$ are additional student ability and item difficulty parameters respectively. In [18], Equations 1 and 3 without h_θ are called HYBRID BOTH and Equation 2 is called DASH. Henceforth, we refer to Equations 1, 3, and 2 as GPL.

3.2 Reinforcement Learning

Reinforcement learning [31] is a framework for sequential decision-making under uncertainty. Consider a Markov Decision Process (MDP) with a set of states $\mathcal{S}$, actions $\mathcal{A}$, transition distribution $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$, reward function $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, and discount factor $\gamma \in [0, 1]$. The agent's goal is to find a policy $\pi^* : \mathcal{S} \times \mathcal{A} \to [0, 1]$ that maximizes expected future discounted return:

$$\pi^* = \underset{\pi \in \Pi}{\operatorname{argmax}} \, \mathbb{E}_{s_0, a_0, \ldots} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \mid \pi \right]$$

4 Spaced Repetition via Model-Free Reinforcement Learning

Prior work has formulated teaching as a partially-observable Markov decision processes (POMDP) (e.g., [25]). We take a similar approach to formalizing spaced repetition as a POMDP.

4.1 Formulation

The state space $\mathcal{S}$ depends on the student model. For EFC, $\mathcal{S} = \mathbb{R}_+^{3n}$ encodes the item difficulty, delay, and memory strength for n items. For HLR, $\mathcal{S} = \Theta \times (\mathbb{R}_+ \times \mathcal{X})^n$ encodes the model parameters, delay, and memory strength for n items. For GPL, $\mathcal{S} = \mathbb{R} \times (\mathbb{R} \times \mathbb{N}^{2W})^n$ encodes the student ability, item difficulty, number of attempts over W windows, and number of correct answers over W windows for n items. Note that the state space is only relevant for building the student simulator – the teaching agent cannot directly access the state, and instead receives observations (see below). The action space $\mathcal{A} = [n]$ consists of n items that the agent can show to the student, where $[n] = \{1, 2, ..., n\}$. The transition function $T(s_{t+1} \mid s_t, a)$ depends on the student model. We do not explicitly write out the transition functions for each of the three student models, but they are straightforward to understand: static hyperparameters like student ability, item difficulty, and log-linear model coefficients are held constant, and dynamic quantities like number of attempts and correct answers are deterministically incremented.

The reward function depends on the learning objective. If the student's goal is to maximize the expected number of items recalled, then

$$R(s, \cdot) = \sum_{i=1}^{n} \mathbb{P}[Z_i = 1 \mid s] \tag{4}$$

To maximize the likelihood of recalling *all* items,

$$R(s, \cdot) = \sum_{i=1}^{n} \log \mathbb{P}[Z_i = 1 \mid s] \tag{5}$$

The discount factor γ encodes the urgency of student learning. Smaller γ encourages the agent to help the student cram, and larger γ promotes life-long learning. At each timestep, the agent receives observation $o \in \mathcal{O}$, where $\mathcal{O} = \{0, 1\}$ encodes whether or not the student correctly recalled the item shown to them. The observation distribution $O(z \mid s, a) = \mathbb{P}[Z_a = z \mid s]$ is given by the recall likelihood specified by the student model.

4.2 Algorithm

We approximately solve the POMDP $(\mathcal{S}, \mathcal{A}, T, R, \mathcal{O}, O, \gamma)$ using trust region policy optimization (TRPO) [28] with a gated recurrent unit (GRU) neural network policy architecture [6]; specifically, an off-the-shelf implementation from rllab [9]. At each timestep, the policy takes as input a triple

containing the identity of the previous item, the binary outcome indicating whether or not it was correctly recalled, and the time elapsed since that event, and outputs the identity of the next item to be shown to the user. We use the random projection trick described in [22] to reduce the dimensionality of the one-hot encoding of the item identity in the policy input, enabling the policy to scale to large numbers of items.

5 Experiments

Our experiments aim to answer the following research question: can the DRL scheduling algorithm described in Section 4 learn to help students achieve their educational objectives better than baseline methods, under various assumptions about student learning? We manipulate the student learning model underlying the environment, and measure the teaching performance of the DRL scheduler relative to a control group of random and heuristic policies. We study three student learning models, i.e., environments: EFC, HLR, and GPL; two teaching performance metrics: expected recall likelihood and log-likelihood; and four baseline policies: RANDOM, LEITNER, SUPERMNEMO, and THRESHOLD.

Environments. For scheduler training and evaluation, we implemented three student simulators based on the EFC, HLR, and GPL memory models described in Section 3.1 as OpenAI Gym [5] environments.

Baselines. We compare TRPO to several baseline schedulers: (1) a random policy that selects an item uniformly at random; (2) a Leitner system with arrival rate λ, infinitely many queues, and a sampling distribution $p_i \propto 1/\sqrt{i}$ over non-empty queues $\{1, 2, ..., \infty\}$; (3) a variant of SuperMemo (SM3) implemented in the Mnemosyne flashcard software; and (4) a threshold-based policy that selects the item with predicted recall likelihood closest to some fixed threshold $z^* \in [0, 1]$, conditioned on the appropriate student model. λ and z^* are selected to maximize reward using a uniformly-spaced hyperparameter sweep over the unit interval. Note that (4) is 'cheating' in that it uses direct access to the latent parameters of the student simulator to compute recall likelihoods, and is inspired by Bjork's notion of *desirable difficulty* [4]. We expect (4) to serve as a rough upper bound in our controlled experiments, not necessarily as a practical example of a baseline scheduling algorithm.

Implementation details. Roughly following the parameter settings for the Amazon Mechanical Turk experiments described in [27], we set the number of items $n = 30$, number of steps per episode $T = 200$, and constant delay $D = 5$ seconds between steps. For the EFC student model, we sample item difficulty θ from a log-normal distribution: $\log \theta \sim \mathcal{N}(\log 0.077, 1)$. For the HLR student model, we set the memory strength model parameters to be $\vec{\theta} = (1, 1, 0, \theta_3 \sim \mathcal{N}(0, 1))$ and the features for item i to be $\vec{x}_i = (\text{num attempts}, \text{num correct}, \text{num incorrect}, \text{one-hot encoding of item } i \text{ out of } n \text{ items})$. For the GPL student model, we set student abilities $a = \tilde{a} = 0$, sample item difficulties $d \sim \mathcal{N}(1, 1)$ and $\log \tilde{d} \sim \mathcal{N}(1, 1)$, sample delay coefficient $\log r \sim \mathcal{N}(0, 0.01)$, set window coefficients $\theta_{2w} = \theta_{2w-1} = 1/\sqrt{W - w + 1}$, and number of windows $W = 5$. We ran TRPO with a batch size of 4000, discount rate $\gamma = 0.99$, and step size 0.01. The recurrent neural network policy uses a hidden layer of 32 units.

Analysis. The results in Figure 1 show that TRPO, an off-the-shelf policy gradient algorithm with minimal hyperparameter tuning, is competitive with complex heuristic schedulers like LEITNER and SUPERMNEMO. TRPO performs better than all the baselines on the GPL student model, and has mixed results on the EFC and HLR student models. The fact that TRPO generally performs worse than THRESHOLD is unsurprising, since THRESHOLD has unfair access to the latent parameters of the student. It is surprising that TRPO doesn't always learn to perform at least as well as SUPERMNEMO and LEITNER on the EFC and HLR students. It's possible that additional hyperparameter and policy architecture tuning could improve the performance of TRPO, since it can in principle learn arbitrary teaching policies.

Code for reproducing these results is publicly available at `https://github.com/rddy/deeptutor`.

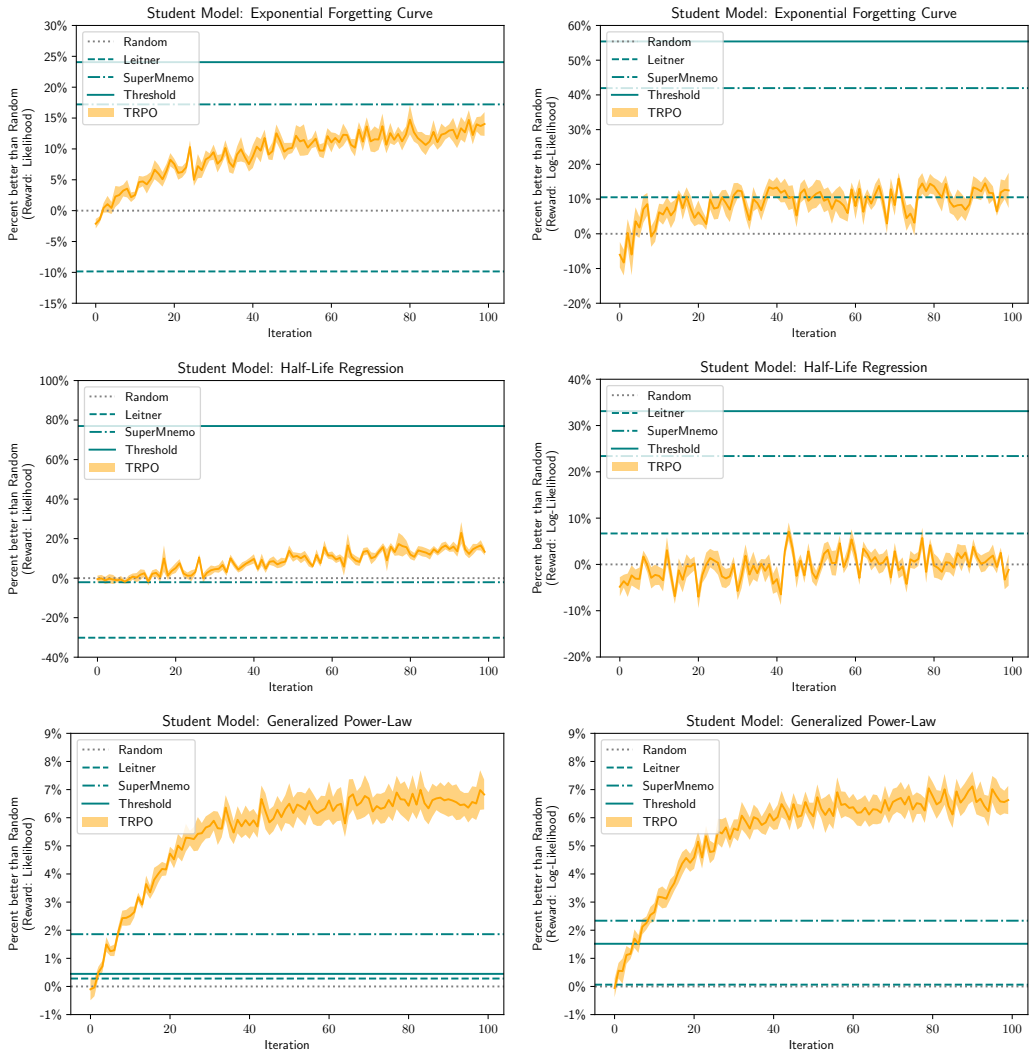

Figure 1: The left column corresponds to the reward function in Equation 4, and the right column to Equation 5. The first row corresponds to the EFC student model, the second to HLR, and the third to GPL. The red colored bands show the standard error of rewards over ten different random settings of simulator parameters. Standard error for the baselines is negligibly small, hence the lack of error bars for the dashed lines.

6 Discussion

Summary. In this paper, we study a flexible and scalable review scheduling algorithm for spaced repetition software that uses model-free reinforcement learning with neural network function approximation. Preliminary experiments on simulated students demonstrate the potential for this method to outperform widely-used heuristics like SuperMemo and the Leitner system.

6.1 Limitations and Future Work

There are several directions in which to proceed with addressing the limitations of this paper. The most critical follow-up work is to run user studies with real students. To avoid a long training phase during which the policy is serving suboptimal item recommendations to real users, it may be helpful to use an intelligent initialization scheme, e.g., pre-training the policy in simulation. One could initialize the teaching policy through imitation learning on log data collected from scheduling algorithms that have previously been deployed, e.g., the default variant of SuperMemo (SM3) used in

Mnemosyne, then fine-tune the policy with reinforcement learning in simulation [13, 33]. To improve the realism of the simulator, we could estimate the latent parameters of the student model using log data collected from real students, e.g., from Mnemosyne [27] and Duolingo [29]. Finally, to increase our confidence in an agent trained with a student simulator, one could use off-policy evaluation [32] to assess the performance of the agent on historical log data from real students before deploying on a live platform.

One challenge faced by scheduling algorithms in the wild that we have ignored in our laboratory setting is that students frequently create new items and decrease the priority of learning older items. A policy architecture that operates on a dynamic set of items and non-stationary reward function would be helpful. One possibility is a potential function that assigns a scalar 'priority' to an item based on its study history, and selects an item to show the student by taking a softmax over item priorities. This architecture has the added benefit of permutation-invariance.

Binary recall data contains very limited information about item difficulties and similarities. When rich features for item text, formulas, and images are available (as they are for a subset of the Mnemosyne data set), it makes sense to incorporate them into the agent's observations. Neural network policies are capable of operating on high-dimensional inputs, so DRL may be particularly suited to leveraging content features in spaced repetition systems.

Acknowledgments

We thank Igor Labutov, Siddhartha Banerjee, Thorsten Joachims, and Kevin Wilson for helpful discussions about spaced repetition. This work was supported in part by a Berkeley EECS Department Fellowship for first-year Ph.D. students.

References

[1] The mnemosyne project. `http://mnemosyne-proj.org`, 2006.

[2] Spaced repetition. `http://www.gwern.net/Spaced%20repetition`, 2016.

[3] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. A brief survey of deep reinforcement learning. *arXiv preprint arXiv:1708.05866*, 2017.

[4] Robert A Bjork. Memory and metamemory considerations in the. *Metacognition: Knowing about knowing*, page 185, 1994.

[5] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[6] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[7] 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.

[8] Frank N Dempster. Spacing effects and their implications for theory and practice. *Educational Psychology Review*, 1(4):309–330, 1989.

[9] Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning*, pages 1329–1338, 2016.

[10] Hermann Ebbinghaus. *Memory: A contribution to experimental psychology*. Number 3. University Microfilms, 1913.

[11] Chaitanya Ekanadham and Yan Karklin. T-skirt: Online estimation of student proficiency in an adaptive learning system. *Machine Learning for Education Workshop at ICML*, 2015.

[12] Damien Elmes. Anki. `http://ankisrs.net`, 2015.

[13] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Andrew Sendonaris, Gabriel Dulac-Arnold, Ian Osband, John Agapiou, et al. Learning from demonstrations for real world reinforcement learning. *arXiv preprint arXiv:1704.03732*, 2017.

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

[15] Andrew S Lan, Christoph Studer, and Richard G Baraniuk. Time-varying learning and content analytics via sparse factor analysis. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 452–461. ACM, 2014.

[16] Sebastian Leitner. *So lernt man lernen*. Herder, 1974.

[17] 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.

[18] Michael C Mozer and Robert V Lindsey. Predicting and improving memory retention: Psychological theory matters in the big data era, 2016.

[19] Timothy P Novikoff, Jon M Kleinberg, and Steven H Strogatz. Education of a model student. *Proceedings of the National Academy of Sciences*, 109(6):1868–1873, 2012.

[20] Kaustubh R Patil, Xiaojin Zhu, Łukasz Kopeć, and Bradley C Love. Optimal teaching for limited-capacity human learners. In *Advances in neural information processing systems*, pages 2465–2473, 2014.

[21] Philip I Pavlik Jr, Hao Cen, and Kenneth R Koedinger. Performance factors analysis–a new alternative to knowledge tracing. *Online Submission*, 2009.

[22] Chris Piech, Jonathan Bassen, Jonathan Huang, Surya Ganguli, Mehran Sahami, Leonidas J Guibas, and Jascha Sohl-Dickstein. Deep knowledge tracing. In *Advances in Neural Information Processing Systems*, pages 505–513, 2015.

[23] Christopher James Piech. *Uncovering Patterns in Student Work: Machine Learning to Understand Human Learning*. PhD thesis, Stanford University, 2016.

[24] Paul Pimsleur. A memory schedule. *Modern Language Journal*, pages 73–75, 1967.

[25] Anna N Rafferty, Emma Brunskill, Thomas L Griffiths, and Patrick Shafto. Faster teaching via pomdp planning. *Cognitive science*, 40(6):1290–1332, 2016.

[26] Georg Rasch. *Probabilistic models for some intelligence and attainment tests.* ERIC, 1993.

[27] S. Reddy, I. Labutov, S. Banerjee, and T. Joachims. Unbounded human learning: Optimal scheduling for spaced repetition. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2016.

[28] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 1889–1897, 2015.

[29] Burr Settles and Brendan Meeder. A trainable spaced repetition model for language learning. In *ACL (1)*, 2016.

[30] Adish Singla, Ilija Bogunovic, Gábor Bartók, Amin Karbasi, and Andreas Krause. Near-optimally teaching the crowd to classify. In *ICML*, pages 154–162, 2014.

[31] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[32] Philip S Thomas and Emma Brunskill. Data-Efficient Off-Policy Policy Evaluation for Reinforcement Learning.

[33] Matej Večerík, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817*, 2017.

[34] Jacob Whitehill and Javier Movellan. Approximately optimal teaching of approximately optimal learners. *IEEE Transactions on Learning Technologies*, 2017.

[35] Kevin H Wilson, Yan Karklin, Bojian Han, and Chaitanya Ekanadham. Back to the basics: Bayesian extensions of irt outperform neural networks for proficiency estimation. *arXiv preprint arXiv:1604.02336*, 2016.

[36] John T Wixted and Shana K Carpenter. The wickelgren power law and the ebbinghaus savings function. *Psychological Science*, 18(2):133–134, 2007.

[37] PA Wozniak and Edward J Gorzelanczyk. Optimization of repetition spacing in the practice of learning. *Acta neurobiologiae experimentalis*, 54:59–59, 1994.

[38] Xiaojin Zhu. Machine teaching: An inverse problem to machine learning and an approach toward optimal education. In *AAAI*, pages 4083–4087, 2015.