Background and Related Work

sugandh@kth.se

April 2018

1 Field History

Machine learning has become increasingly popular in the recent years. Over the years, the area of machine learning has had some rapid progress. And now, with the advent of deep learning, it has become the focus of research when it comes to the field of Artificial Intelligence as artificial neural networks try to mimic the activity of the neurons in the human brain. Even though the idea is very old which could be traced back to as early as 1943 in one of the seminal papers of McCulloch and Pitts [22], it only recently became feasible to implement these complex networks on large datasets because of the advancements in the hardware. Since, their inception, deep learning has now found its way in almost every area from image classification to drug discovery.

In a broader sense, we can consider learning algorithms to fall into three main categories based on the feedback that they receive from the environment. On one extreme, there is supervised learning in which the algorithms are presented with a target value that they check after each iteration to set the value of hyperparameters to an optimum value such that the error is minimized. On the other extreme, there is unsupervised algorithms where no feedback is provided by the environment to the learning algorithms and the algorithm has to figure out the parameters using the patterns found in the features that are provided as input. Between these two extremes lies reinforcement learning.

One of the most notable characteristics of humans and animals has always been their ability to learn from their experience by interacting with their environment. Interacting with the environment provides a great deal of information such as cause and effect, actions needed to achieve a goal and ramifications of an action. Most of the actions that we take are taken by keeping into account the goal that we are trying to achieve and the kind of responses that we expect that we will receive when we take certain actions. Reinforcement learning is one such approach in the paradigm of learning systems which focuses on goal directed learning as compared to other machine learning algorithms [37].

Reinforcement learning is fast becoming another hot topic of research, one of the major reason for this could be attributed to the pioneer work of Silver et.

al on Alpha Go Zero [34]. The team behind it started off with AplhaGo [33], the first program ever to defeat the world champion in the game of Go. Now, they have gone a step further than that in the sense that AlphaGo Zero does not even need the training data from thousands of game to learn to play. It does that by learning to play against itself and is much faster in learning to become an expert.

One of the most earliest work which could be said to have formed the basis of reinforcement learning is the work by psychologist Edward L. Thorndike in which he developed his law of effect [38]. This principle states that the responses which lead to pleasant outcomes are more likely to occur in similar situations while the responses that create unpleasant outcomes are less likely to occur.

One of the most challenging aspect of reinforcement learning is the huge amount of data and computational power it needs as well as the reproducability of research results [16]. So, currently, reinforcement learning, mostly finds it applications in non-mission critical applications of sequential decision making problems. But, despite all these challenges reinforcement learning is starting to make its impact and is being used now in robotics, recommending online content, in the field of medicine for optimal medication dosing [27] and recommending use of medical tools [24]; it is even being used now in tuning neural network [6], [11].

Although human tutoring has always been the de facto standard when it comes to teaching, over the past few years e-learning has become massively popular because of their ease of access that allows users to learn anywhere at anytime. For e.g. - MOOCs like Coursera, EdX, etc. and learning applications like Duolingo, Quizzlet, etc. At first, most of the learning systems just informed users of the wrong answers without considering anything of the learning process (e.g.,[4],[36]). These first generation systems were called CAI short for computer-assisted instruction tutors. These traditional systems sometimes are also referred to as Computer-Based Instruction (CBI), Computer Aided Learning(CAL), and Computer-Based Training(CBT). But soon, researchers started to look into systems with the intent of getting more ‘intelligent’ approach to learning that would oversee the learning process(e.g.,[12], [15], [35]).

As described by Shute [31], “A system must behave intelligently, not actually be intelligent, like a human”. These systems were the second generation systems and were called ITS (Intelligent Tutoring systems).

As mentioned in [1], VanLehn [39] has summarized common beliefs about the effectiveness of different types of tutoring. According to VanLehn, CAI tutors are generally believed to boost examination scores by 0.3 standard deviations over usual levels, or from the 50th to the 62nd percentile. ITSs are thought to be more effective, raising test performance by about 1 standard deviation, or from the 50th to the 84th percentile. Human tutors are thought to be most effective of all, raising test scores by 2 standard deviations, or from the 50th

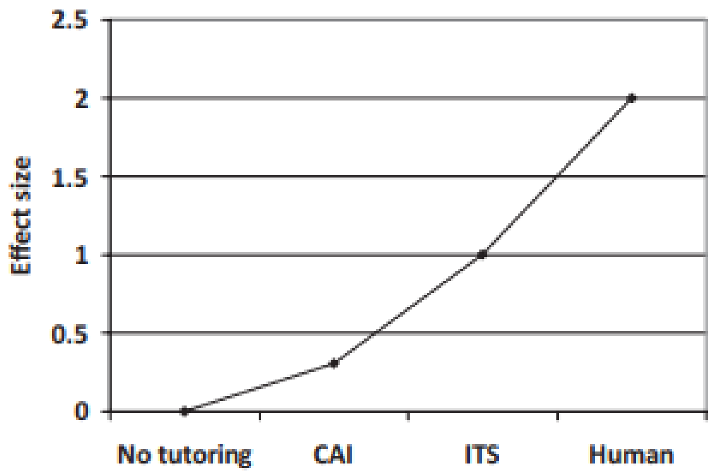

Figure 1: Common belief about effect sizes of types of tutoring. Image from Kurt Vanlehn, *The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems.* in Educational Psychologist, vol. 46, p 197–221

to the 98th percentile. Fig. 1 shows this trend among different types of tutoring.

Research in ITS is a very challenging task as it is an interdisciplinary field combining AI, Cognitive psychology and educational theory. Fig. 2 captures the relation between various fields and ITS. Because of this interdisciplinary nature, the research goals, terminology, theoretical frameworks, and emphases amongst researchers vary a lot. Although ITS have not been widely adapted yet but the research in ITS combined with machine learning is bound to grow even more.

2 Related Theory

2.1 Intelligent Tutoring Systems

ITS, short for Intelligent Tutoring Systems are systems that have been developed with the intention to teach students. As mentioned in [25], the architectures of ITS can vary a lot. But earlier studies identified 3 core modules present in all ITS [7], [8]

- The expert knowledge module.
- The student model module.

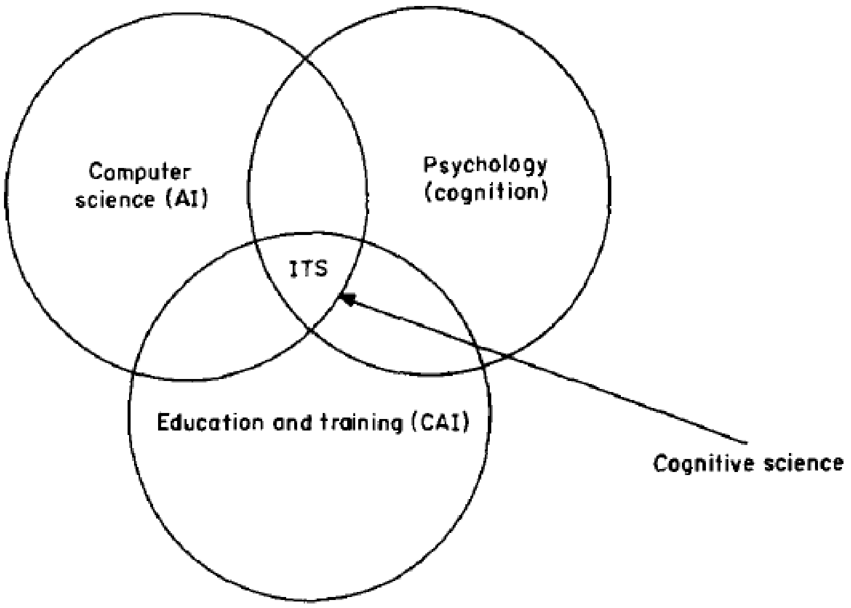

Figure 2: ITS Domains. Image from Hyacinth S. Nwana , *Intelligent Tutoring Systems: an overview* in Artificial Intelligence Review, vol. 4, p 251-277

- The tutoring module.

But, more recent studies made researchers to agree on a fourth module as well.[40], [21], [9]

- The expert knowledge module.

Fig. 3 shows the general ITS architecture. The **expert module** represents the source of the knowledge which is taught to the students. This module should be able to generate questions, answers and sometimes, even the steps to solve a problem.

The **student model** module refers to the ever-changing representation of the skills and knowledge that the student possess. In order to adapt the tutor module to the respective needs of student, it is important that the tutor module gets the current skills and knowledge level of student.

The **tutor module**, sometimes also called as the teaching strategy, guides the student through the learning process. This module is responsible for which lesson to present to the student and when to present it.

The **user interface** module is a bidirectional communication channel through which the student and the system interact. How the information is presented

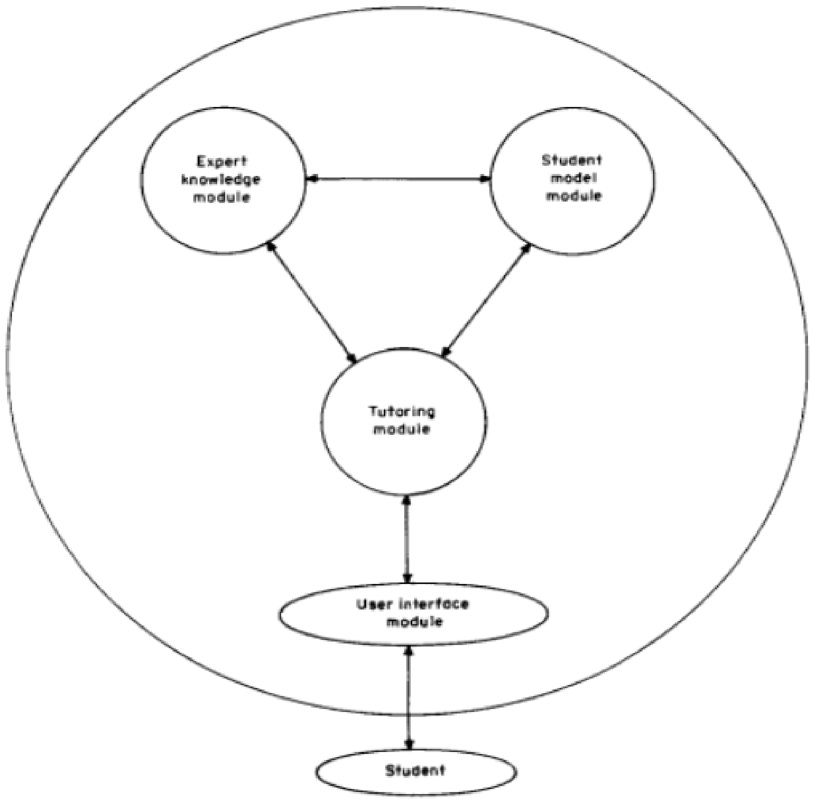

Figure 3: General ITS Architecture. Image from Hyacinth S. Nwana , *Intelligent Tutoring Systems: an overview* in Artificial Intelligence Review, vol. 4, p 251-277

to the student can easily affect the effectiveness of the entire system.

2.2 Reinforcement Learning

As described by Sutton and Barto [37], reinforcement learning can be described as a Markov Decision Process consisting of:

- A set of states $\mathcal{S}$.
- A set of actions $\mathcal{A}$.
- A set of state-transition probability distribution, $p : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$,

$$p(s' \mid s, a) \doteq Pr\{S_t = s' \mid S_{t-1} = s, A_{t-1} = a\} = \sum_{r \in \mathcal{R}} p(s', r \mid s, a$$

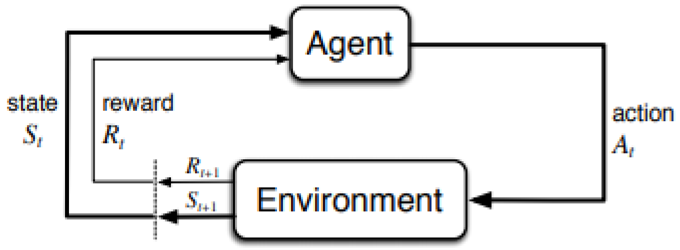

Figure 4: The agent–environment interaction in a Markov decision process. Image from Richard S. Sutton and Andrew G. Barto, 1998, *Introduction to Reinforcement Learning (1st ed.)*, MIT Press, Cambridge, MA, USA.

- An immediate reward function that gives either the expected rewards for state–action pairs as a two argument function $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$

$$r(s,a) \doteq \mathbb{E}[R_t \mid S_{t-1} = s, A_{t-1} = a] = \sum_{r \in R} r \sum_{s' \in S} p(s', r \mid s, a),$$

  or gives the expected rewards for state-action-next-state triples as a three-argument function $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$,

$$r(s,a,s') \doteq \mathbb{E}[R_t \mid S_{t-1} = s, A_{t-1} = a, S_t = s'] = \sum_{r \in R} r \frac{p(s', r \mid s, a)}{p(s' \mid s, a)}.$$

- A discount factor, $\gamma \in (0, 1)$.
- Sometimes, a distribution of initial state s_0 is also given $\rho_0 : \mathcal{S} \rightarrow \mathbb{R}$

2.2.1 Elements of Reinforcement Learning

2.2.1.1 Agent and Environment

An agent is an entity that interacts with its environment to achieve its goal by learning and taking appropriate actions. Everything surrounding the agent comprises the environment. The agent must be able to sense the state of the environment in order to take appropriate action which would affect the state of the environment. At each time step, the agent interacts with the environment. Fig. 4 shows the process of interaction between an agent and its environment in an MDP. MDPs usually include three aspects - sensation, actions and goals. Because of limiting itself o just these three aspects, MDPs may not be sufficient to solve all decision-learning problems.

2.2.1.2 Policy

A policy defines how the agent will act in a particular state of the environment. Policies may be stochastic. It is simply a mapping from states to probabilities of selecting each possible action It can be viewed as stimulus-response rules in biological systems.

If the agent is following policy π at time t, then $\pi(a \mid s)$ is the probability that $A_t = a$ if $S_t = s$.

2.2.1.3 Reward signal

A reward is a number which is sent to the agent from the environment after each time step. The goal of the agent in a reinforcement learning problem is to maximize the total reward accrued over time. The reward signal determines whether an event is good or bad for the agent. if the reward received by the agent after performing an action,An selected by the policy, was a low reward then the policy may be changed.

2.2.1.4 Episodes and Returns

If an agent-environment interaction can be naturally broken into subsequences then we call these subsquence as episodes and such tasks are called episodic tasks. Each episode ends in a special state called the terminal state. If the agent-environment interaction can not be naturally broken into episodes then these tasks are called continuing tasks.

The expected return, G_t, is defined as the sum of rewards. If $R_{t+1}, R_{t+2}, R_{t+3}, ...,$ denotes the sequence of rewards obtained after time step t then expected return, G_t, is given by:

$$G_t \doteq R_{t+1} + R_{t+2} + R_{t+3} + ... + R_t.$$

When discounting is used, the agent select actions to maximize the sum of the discounted rewards it receives over the future.

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

where γ is a parameter, $0 \leq \gamma \leq 1$, called the discount rate.

Calculating return for continuing tasks is challenging because the final time step in this case would be T = ∞, and the return could become infinite as well.

2.2.1.5 Value function

A value of a state is defined as the total reward that is accrued by the agent when starting from that particular state and the states that follow. The value function simply determines the goodness of a state for an agent. In contrast to reward signal, which defines the desirability of an action in an immediate context, the value function describes the desirability of an action on a long term basis. For instance, an action might give a very low immediate reward but in the long run, it may produce a high value.

The value of a state s under a policy π, denoted by $v_\pi(s)$, is the expected return when starting in s and following π thereafter. For MDPs, v_π is given by

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s\right], \forall s \in S$$

where $\mathbb{E}_\pi[\cdot]$ denotes the expected value of a random variable given that the agent follows policy π, and t is any time step. The value of the terminal state is always zero. v_π is the state-value function for policy π.

Similarly, the value of taking action a in state s under a policy π, denoted $q_\pi(s,a)$, as the expected return starting from s, taking the action a, and thereafter following policy π is given by:

$$q_\pi(s,a) \doteq \mathbb{E}_\pi[G_t \mid, A_t = a] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a\right].$$

q_π is the action-value function for policy π.

2.2.1.6 Model of the environment (optional):

A model of the environment imitates the the environment and its behaviour in which the agent is going to perform. It is optional as there are some reinforcement learning algorithms that are model free and use trial and error methods because it is not always possible to build a model of the environment.

2.2.2 Rewards, Returns and Value functions

Rewards are essential component of reinforcement learning as without it, it is impossible to estimate value. After every time step the action selected by the agent should be the action with the highest value not the highest reward. Also, it is easy to estimate rewards as they are given directly by the environment while estimating value is a very challenging task because they are calculated based on the actions that the agent takes and then re-calculated again after each time step.

While return gives the expected discounted sum of rewards for one episode, a value function gives the expected discounted sum of rewards from a certain state.

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s]$$

2.2.3 Exploration/Exploitation trade-off

One of the key challenges in reinforcement learning is keeping a balance between letting agent use an action that it has used previously and has found it to be effective (exploitation) versus using a new randomly selected action and determine its effectiveness in that particular situation (exploration).

An agent that only performs exploitation can be thought of as using Greedy algorithm and it selects action with the highest value. In this case the action is selected based on the following expression,

$$a_t^* = argmax_{a \in A} Q_t(a),$$

where a^* is the selected action at time step t , $argmax_a$ denotes the action a for which the expression that follows is maximized, $Q_t(a)$ is the mean reward of action a at time step t and finally $Q(a)$ is given as:

$$Q(a) = \mathbb{E}[r \mid a],$$

An agent using only Greedy algorithm can behave sub-optimally forever. On the other hand, an agent that only performs exploration never uses the knowledge that it has gained over time.

One of the simplest algorithms that can be used for solving exploitation/exploration trade-off is the ϵ-Greedy algorithm which allows agent to select a random action with small probability ϵ and with probability 1 - ϵ use the action that the agent knows that was effective previously.

2.3 Recurrent Neural Networks

RNNs are mostly used for tasks that involve sequential inputs such as speech and language. One of the major problems with traditional neural networks is that they do not have persistence, they need to be trained from scratch every time. They do not consider the dependence of inputs and/or outputs to each other. This issue is addressed by RNNs as they use history to predict future outcomes. They are network with loops in it so that the information can flow from one step of the network to the other, making it persistent.

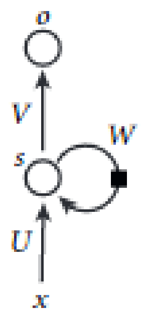

Figure 5: High level representation of RNN. Image from LeCun et. al. *Deep learning* in Nature, vol. 521

Fig. 5 shows a general high level representation of an RNN. Loops in the RNN can be expanded in a chain like architecture which can be seen as a very deep feedforward networks so that information from one step of the network can be fed into the next step and so on. Fig. **??** shows an unfolded RNN architecture.

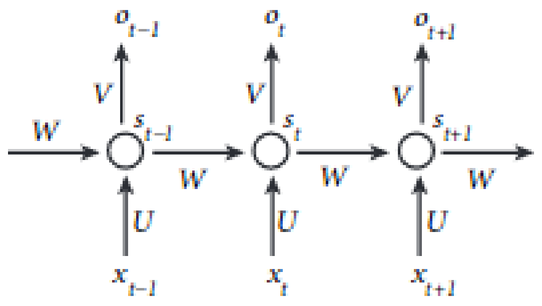

Figure 6: A typical unfolded representation of RNN. Image from LeCun et. al. *Deep learning* in Nature, vol. 521

Here, x_t represents the input at time step t, s_t is the hidden state at time step t which can be assumed to be memory of the network which contains information about the history of all the past elements of the sequence and is calculated based on the previous hidden state and the input at the current step: $s_t = f(Ux_t + Ws_{t-1} + b_s)$;the function f usually is a non linear activation function such as $tanh$ or $ReLU$ and b_s is the bias for latent units, o_t is the output at time step t and may be calculated as $o_t = \sigma(Vs_t + b_o)$ where b_o is the bias for the readout unit. The same parameters (matrices U, V and W) are used at each time step.

RNNs are different from feedforward networks as the RNNs have the ability to make use of sequence of inputs by using their memory. These sequences can vary in size and RNNs can adapt to them dynamically. Also, since the RNNs have output at each time step, there is a cost function associated with the output of each time step as compared to feedforward network where the cost function is applied to its final output.

2.4 Spaced Repetition

Kang [18] states that having the initial study and subsequent review or practice be spaced out over time generally leads to superior learning than having the repetition(s) occur in close temporal succession (with total study time kept equal in both cases). This phenomenon is called the spacing effect and this technique is referred to as spaced repetition. Fig. 7 shows how the knowledge learned is forgotten over time.

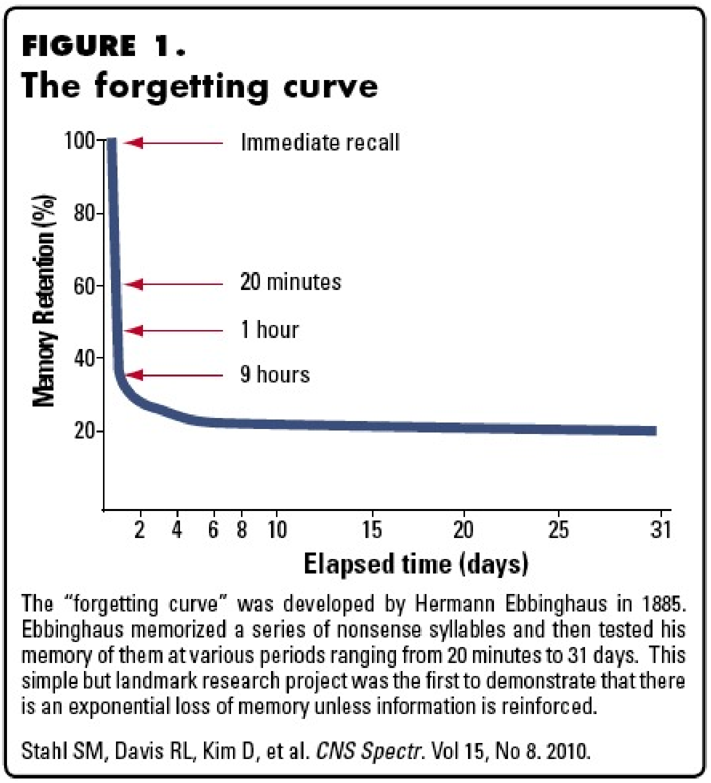

Figure 7: Graphical representation of forgetting curve. Image from *http://www.gwern.net/Spaced-repetition*

2.5 Relation between Tutoring Systems and Student learning

Capturing a student's knowledge level is a very challenging task but at the same time it is also an imperative task that needs to be done when implementing an intelligent tutoring system. In order to tailor a good set of recommended questions to students, a system should be able to determine the current level of student's knowledge and also, should be able to predict the likelihood of student answering the questions, presented by the system, correctly and in the process also modify the level of student's knowledge. The system presents a question to the system and based on the student's answer receives a corrective feedback. Duration and the manner in which the material was learned is another factor that highly affects the estimation of student's current level of knowledge [20]. Individual differences among students also vary which is yet another factor that supports the reason for estimating the knowledge level of students.

2.6 Models of human memory

2.6.1 Exponential Forgetting Curve:

Ebbinghaus's [14]classic study on forgetting learned materials states that when a person learns something new, most of it is forgotten in an exponential rate within the first couple of days and after that the rate of loss gradually becomes weaker.
Reddy et al. [32] give the probability of recalling an item as:

$$P[recall] = exp(-\theta \cdot d/s), \tag{1}$$

where θ is the item difficulty, d is the time elapsed since the material was last reviewed and s is the memory strength.

2.6.2 Half life regression:

As described by Settles and Meeder [30], the memory decays exponentially over time:

$$p = 2^{\Delta/h}, \tag{2}$$

In this equation, p denotes the probability of correctly recalling an item (e.g., a word), which is a function of Δ, the lag time since the item was last practiced, and h, the half-life or measure of strength in the learner's long-term memory. When $\Delta = h$, the lag time is equal to the half-life, so $p = 2^{-1} = 0.5$, and the student is on the verge of being unable to remember.
Assuming that the half-life should increase exponentially with each repeated exposure. The estimated half life $\hat{h}_\Theta$ is given by

$$\hat{h}_\Theta = 2^{\Theta \Delta x}, \tag{3}$$

where x is a feature vector that describes the study history for the student-item pair and the vector Θ contain weights that correspond to each feature variable in x.

2.6.3 Generalized power law:

Wixted and Carpenter [41] state that the probability of recalling decays according to a generalized power law as a function of t:

$$P[recall] = \lambda(1 + \beta.t)^{-\Psi}, \tag{4}$$

where t is the retention interval, λ is a constant representing the degree of initial learning, β is a scaling factor on time ($h > 0$) and Ψ represents the rate of forgetting.

2.7 Leitner system

Leitner System [19] is one of the most widely used method used in flashcards for spaced repetition. The basic idea behind this technique is to make users interact more with items that they are likely to forget and let them spend less time on items that they are able to recall efficiently. Fig. 8 shows the working of Leitner system. This system manages a set of n decks. When the user sees an item for the first time then that item is placed in deck 1. Afterwards, when the user sees an item placed in deck i and recalls it correctly, the item is moved to the bottom of deck i+1. However, if the user answers incorrectly then it is moved to the bottom of deck i-1. The goal of the Leitner system is to make user spend more time on lower decks which contains the items that the user is not able to recall efficiently. [32] uses a Queue based Leitner system to generate a

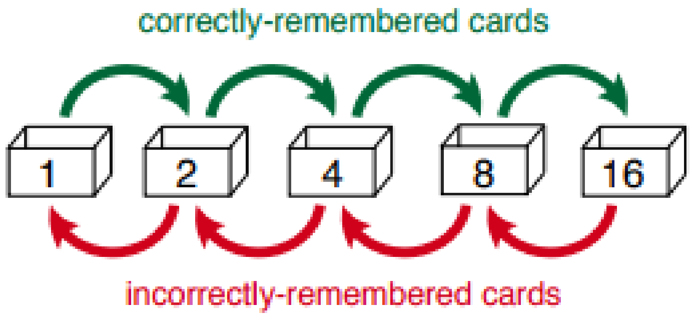

Figure 8: Leitner System. Image from Settles and Meeder, *A Trainable Spaced Repetition Model for Language Learning* in ACL 2016

mathematical model for spaced repetition system.

2.8 Supermemo System

There have been different versions of SuperMemo or SM algorithms. The first computer based SM algorithm was SM 2 and since then there have been multiple revisions. The SM 2 algorithm as described by Wozniak [5] is described as below:

1. Split the knowledge into smallest possible items.
2. With all items associate an E-Factor equal to 2.5.
3. Repeat items using the following intervals:
   I(1):=1
   I(2):=6
   for n>2: I(n):=I(n-1)*EF
   where:
   I(n) - inter-repetition interval after the n-th repetition (in days),
   EF - E-Factor of a given item which is easiness factor reflecting the easiness

of memorizing and retaining a given item in memory
If interval is a fraction, round it up to the nearest integer.

4. After each repetition assess the quality of repetition response in 0-5 grade scale:
   5 - perfect response
   4 - correct response after a hesitation
   3 - correct response recalled with serious difficulty
   2 - incorrect response; where the correct one seemed easy to recall
   1 - incorrect response; the correct one remembered
   0 - complete blackout.

5. After each repetition modify the E-Factor of the recently repeated item according to the formula:
   EF':=EF+(0.1-(5-q)*(0.08+(5-q)*0.02))
   where:
   EF' - new value of the E-Factor,
   EF - old value of the E-Factor,
   q - quality of the response in the 0-5 grade scale.
   If EF is less than 1.3 then let EF be 1.3.

6. If the quality response was lower than 3 then start repetitions for the item from the beginning without changing the E-Factor (i.e. use intervals I(1), I(2) etc. as if the item was memorized anew).

7. After each repetition session of a given day repeat again all items that scored below four in the quality assessment. Continue the repetitions until all of these items score at least four.

2.9 Trust Region Policy Optimization

TRPO has been developed by Schulman et. al [29] and is an iterative procedure that gives guaranteed monotonic improvements for optimizing policies. Given an MDP defined by the tuple $(\mathcal{S}, \mathcal{A}, P, r, \gamma, \rho_0)$ where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $\mathcal{P}$ is the initial state-transition probability, r is the reward function, γ is the discount factor and ρ_0 is the initial state distribution. Let π be a stochastic policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$) and let $\eta(\pi)$ be its expected discounted reward:

$$\eta(\pi) = \mathbb{E}_{s_0,a_0,\ldots}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t)\right],$$

where $s_0 \sim \rho_0(s_0), a_t \sim \pi(a_t|s_t), s_{t+1} \sim P(s_{t+1}|s_t, a_t)$.

Let Q_π be the state-action value function:

$$Q_\pi(s_t, a_t) = \mathbb{E}_{s_{t+1},a_{t+1},\ldots}\left[\sum_{l=0}^{\infty} \gamma^l r(s_{t+l})\right].$$

V_π be the value function :

$$V_\pi(s_t) = \mathbb{E}_{a_t, s_{t+1},, \ldots} \left[\sum_{l=0}^{\infty} \gamma^l r(s_{t+l}) \right].$$

and let A_π be the advantage function:

$$A_\pi(s, a) = Q_\pi(s, a) - V_\pi(s),$$

where $a_t \sim \pi(s_t \mid a_t), s_{t+1} \sim P(s_{t+1} \mid s_t, a_t)$ for $t \geq 0$.

Now, advantage of another policy $\hat{\pi}$ can be written in terms of advantage over the first policy:

$$\eta(\hat{\pi}) = \eta(\pi) + \mathbb{E}_{s_0, a_0, \ldots \sim \hat{\pi}} \left[\sum_{t=0}^{\infty} \gamma^t A_\pi(s_t, a_t) \right], \tag{5}$$

where $\mathbb{E}_{s_0, a_0, \ldots \sim \hat{\pi}}[\ldots]$ indicates that the actions are sampled $a_t \sim \hat{\pi}(\cdot \mid s_t)$ Let ρ_π be the discounted visitation frequencies

$$\rho_\pi(s) = P(s_0 = s) + \gamma P(s_1 = s) + \gamma^2 P(s_2 = s) + \ldots = \sum_{t=0}^{\infty} \gamma^t P(s_t = s \mid \pi)$$

where $s_0 \sim \rho_0$ and the actions are chosen according to π.

Now eq. 5 can be rewritten as sum over states instead of time step:

$$\begin{aligned}
\eta(\hat{\pi}) &= \eta(\pi) + \sum_{t=0}^{\infty} \sum_{s} P(s_t = s \mid \hat{\pi}) \sum_{a} \hat{\pi}(a \mid s) \gamma^t A_\pi(s, a) \\
&= \eta(\pi) + \sum_{s} \sum_{t=0}^{\infty} \gamma^t P(s_t = s \mid \hat{\pi}) \sum_{a} \hat{\pi}(a \mid s) A_\pi(s, a) \\
&= \eta(\pi) + \sum_{s} \rho_{\hat{\pi}}(s) \sum_{a} \hat{\pi}(a \mid s) A_\pi(s, a) \qquad (6)
\end{aligned}$$

Eq. 6 implies that any policy update with non-negative expected advantage at every state i.e. $\sum_a \hat{\pi}(a \mid s) A_\pi(s, a) \geq 0$ is guaranteed to increase the policy performance η or leave it constant if the expected advantage is zero everywhere. However, in the approximate setting, there will be some states with negative expected advantage. Hence, following approximation to η is used:

$$L_\pi(\hat{\pi}) = \eta(\pi) + \sum_{s} \rho_\pi(s) \sum_{a} \hat{\pi}(a \mid s) A_\pi(s, a) \tag{7}$$

If there is a parameterized policy π_θ where $\pi_\theta(a|s)$ which is differentiable w.r.t θ then L_π matches η to first order [17]. For parameter θ_0

$$L_{\pi_{\theta_0}}(\pi_{\theta_0}) = \eta(\pi_{\theta_0}), \tag{8}$$

$$\nabla_\theta L_{\pi_{\theta_0}}(\pi_{\theta_0}) \mid_{\theta=\theta_0} = \nabla_\theta \eta(\pi_{\theta_0}) \mid_{\theta=\theta_0} \tag{9}$$

Eq. 8 shows that small steps improve $L_{\pi_{\theta_{old}}}$ and in turn η but it does not tell what should be the size of the step. To address this, Kakade and Langford [17] proposed conservative policy iteration that provides explicit lower bounds on the improvement of η.

Let η_{old} denote the current policy, and let $\pi^{'} = argmax_{\pi'} L_{pi_{old}}(\pi')$. The new policy π_{new} is defined to be the following mixture:

$$\pi_{new}(a|s) = (1-\alpha)\pi_{old}(a|s) + \alpha\pi^{'}(a|s). \tag{10}$$

Kakade and Langford derived the following lower bound:

$$\eta(\pi_{new}) \geq L_{\pi_{old}}(\pi_{new}) - \frac{2\epsilon\gamma}{(1-\gamma)^2}\alpha^2, \text{where } \epsilon = \max_s \mid \mathbb{E}_{a\sim\pi'(a|s)}[A_\pi(s,a)] \mid . \tag{11}$$

For general stochastic policies eq. 11 can be extended by replacing α with a distance measure between π and $\pi^{'}$ and changing ϵ.

If distance measure used is total variation divergence defined by $D_{TV}(p \mid\mid q) = \frac{1}{2}\sum_i \mid p_i - q_i \mid$ for discrete probability distributions p, q then

$$D_{TV}^{max} = \max_s D_{TV}(\pi(\cdot \mid s) \mid\mid \hat{\pi}(\cdot \mid s)) \tag{12}$$

If $\alpha = D_{TV}^{max}(\pi_{old}, \pi_{new})$, then the following bound holds:

$$\eta(\pi_{new}) \geq L_{\pi_{old}}(\pi_{new}) - \frac{4\epsilon\gamma}{(1-\gamma)^2}\alpha^2, \text{where } \epsilon = \max_{s,a} A_\pi(s,a) \mid . \tag{13}$$

Now, total variation divergence relates to the KL divergence as $D_{TV}(p \mid\mid q)^2 \leq D_{KL}(p \mid\mid q)$. Let $D_{KL}^{max} = \max_s D_{KL}(\pi(\cdot \mid s) \mid\mid \hat{\pi}(\cdot \mid s))$ then the following bound holds:

$$\eta(\pi_{new}) \geq L_{\pi_{old}}(\pi_{new}) - CD_{KL}^{max}(\pi_{old}, \pi_{new}), \text{where } C = \frac{4\epsilon\gamma}{(1-\gamma)^2}. \tag{14}$$

Algorithm 1: Policy iteration algorithm guaranteeing non-decreasing expected return η

```
Initialize $\pi_0$.;
**for** $i = 0, 1, 2, ...$ *until convergence* **do**
    Compute all advantage values $A_{\pi_i}(s,a)$.;
    Solve the constrained optimization problem;
    $\pi_{i+1} = \underset{\pi}{\text{argmax}}\,[L_{\pi_i}(\pi) - CD_{KL}^{max}(\pi_i, \pi)]$;
    where$C = \frac{4\epsilon\gamma}{(1-\gamma)^2}$ ;
    and $L_{\pi_i}(\pi) = \eta(\pi_i) + \sum_s \rho_{\pi_i}(s) \sum_a \hat{\pi}(a \mid s) A_{\pi_i}(s,a)$;
**end**
```

Algorithm 1 is a type of maximization-minimization algorithm that guarantees to generate a monotonically improving sequence of policies. Maximizing $M_i(\pi) = L_{\pi_i}(\pi) - CD_{KL}^{max}(\pi_i, \pi)$ at each iteration guarantees that η is non-decreasing. Here, M_i becomes the surrogate function that minorizes η with equality at eta_i and guarantees to improve η. However, in practice, if the penalty coefficient C, mentioned above, is used then the step sizes would be very small. Since parameterized policies $\eta_\theta(a|s)$ with parameter vector θ were used; overload previous notations to use functions of θ instead of π. A trust region constraint is introduced that makes it possible to take larger steps in a robust way by using a constraint on KL divergence between the new policy and the old policy:

$$\begin{gathered}\underset{\theta}{\text{maximize}} L_{\theta_{old}}(\theta) \\ \text{subject to } D_{KL}^{max}(\theta_{old}, \theta)\end{gathered} \tag{15}$$

This problem imposes a constraint that the KL divergence is bounded at every point in the state space. While it is motivated by the theory, this problem is impractical to solve due to the large number of constraints. Instead, a heuristic approximation is used which considers the average KL divergence

$$\overline{D}_{KL}^{\rho}(\theta_1, \theta_2) \doteq \mathbb{E}_{s\sim\rho}[D_{KL}(\pi_{\theta_1}(\cdot \mid s) \mid\mid \pi_{\theta_2}(\cdot \mid s))]. \tag{16}$$

Following optimization problem can be solved to generate a policy update:

$$\begin{gathered}\underset{\theta}{\text{maximize}} L_{\theta_{old}}(\theta) \\ \text{subject to } \overline{D}_{KL}^{\rho_{\theta_{old}}}(\theta_{old}, \theta) \leq \delta\end{gathered} \tag{17}$$

The objective and constraint functions can be approximated using Monte Carlo simulation. Following optimization problem, obtained by expanding $L_{\theta_{old}}$ in Equation 17 needs to be solved :

$$\begin{gathered}\underset{\theta}{\text{maximize}} \sum_s \rho_{\theta_{old}}(s) \sum_a \pi_\theta(a \mid s) A_{\theta_{old}}(s, a) \\ \text{subject to } \overline{D}_{KL}^{\rho_{\theta_{old}}}(\theta_{old}, \theta) \leq \delta\end{gathered} \tag{18}$$

Replace $\sum_s \rho_{\theta_{old}}(s)[...]$ in the objective by the expectation $\frac{1}{1-\gamma}\mathbb{E}_{s\sim\rho_{\theta_{old}}}[...]$. Next, replace the advantage values $A_{\theta_{old}}$ by the Q-values $Q_{\theta_{old}}$. Last, replace the sum over the actions by an importance sampling estimator. Using q to denote the sampling distribution, the contribution of a single s_n to the loss function is

$$\sum_a \pi_\theta(a \mid s_n) A_{\theta_{old}}(s_n, a) = \mathbb{E}_{a\sim q}\left[\frac{\pi_\theta(a \mid s_n)}{q(a \mid s_n)} A_{\theta_{old}}(s_n, a)\right]. \tag{19}$$

Now, optimization problem in eq. 18 is equivalent to

$$\begin{gathered}\underset{\theta}{\text{maximize}} \mathbb{E}_{s\sim\rho_{\theta_{old}}, a\sim q}\left[\frac{\pi_\theta(a \mid s)}{q(a \mid s_n)} Q_{\theta_{old}}(s, a)\right] \\ \text{subject to } \mathbb{E}_{s\sim\rho_{\theta_{old}}}[D_{KL}(\pi_{\theta_{old}}(\cdot \mid s) \mid\mid \pi_\theta(\cdot \mid s))] \leq \delta.\end{gathered} \tag{20}$$

There are two sampling schemes for performing Monte Carlo estimation - *single path* and *vine*. In *single path* estimation procedure, a sequence of states is collected by sampling $s_0 \sim \rho_0$ and then the policy $\pi_{\theta_{old}}$ is simulated for some number of time steps to generate a trajectory $s_0, a_0, s_1, a_1, ..., s_{T-1}, a_{T-1}, s_T$. Hence, $q(a \mid s) = \pi_{\theta_{old}}(a \mid s).Q_{\theta_{old}}(s, a)$ is computed at each state-action pair (s_t, a_t) by taking the discounted sum of future rewards along the trajectory. In *vine* estimation procedure, $s_0 \sim \rho_0$ is sampled and then the policy π_{θ_i} is simulated to generate a number of trajectories. A subset of N states is chosen along these trajectories, denoted $s_1, s_2, ..., s_N$, which are the "rollout set". For each state s_n in the rollout set, K actions are sampled according to $a_{n,k} \sim q(cdot \mid s_n)$. For each action $a_{n,k}$ sampled at each state s_n, $\hat{Q}_{\theta_i}(s_n, a_{n,k})$ is estimated by performing a rollout starting with state s_n and action $a_{n,k}$. The practical TRPO algorithm works by repeatedly performing the following steps:

1. Use the single path or vine procedures to collect a set of state-action pairs along with Monte Carlo estimates of their Q-values.

2. By averaging over samples, construct the estimated objective and constraint in eq. 20.

3. Approximately solve this constrained optimization problem to update the policy's parameter vector θ. Conjugate gradient algorithm followed by a line search could be used.

3 Related Work

Reddy et. al. [28] present a way in which deep reinforcement learning can learn effective teaching policies for spaced repetition that can select contents that should be presented next to students without explicitly modelling students. They formalized teaching for spaced repetition as POMDP and solved it approximately using TRPO on simulated students. Reward function was based on the objective, if the student's goal was to maximize the expected number of items recalled, then

$$R(s, \cdot) = \sum_{i=1}^{n} P[Z_i = 1|s] \tag{21}$$

and if the goal was to maximize the likelihood of recalling all items,

$$R(s, \cdot) = \sum_{i=1}^{n} logP[Z_i = 1|s] \tag{22}$$

At each time step, the teaching agent receives observation $o \in O$, where $\mathcal{O} = \{0, 1\}$ encodes whether or not the student correctly recalled the item shown to them. The observation distribution $O(z|s, a) = P[Z_a = z|s]$ is given by the recall likelihood specified by the student model. They concluded that deep reinforcement learning was able to give relatively good performance when compared with widely used heuristics like Leitner and SuperMemo. However, the reward function used by them was directly using the hyperparameters of the student models to predict the rewards. In this work, we will make use of an RNN that will predict the future rewards.

Abe et. al. [2] evaluated various types of reinforcement learning algorithms in the area of sequential targeted marketing by trying to optimize cost sensitive decision. This problem domain is similar to this work's problem domain as the state space can get really huge to represent explicitly. They compare indirect reinforcement learning, direct reinforcement learning and hybrid or semi-direct reinforcement learning. They have also used a technique called Q sampling for semi direct methods which combines sarsa learning with Q-learning. In Q-learning some initial estimates of the Q-values are estimated first for each state and then the method updates them at each time step according to the following equation:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_(t+1) + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t)) \tag{23}$$

Sarsa learning, on the other hand is a slightly less aggressive then Q-learning. In sarsa learning, there is no maximization over the actions in the transition state s_{t+1}. It also starts with some initial estimates of the Q-values for each state and uses the following equation to dynamically update them:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_(t+1) + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)) \tag{24}$$

The Q-sampling method selects only those states, from within randomly sampled sub-episodes, that satisfy the condition that the action taken in the next state is the best action with respect to the current estimate of the Q-value function. The value update is similar to eq. 24 used in sarsa learning, but the effect of the learning that occurs is similar to eq. 23 used in Q-learning because the sampling strategy ensures that $Q(s_{t+1}, a_{t+1}) = \max_a Q(s_{t+1}, a)$. They conducted two sets of experiments - first, in which complete modelling of the environment was possible and reinforcement learning methods had access to all features of the data and second, in which the modelling was imperfect and the learning methods had access to a subset of features characterizing the data. They concluded that indirect reinforcement learning algorithms work best when complete modelling of the environment is possible and also that their performance is not upto the par when complete modelling is not possible. They found that Hybrid methods can reduce computation to attain a given level of performance and give decent policies.

Zhao et. al. [42] introduces deep reinforcement learning to the area of recommender systems. They propose a system capable of continuously improving its strategy while interacting with the users as compared to traditional recommender systems that make recommendations by following fixed strategies. Typical recommender systems generate a list of items that are recommended to users but to make these systems more effective, they should also capture the relationship among recommended items and generate a list of complementary items that could be presented to the users. They model the interaction between users and the recommender system as a Markov decision process and they make use of an online user-agent environment simulator to predict a reward based on current state and a selected action, i.e., $f : (s_t, a_t) \rightarrow r$. They build upon the Actor-Critic framework for recommending policy. Fig. 9 shows their proposed framework. Their Actor-framework consists of two steps:1) generate

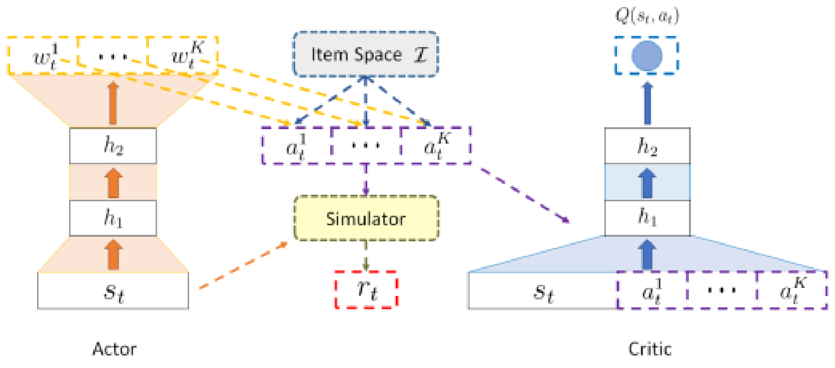

Figure 9: Framework proposed by Zhao et. al. for list-wise recommendation system using deep reinforcement learning. Image from Zhao et. al, 2017, *Deep Reinforcement Learning for List-wise Recommendations*

state-specific scoring function parameter which maps user's current state to a list of weight vectors 2) action generation which uses the scoring function parameter to compute scores for all the items. The recommender agent then picks the item with the highest score as its action. The Critic framework uses an approximator to learn an action-value function $Q(s_t, a_t)$ which decides whether the action selected by the Actor-framework matches the current state or not. The Actor-framework then uses $Q(s_t, a_t)$ to update its parameters and improve its performance in the following iterations.
However, there was no mention about the effectiveness of their proposed system and how well it performs when compared to other recommender systems.

Piech et. al. [26] used Recurrent Neural Networks to model student learning. Modelling student knowledge so that it becomes possible to accurately predict the performance of student on future interactions is known as Knowledge Tracing. More formally, given observations of interactions $x_0, ..., x_t$ taken by a student on a particular learning task, the knowledge tracing task is to predict characteristics of their next interaction $_{t+1}$. Interactions usually are defined by $x_t = \{q_t, a_t\}$ where q_t is the exercise being answered and a_t is a binary variable that tells whether or not the exercise was answered correctly. They used two different types of RNNs namely, vanilla RNN with sigmoid units and an LSTM network. They provide a novel way to encode student interactions into a sequence of fixed length input vectors x_t which was fed into these networks. When the datasets had small number of exercises, x_t was set to one-hot encoding of student interaction tuple $\{q_t, a_t\}$. When there were too many exercises, a random vector $\mathbf{n}_{q,a} \sim \mathcal{N}(0, 1)$ was assigned to each input tuple where $\mathbf{n}_{q,a} \in \mathbb{R}^N$. x_t was then set to the corresponding random vector $\mathbf{n}_{q,a}$. If $\delta(q_{t+1})$ denoted the one-hot encoding of which exercise was answered at time $t + 1$ and l denoted the binary cross entropy, then the loss for a given prediction was $l(y^T \delta(q_{t+1}), a_{t+1})$, and the loss for a single student was given by:

$$L = \sum_t l(y^T \delta(q_{t+1}), a_{t+1}) \tag{25}$$

Eq. 25 was minimized using stochastic gradient descent on minibatches. They used their model to choose the best exercises to present to the students and also to find latent relationships between exercises. They achieved a gain of 25% in AUC on one of the datasets and, according to them , was more than triple the largest gain achieved till now.

Mu, Goel and Brunskill [23] attempted to create an ITS by combining two works together for the problem domain of addition : automate creation of curriculum from execution traces and progressing of students using multi-armed bandit problem. Specifically, they use ZPDES, which is a multi-armed bandit algorithm, for problems selection on their underlying student models. The student models were modelled using **Bayesian Knowledge Tracing** [13]. By using ZPDES, they present students with different types of problems and at different intervals. They also propose a new probabilistic greedy, entropy based algorithm to estimate the initial level of the student also known as Zone of Proximal Development (ZPD) for initializing the ZPDES algorithm. They compare their system with a baseline of a fixed progression modeled off of progressions found in addition worksheet from various education websites online. They found out that their system, with pre-assessment, was able to perform well when compared with fixed progression of students.

Antonova [3] investigated the problem of developing sample-efficient approaches for learning adaptive strategies in domains where the cost of determining the effectiveness of any policy is too high. The work focuses on searching policy parameters space to find policies that could work well for a given objective and terms the method as direct policy search. For this work, two problem domains with costly samples were used - first, robotic locomotion and second, which is of particular interest for this work, the domain of ITS. Determining the effectiveness of policies is costly for ITS because evaluating a policy could take hours or weeks as it requires time and attention of student. The objective for this domain was to learn optimal stopping policies that would tell the number of exercises that were sufficient to learn a particular topic or skill. Bayesian Knowledge Tracing was used to model the students. At each time step, the agent had to choose between stopping or presenting student with more questions. For a policy π, it's performance, $R(\pi)$ is given by

$$R(\pi) = I(o_{t+1} = c)\frac{t_{max} - t}{t_{max} + 1} \tag{26}$$

where I is the indicator function, c stands for correct, and t is number of practice activities completed. Additive factors model was used to predict whether the student will be able to answer the next question correctly or not and was defined as

$$p(o_t = c \mid o_1, .., o_{t-1}) = \frac{1}{1 + \exp -(\beta_1 + \beta_2 n_c)} \tag{27}$$

where n_c is the number of times the student got the item correct. The instructional policy would halt if $p(o_t = c \mid o_1, .., o_{t-1}) > \beta_3$. $\mathcal{B} = (\beta_1, \beta_3, \beta_3)$ are three parameters parameterizing policy class. A new model-free approach named BO-RESAMPLE was developed to accelerate direct policy search with Bayesian Optimization by making use of sequence of observations which is usually not directly used by Bayesian Optimization. This approach works by performing simulated off-policy evaluations using a set of trajectories generated from past policy evaluations and the result is then passed back to BO. This process is repeated a fixed number of times until all the trajectories have been used or until a point is reached that can not be evaluated using the given history.

Kang [18], in his work, studied the effects of spaced repetition and how it could be used to produce superior long-term learning among school and college students. He states that while massed practice of the learning materials might produce effective short-term memory, spaced practice produces a durable long-term memory. According to one theory, when an item is repeated, it reminds the learner of its previous occurrence, which in turn prompts retrieving the previous presentation of the item, a process that improves memory retention. Massed repetition, on the other hand, removes this retrieval process as there is no need to retrieve the item from memory because the same item was just presented to the learner. He also states that the timing of the practice of items is equally important factor that affects learning. It can also be correlated with interleaved practice in which items are presented in an intermixed fashion during practice. In contrast, blocked practice presents the same kind of items together during practice. Although interleaved practice does provides spacing, it is different from spacing. Spaced review or practice proves useful to various forms of learning, including memory retention, problem solving, and generalization of earned concepts to new situations. He concludes that spaced review or practice is a efficient and cost-effective way to improve education outcomes.

Rubin and Wenzel [10] used variety of published dataset to determine if one retention function can generalize all kinds of memory or different function for a small number of different kinds of memory.

Settles and Meeder [30] proposed a new model for spaced repetition practice learning named Half-life regression which was applied to the area of language learning. In their system, in order to go to next lesson, all of the target words in the previous lesson must be learned. Once all the target words are learned, they are added to the student model which captures the knowledge of the student and their ability to recall items. It is, however, not mentioned in the paper, the approach to model the students. Half-Life Regression has already been defined above in the related theory section in eq. 2 and 3. Their goal was to find the optimal model weights Θ^* to minimize some loss function ℓ:

$$\Theta^* = \arg\min_{\Theta} \sum_{i=1}^{D} \ell(\langle p, \Delta, x\rangle_i; \Theta) \tag{28}$$

where data set $D = \{\langle p, \Delta, x\rangle_i\}_{i=1}^{D}$ made up of student-word practice sessions, recall rate p, lag time Δ since the word was last seen, and a feature vector x to personalize the learning experience. Loss function, ℓ was a variant of L2-regularized squared loss function and was given by:

$$\ell(\mathsf{x} : \Theta) = (p - \hat{p}_\Theta)^2 + \alpha(h - \hat{h}_\Theta)^2 + \lambda \mid\mid \Theta \mid\mid_2^2 \tag{29}$$

where ($\mathsf{x} = \langle p, \Delta, x\rangle$ defines the training data instance, λ is a parameter to control the regularization term, h is the half-life, $\hat{h}_\Theta$ is the estimated half-life, p is the recall rate, $\hat{p}_\Theta$ is the model prediction of recall rate and α is a parameter to control the relative importance of the half-life term in the overall training objective function. They compared performance of HLR against Leitner, Pimsleur and Logistic regression. The HLR model seemed to perform really well and they also stated that Leitner and Pimsleur can be assumed to be special cases of eq. 3 with hand-picked weights.

References

[1] James A. Kulik and J. D. Fletcher. Effectiveness of intelligent tutoring systems: A meta-analytic review. *Review of Educational Research*, 86, 04 2015.

[2] Naoki Abe, Edwin P. D. Pednault, Haixun Wang, Bianca Zadrozny, Wei Fan, and Chidanand Apté. Empirical comparison of various reinforcement learning strategies for sequential targeted marketing. In *Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM 2002), 9-12 December 2002, Maebashi City, Japan*, pages 3–10, 2002.

[3] Rika Antonova. Sample efficient bayesian optimization for policy search: Case studies in robotics and education. Master's thesis, Carnegie Mellon University, 2016.

[4] R. C. Atkinson. Computerized instruction and the learning process. *American Psychologist*, 23:225–239, 1968.

[5] Piotr A.Wozniak. Optimization of learning. Master's thesis, University of Technology in Poznan, 1990.

[6] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. *CoRR*, abs/1611.02167, 2016.

[7] A. Barr, P.R. Cohen, and E.A. Feigenbaum. *The handbook of artificial intelligence*. Number v. 1 in The handbook of artificial intelligence. Heuris-Tech Press, 1982.

[8] A. Bonnet. Artificial intelligence: Promise and performance. 1985.

[9] H. L. Burns and C. G. Capps. *Foundations of intelligent tutoring systems: an introduction*. Lawrence Erlbaum, London, 1988.

[10] David C. Rubin and Amy E. Wenzel. One hundred years of forgetting: A quantitative description of retention. 103:734–760, 10 1996.

[11] Han Cai, Tianyao Chen, Weinan Zhang, Yong Yu, and Jun Wang. Reinforcement learning for architecture search by network transformation. *CoRR*, abs/1707.04873, 2017.

[12] J. R Carbonell. Ai in cai: An artificial-intelligence approach to computer assisted instruction. *IEEE Transactions on Man-Machine Systems*, 11:190–202, 1970.

[13] 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, Dec 1994.

[14] Hermann Ebbinghaus. *Memory: A contribution to experimental psychology*. Teachers College, Columbia University., 1885. Translated by Henry A. Ruger & Clara E. Bussenius (1913).

[15] J. D Fletcher. Intelligent instructional systems in training. *Applications in artificial intelligence*, page 427–451, 1985.

[16] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. *CoRR*, abs/1709.06560, 2017.

[17] Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *Proceedings of the Nineteenth International Conference on Machine Learning*, ICML '02, pages 267–274, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[18] Sean H. K. Kang. Spaced repetition promotes efficient and effective learning: Policy implications for instruction. *Policy Insights from the Behavioral and Brain Sciences*, 3(1):12–19, 2016.

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

[20] Robert Victor Lindsey. *Probabilistic Models of Student Learning and Forgetting*. PhD thesis, University of Colorado Boulder, 2014.

[21] H. Mandl and A. Lesgold. *Learning Issues for Intelligent Tutoring Systems*. Springer, New York, NY, 1988.

[22] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, Dec 1943.

[23] Tong Mu, Karan Goel, and Emma Brunskill. Program2tutor: Combining automatic curriculum generation with multi-armed bandits for intelligent tutoring systems. 2017.

[24] Shamim Nemati, Mohammad M. Ghassemi, and Gari D. Clifford. Optimal medication dosing from suboptimal clinical examples: A deep reinforcement learning approach. In *38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBC 2016, Orlando, FL, USA, August 16-20, 2016*, pages 2978–2981, 2016.

[25] Hyacinth S. Nwana. Intelligent tutoring systems: an overview. *Artificial Intelligence Review*, 4(4):251–277, Dec 1990.

[26] Chris Piech, Jonathan Bassen, Jonathan Huang, Surya Ganguli, Mehran Sahami, Leonidas Guibas, and Jascha Sohl-Dickstein. Deep knowledge tracing. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pages 505–513, Cambridge, MA, USA, 2015. MIT Press.

[27] Niranjani Prasad, Li-Fang Cheng, Corey Chivers, Michael Draugelis, and Barbara E Engelhardt. A reinforcement learning approach to weaning of mechanical ventilation in intensive care units. *Proceedings of Uncertainty in Artificial Intelligence (UAI)*, pages 1–9, 2017.

[28] Siddharth Reddy, Sergey Levine, and Anca Dragan. Accelerating human learning with deep reinforcement learning. 2017.

[29] John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization. *CoRR*, abs/1502.05477, 2015.

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

[31] V. Shute. Regarding the i in its: Student modeling. *Proceedings of ED-MEDIA 94*, 1944.

[32] Siddhartha Banerjee Thorsten Joachims Siddharth Reddy, Igor Labutov. Unbounded human learning: Optimal scheduling for spaced repetition. *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2016.

[33] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine

Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, January 2016.

[34] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nature*, 550:354–, October 2017.

[35] D. Sleeman and J. S. Brown. *Intelligent tutoring systems.* Academic Press, 1982.

[36] Patrick Suppes and Mona Morningstar. Computer-assisted instruction. *Science*, 166(3903):343–350, 1969.

[37] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction.* MIT Press, Cambridge, MA, USA, 1998.

[38] E.L. Thorndike. *Animal Intelligence: Experimental Studies.* Animal behavior series. Macmillan, 1911.

[39] Kurt Vanlehn. The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. *Educational Psychologist*, 46:197–221, 10 2011.

[40] E. Wenger. *Artificial Intelligence and Tutoring Systems.* Morgan Kaufmanne, Los Altos, CA, 1987.

[41] J. T. Wixted and S. K. Carpenter. The wickelgren power law and the ebbinghaus savings function. *Psychological Science*, 18(2):33–134, feb 2007.

[42] Xiangyu Zhao, Liang Zhang, Zhuoye Ding, Dawei Yin, Yihong Zhao, and Jiliang Tang. Deep reinforcement learning for list-wise recommendations. 12 2017.