RNNs for knowledge tracing and curriculum generation

s1758150 s1771906 s1784599

Abstract

Personalised online education has the potential to improve the education of millions. Online education platforms have millions of users and vast amounts of data, however there has been relatively little work applying machine learning to personalise online education. This is despite the huge potential impact, availability of data, and initial evidence that current machine learning techniques can be applied effectively.

We reimplement a recurrent neural network (RNN) that reported state-of-the-art results in modeling knowledge acquisition in students (a task known as "knowledge tracing"). So far, our results achieve 65% AUC, compared to 75% reported in the literature. We discuss how using the hyperparameters reported in the literature, as well as novel dropout methods, failed to reproduce the result.

We are still hopeful that the state-of-the-art result can be replicated, and discuss how better knowledge tracing can be used to understand learning and to generate personalised curricula. We close with a proposal for training a RNN with richer latent structure to build a more complex representation of student knowledge acquisition.

1. Introduction

Providing personalised, high-quality education for everyone is one of the great promises of artificial intelligence. Massive online learning platforms such as Coursera, Duolingo, EdX and Khan Academy, have made it possible for anyone with an internet connection to access a variety of free, high-quality courses and even gain academic credentials. However, most online education systems guide a user through a hand-curated curriculum which is the same for all students. Personalised online education is yet to be achieved, but may have huge benefits. The famous "two sigma" estimate is that the average student under personalised tutoring performs about two standard deviations above the average student in a conventional class (Bloom, 1984). Of course individual in-person tutoring is also the most expensive and labour intensive form of education, and the "two sigma problem" is to provide this quality of education |under more practical and realistic conditions" (Bloom, 1984). Combining machine learning algorithms with the massive amounts of data available to online education platforms may now make personalised education realistic.

With millions of students using online education platforms and advances in machine learning, we may now have the data and techniques to model a student's knowledge and guide each individual through an effective, personalised curriculum. We would like to encourage more machine learning research in this area, and give a brief overview of research in machine learning for education before describing our work applying current machine learning techniques to the problem of personalised education.

There are two lines of research in machine learning related to personalising online education:

1. **Knowledge tracing**: Modelling a student's knowledge over time to predict how successful they will be at completing future tasks. (Corbett & Anderson, 1994; Piech et al., 2015)

2. **Curriculum generation**: Determining the optimal sequence of tasks to maximise the studentâĂŹs expected long-term knowledge. (Rafferty et al., 2016; Mu et al., 2017; Tabibian et al., 2017)

The current state-of-the-art in knowledge tracing is a recurrent neural network (RNN) trained on a sequence of (question, score) pairs which is trained to predict whether or not a student will get a question right given their history. This is known as Deep Knowledge Tracing (DKT) (Piech et al., 2015). The main alternative approach is Bayesian Knowledge Tracing (BKT) which models each skill as using a Hidden Markov Model (HMM) with a Bernoulli latent variable encoding whether or not the skill has been learned (Corbett & Anderson, 1994). DKT achieves better predictive performance (Piech et al., 2015; Khajah et al., 2016; Xiong et al., 2016), however the simple BKT model has a more interpretable latent representation of student knowledge. With a generative model such as BKT it is possible to simulate human students. Furthermore, cognitive scientists are interested in these models of knowledge acquisition in order to help understand the mechanisms of human learning (Lindsey et al., 2013; Patil et al., 2014).

Knowledge tracing can supplement curriculum generation. We can combine models of student knowledge with a model of how tasks increase knowledge to build an optimal curriculum. Alternatively, we can design curricula without a generative model of student learning using model-free reinforcement learning (RL) (Piech et al., 2015; Rafferty et al., 2016; Reddy et al., 2017). In this case, we require only observations of whether or not a student answers a proposed questions correctly or not, which may be a prediction from a knowledge tracing model or some other mathematical model of human knowledge (Reddy et al., 2017).

2. Aims and objectives

Our overall aim is to learn a richer representation of student knowledge acquisition than has previously been demonstrated in the literature, by training a latent variable RNN (Chung et al., 2015; Fraccaro et al., 2016) for the task of knowledge tracing . We aim to evaluate based on predictive performance as a basic goal, and as a stretch goal investigate the usefulness of better knowledge tracing for curriculum generation.

In this initial paper we present baseline results that attempt to reproduce the current state-of-the-art in knowledge tracing (Piech et al., 2015; Xiong et al., 2016), implementing a Long Short-Term Memory (LSTM) model (Hochreiter & Schmidhuber, 1997). We make modest extensions to this model by analysing the effect of an alternative dropout method (described in section 5). We are not able to match the current state-of-the art using the same model and data set reported in Piech et al. (2015) and Xiong et al. (2016) (full experimental results in section 6).

Ideally, knowledge tracing models would have both good predictive performance and a rich latent representation which can be use to better understand a studentâĂŹs current state of knowledge. RNNs which include latent variables in the hidden states, such as Variational RNN (VRNN) (Chung et al., 2015) and Stochastic RNN (SRNN) (Fraccaro et al., 2016), have recently been shown to perform well on sequences for the task of speech modeling. The objective of our future work to implement a latent variable RNN over both questions and scores, and evaluate the performance compared to the baseline for the task of knowledge tracing. We conclude with a full plan of work (section 8).

3. Data set and task

We test our knowledge tracing models on the Assistments 2009-2010 'skill-builder' public benchmark dataset [1]. Assistments is an online, automated tutor which assesses students in mathematics containing 222,332 answered problems from 3,104 different students on 146 types of problems. This dataset was used in the original deep knowledge tracing paper (Piech et al., 2015), but importantly our results will not be directly comparable to Piech et al. (2015) since the original version of the dataset was corrupted, containing approximately 23% duplicate rows. This duplication was discovered by Xiong et al. (2016), who subsequently cleaned the data. We use this cleaned dataset in our experiments [2].

The data consists of sequences of (question, score) pairs $(\mathbf{x}_{1:n_i}, \mathbf{y}_{1:n_i})$ where n_i is the number of questions answered by student i, $\mathbf{x}_t$ is the identifier of a distinct type of question, such as 'trigonometry' and $\mathbf{y}_t$ is a binary label where 0 indicates incorrect and 1 indicates correct. At each time step t, our goal is to predict the label $\mathbf{y}_t$, given the question $\mathbf{x}_t$ and the student's current history of questions and answers $(\mathbf{x}_{1:t-1}, \mathbf{y}_{1:t-1})$. The loss function for this sequence of predictions can thus be written as:

Questions answered	Students	Types of Questions
222332	3104	146

Table 1. Assistments 2009 skill builder data set

$$L(\theta) = \sum_{i}^{S} \sum_{t}^{n_i} l(f(\mathbf{x}_t, \theta), \mathbf{y}_t)) \tag{1}$$

where S is the total number of students, l is the binary entropy loss between a prediction $f(\mathbf{x}_t, \theta)$ and target $\mathbf{y}_t$, and f is parameterised by a RNN with weights θ.

Our key evaluation metric is the Area Under the Curve (AUC), where 'the curve' refers to the Receiver Operator Characteristic (ROC) curve. Classification accuracy (with a threshold of 0.5) is the dominant performance metric in machine learning generally, so we also include accuracy as a secondary metric in our results. One practical reason to use AUC is that our work builds on the Bayesian knowledge tracing literature, which has historically preferred AUC, and so we need to use it for our results to be comparable. There is also a deeper reason: theoretical comparisons of performance metrics. Hernández-Orallo et al. (2012) argue that AUC is a better measure than accuracy when you are uncertain of the relative cost of misclassification between classes. In the educational setting, false positives are typically more costly than false negatives, but there is uncertainty in the margin of difference. This asymmetry exists because if you wrongly believe a student will answer a question correctly and thus recommend that the student attempt it, the student might become demotivated upon failing.

4. Data preprocessing

Three key preprocessing steps are required before feeding the data into an RNN.

1. The distinct question ids x_t are embedded in a higher dimension: a one-dimensional encoding suggests that you can interpolate between questions 1 and 3 to get question 2, which is false in general.

2. The x_t are combined with the corresponding y_t, producing a vector $\mathbf{z}_t$. We can then specify a student's history of question/answer pairs, for any t, with a sequence of vectors $\mathbf{z}_{1:t}$.

3. Finally, each student's full history, $\mathbf{z}_{1:n_i}$ is right-padded with zero-vectors, such that every student has the same length sequence. So, for example, if the maximum number of questions answered by any student is 100,

[1] https://sites.google.com/site/assistmentsdata/home/assistment-2009-2010-data/skill-builder-data-2009-2010

[2] https://github.com/siyuanzhao/2016-EDM/tree/master/data

then we right pad each student's sequence with zero-vectors until they all have length 100. We denote this maximum length as max_{id}, or when used as a subscript: m.

The precise transformations used in steps 1 and 2 remain to be specified. We present two transforms from the literature: a one-hot encoding transform and a 'compressed sensing' transform. Finally, we suggest our own novel transform. In this paper, we only experiment with the one-hot transform, but intend to experiment with the other 2 in the second half of the project.

The first transform, used by the authors of Piech et al. (2015) and Xiong et al. (2016) is a one-hot encoding given by:

$$(x_t, y_t) \rightarrow \mathbf{z}_t \quad \text{where } (\mathbf{z}_t)_i = \begin{cases} \delta_{i,x_t} \text{ if } y_t = 0 \\ \delta_{i,m+x_t} \text{ otherwise} \end{cases} \tag{2}$$

Where the resultant vector has dimension $2max_{id}$. Whilst a vector of size max_{id} would be sufficient to embed the distinct questions x_t, we double the dimension to account for the binary answer label y_t.

This embedding is problematic since it scales linearly with the number of distinct question types, which may become very large as we broaden the dataset or make the question types more granular. To remedy this issue, Piech et al. (2015) applies a form of dimensionality reduction to the $\mathbf{z}_t$, using a transform inspired from the field of *compressed sensing*. Specifically, they map the one-hot encoded $\mathbf{z}_t$ to a random normal vector $\mathbf{n}_t \sim \mathcal{N}(0, I)$ in a lower dimensional space.

The one-hot encoding is not only problematic because of its size. It also fails to encode the relationship between the same question being answered correctly versus incorrectly - the two cases are mapped to different dimensions - and so the RNN has to learn that there exists a meaningful connection between $\mathbf{z}_{1:m}$ and $\mathbf{z}_{m:2*m}$. We think it is plausible that having to learn such a relationship might unnecessarily increase task difficulty and reduce predictive performance. We intend to test this claim by experimenting with a different transformation of the data in the second half of the project, given by:

$$(x_t, y_t) \rightarrow \mathbf{z}_t \quad \text{where } (\mathbf{z}_t)_i = \delta_{i,x_t}(-1)^{y_t+1} \tag{3}$$

Where $\mathbf{z}_t$ has dimension max_{id}, and is all zero, except for the dimension x_t, which is equal to -1 if $y_t = 0$, and 1 if $y_t = 1$. This embedding assigns one dimension to each question type, regardless of correctness.

5. Recurrent neural networks

Recurrent neural networks (RNNs, Rumelhart et al. (1986)) are a type of neural network model that specialise in time-series data, achieving state of the art results in on a wide range of sequential classification tasks (Lipton et al., 2015). Feed forward, fully connected neural networks lack a memory component, and so struggle to discover time-dependent structure in a data source. RNNs implement a form of memory by maintaining a *hidden state*, which is updated at each time step. This sequence of hidden states can then be unrolled to generate a directed acyclic computational graph that we can backpropagate through. The net effect of this procedure is a form of memory compression: long sequences are distilled into a single hidden state.

Formally, the state of the hidden layer at time t is computed using input value x_t and the state of the hidden layer at time $t-1$ as follows:

$$H_t = tanh(W_h x_t + W_{hh} h_{t-1} + b_h) \tag{4}$$

$$Y_t = sigmoid(W_{yh} h_t + b_y) \tag{5}$$

where the W terms are weights that are updated through backpropagation.

Unfortunately, this basic RNN architecture still struggles to learn long sequences. In the back-propagation step of RNNs, a gradient value could become an extremely small or large; this is known as the 'vanishing' and 'exploding' gradient problem, respectively. This is caused by the unrolled computational graph becoming too deep, with a long sequence of matrix multiplications and non-linear activations leading to saturation or instability.

The vanishing gradient problem spurred the development of the Long Short Term Memory (LSTM, Hochreiter & Schmidhuber (1997)) architecture. LSTM is comprised of cells which can store, restore, remove and keep information. Depending on analog signals calculated by activation function of gates connected to the cells, each cell decides which operation to perform. There are three types of gate. First, the input gate determines how much input data will be loaded in the cell. Second, the forget gate is to adjust remaining value of the cell. Finally, the output gate calculates the output activation of the LSTM unit. See figure 1 for a visual representation.

6. Experimental results

In this section, we present a sequence of experiments that attempt to replicate the results of Xiong et al. (2016), demonstrating that our data preprocessing and model construction was implemented correctly. However, we encounter significant *overfitting* that we have yet to resolve in this initial phase of the project, despite a series of attempts to regularise our model. We present these attempts here as a foundation for the second half of the project, where we intend to begin by solving the problem of regularisation in our model.

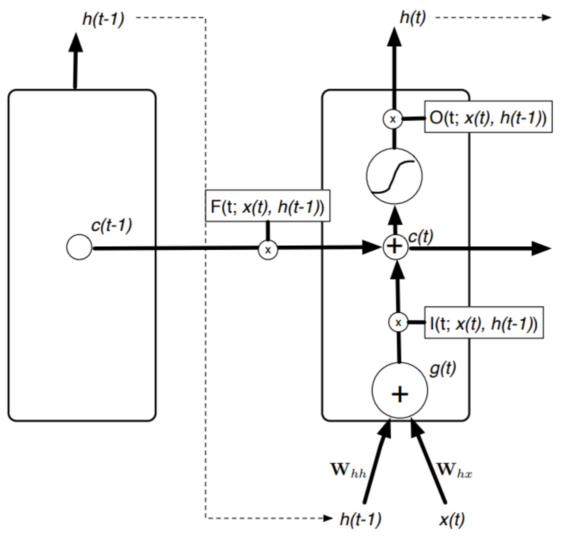

Figure 1. Schematic diagram of a LSTM cell. The small circular nodes marked with an 'x' represent the three types of gates: input, output and forget. The central '+' node represents the hidden state. Figure from MLP course lecture.

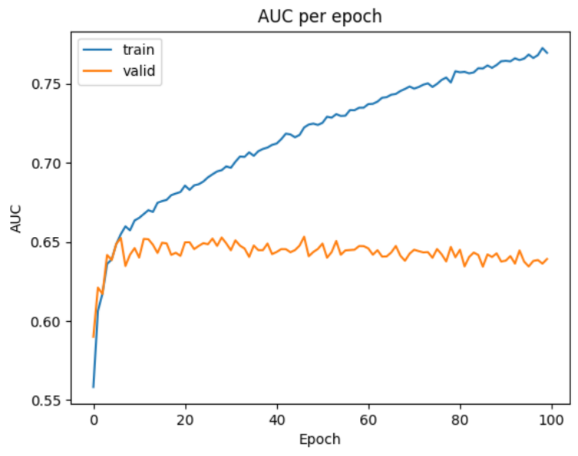

Figure 2. AUC curve showing overfitting in our LSTM model when using dropout just on the outputs of the RNN cell. Learning rate 0.01, and setup as described in section 6.1.

6.1. Experimental setup

For all experiments, we used an LSTM with 1 hidden layer and 200 hidden units. We minimise the loss function given in equation 1 using the Adam optimiser with first moment decay rate of 0.9, and second moment decay rate of 0.999 as recommended by Kingma & Ba (2014). The learning rate itself is varied in our experiments, but we consistently apply learning rate decay with a value of 0.96 every 2 epochs. We prevent the well-known problem of 'exploding gradients' by gradient clipping with a threshold of 20. Training proceeded in mini batches of size 100, which corresponds to 100 students per batch, and lasts for a total of 100 epochs. As explained in section 3, our primary evaluation metric is Area Under the Curve (AUC), hence we choose between models based on their AUC score on the validation set.

6.2. Regularisation with dropout

RNNs are notoriously difficult to regularise. The standard application of dropout to a RNN involves applying a different stochastic binary mask to the network weights at each time step. Empirically, this works for input and output connections, but can have devastating consequences if applied to the hidden state. As such, the papers we reproduce here (Piech et al., 2015; Xiong et al., 2016) only apply dropout to the output weights, which is a rather restricted form of regularisation.

Recent advances in applying variants of dropout to RNNs can potentially alleviate this problem (Gal & Ghahramani, 2016; Krueger et al., 2016). In particular, Gal & Ghahramani (2016) provide a theoretical argument that applying the same stochastic binary mask across all time-steps amounts to a variational approximation to a Bayesian RNN. We test this claim by comparing: a) standard dropout with p=0.6 applied to the output weights and b) variational dropout with p=0.6 applied to all weights.

6.3. Initial Experiments

Our initial experiments are pure replications of Xiong et al. (2016). The set-up is as described above, and dropout with 'keep probability' 0.6 is applied to the outputs of the LSTM cell. We experimented with various learning rates, but found that overfitting was a significant problem in all cases. Figure 2 shows the best performing model, which was trained with learning rate 0.01 and dropout 0.6, and achieved validation AUC of 65%, whereas Xiong et al. (2016) report achieving 75% AUC.

6.4. Experiments with variational dropout

We implemented variational dropout, applying the same dropout mask across all time-steps and applying dropout of 0.6 to both the output of the hidden cell and the hidden state itself. We found that this prevent the model overfitting to the training data but still limited overall performance which plateaus at 65% AUC on the validation set, as shown in 3. Since the training set performance we know not to be limited to 65% and there is evidence in the literature that 75% AUC can be achieved, we are still hopeful that further tuning of hyperparameters can strike a balance that limits overfitting without harming overall performance on the validation set.

We experimented with adding noise to the gradient as well as dropout, where the noise value is generated from a zero mean Gaussian distribution with standard deviation 0.001, and found that it degraded performance.

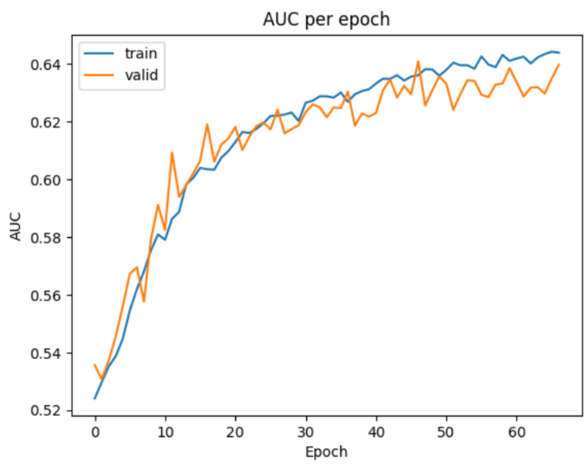

Figure 3. AUC performance curve showing similar performance on training and validation sets when using variational dropout, but only 65% validation AUC. Learning rate 0.01, and setup as described in section 6.1.

7. Interim conclusions

We have shown that our baseline LSTM model can achieve good performance on the training data, but currently suffers from overfitting: only reaching an AUC of 65%, which is 10% less than the state of the art. Our attempts to rectify this problem illustrate the difficultly of regularising RNNs. In particular, our application of dropout to the output units of the network was ineffective, as was variational dropout and gradient noise.

We suspect that some combination of these hyperparameters settings, along with the right learning rate and decay schedule, will ultimately attain state of the art performance. Thus, in the second half of the project we will employ a more exhaustive search of the hyperparameter space.

8. Plan

The first step in the second half of the project is to resolve the problem of overfitting we encountered in our initial baseline experiments, in order to fully reproduce current state of the art results in deep knowledge tracing. We expect this to be solvable with a more exhaustive search of the hyperparameter space, and is therefore not a substantial amount of work.

The core part of our plan is to apply advances in latent variable RNN models to knowledge tracing, with a primary aim to investigate their predictive performance and a secondary aim of investigating their usefulness for curriculum generation.

8.1. Primary task: knowledge tracing

In the second half of the project, we will apply a more sophisticated RNN architecture to the knowledge tracing task, with the primary aim of improving upon our baseline AUC score. Specifically, we will take advantage of recent advances in latent variable RNN models (Chung et al., 2015; Fraccaro et al., 2016) by implementing a stochastic recurrent neural network (SRNN).

An SRNN unifies the set of state space models and RNNs: it simply stacks one on top of the other. The set of state space models include the Hidden Markov Model (HMM), which is used in bayesian knowledge tracing. Thus, an SRNN represents the perfect marriage of bayesian knowledge tracing and deep knowledge tracing, potentially combining the benefits of both to obtain a higher AUC score than our baseline.

SRNNs have demonstrated superior ability to RNNs at modeling complex, but highly structured data. The terms 'complex' and 'highly structured' are non-rigorous, but used to convey the idea that the data is generated from a relatively small set of stochastic causes, whose stochasticity we can model (Chung et al., 2015). Whilst the causes may be few in number, the process that generates the observed data from the causes can be incredibly intricate. We believe that our data matches this description. A student's learning trajectory is, at least in part, shaped by complex interactions between the set of interdependent skills a student possesses; the combinations of skills required to solve each problem and the relationship between problem solving and skill increase.

8.2. Auxiliary task: curricula generation

If we are able to train SRNN that achieves reasonable predictive results on the primary task, then we would like to explore whether the model can be used for the task of curricula generation, which we see as the crucial next step in using deep learning to improve the lives of students.

The primary task essentially involves generating a sequence of scores (correct/incorrect) for a student's answers, where at each time-step we condition on the student's previous question and score. However, we could modify this setup, so that we instead generate a sequence of (question, score) pairs. That is, we could simulate a learning trajectory. Simulated learning trajectories would be a use if we could specify that the sequence must end in the student getting a problem correct, and use the model to fill in the most probable beginning of the sequence. Such a task would essentially correspond to curriculum generation: the model has to create a sequence of problems that would teach a student to correctly answer a specified target question.

There are two challenges to this approach. The first challenge is how to specify the end of a sequence such that the beginning can be easily generated. We speculate that this problem can be overcome by reversing the order of the dataset. That is, we train an SRNN on a studentâĂŹs sequence of problems and scores, but with the sequence reversed. This is desirable because it is straightforward to specify the start of a sequence, and have the model complete it.

The second challenge is how to evaluate success. The problem of evaluation can be tackled by leveraging the fact that an SRNN is a probabilistic model, and so can assign a likelihood to a given sequence. Therefore, we could compare the likelihood of our generated sequences to the average likelihood of a sequence in the training set.

To our knowledge, we are the first to suggest using this method for curriculum design. The small body of related work on this topic uses either simple algorithms derived from cognitive psychology or model-free RL (Lindsey et al., 2013; Rafferty et al., 2016; Mu et al., 2017; Reddy et al., 2017). Of course the baseline RNN model, or an SRNN model that improves on it, could be used in model-free RL approaches to curriculum generation, by being used as a black-box which generates the observation of whether or not the student get the suggested question correct. We leave applications in RL to future research.

9. Summary

We have presented preliminary results showing how to apply a recurrent neural network to the task of predicting student performance on sequences of questions. Our baseline LSTM model can achieve good performance on the training data, but currently suffers from overfitting: only reaching an AUC of 65%, which is 10% less than the state of the art. We have discussed the different regularisation techniques we implemented, and argue that they ought to be sufficient to reproduce state of the art results after we have completed a more exhaustive hyperparameter search in the second half of the project.

We also proposed plans to implement a latent variable stochastic RNN in the second half of the project, which represents a pleasing synthesis of the models from the bayesian and deep knowledge tracing literature. We discuss how such a model can perhaps improve upon the state of the art, and even be used for curricula generation.

References

Bloom, Benjamin S. The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational researcher*, 13(6):4–16, 1984.

Chung, Junyoung, Kastner, Kyle, Dinh, Laurent, Goel, Kratarth, Courville, Aaron C, and Bengio, Yoshua. A recurrent latent variable model for sequential data. In *Advances in neural information processing systems*, pp. 2980–2988, 2015.

Corbett, Albert T and Anderson, John R. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User modeling and user-adapted interaction*, 4(4):253–278, 1994.

Fraccaro, Marco, Sønderby, Søren Kaae, Paquet, Ulrich, and Winther, Ole. Sequential neural models with stochastic layers. In *Advances in neural information processing systems*, pp. 2199–2207, 2016.

Gal, Yarin and Ghahramani, Zoubin. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems*, pp. 1019–1027, 2016.

Hernández-Orallo, José, Flach, Peter, and Ferri, Cèsar. A unified view of performance metrics: translating threshold choice into expected classification loss. *Journal of Machine Learning Research*, 13(Oct):2813–2869, 2012.

Hochreiter, Sepp and Schmidhuber, Jürgen. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Khajah, Mohammad, Lindsey, Robert V, and Mozer, Michael C. How deep is knowledge tracing? *arXiv preprint arXiv:1604.02416*, 2016.

Kingma, Diederik P and Ba, Jimmy. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Krueger, David, Maharaj, Tegan, Kramár, János, Pezeshki, Mohammad, Ballas, Nicolas, Ke, Nan Rosemary, Goyal, Anirudh, Bengio, Yoshua, Courville, Aaron, and Pal, Chris. Zoneout: Regularizing rnns by randomly preserving hidden activations. *arXiv preprint arXiv:1606.01305*, 2016.

Lindsey, Robert V, Mozer, Michael C, Huggins, William J, and Pashler, Harold. Optimizing instructional policies. In *Advances in Neural Information Processing Systems*, pp. 2778–2786, 2013.

Lipton, Zachary C, Berkowitz, John, and Elkan, Charles. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*, 2015.

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

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

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

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

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

Rumelhart, David E, Hinton, Geoffrey E, and Williams, Ronald J. Learning representations by back-propagating errors. *nature*, 323(6088):533, 1986.

Tabibian, Behzad, Upadhyay, Utkarsh, De, Abir, Zarezade, Ali, Schoelkopf, Bernhard, and Gomez-Rodriguez, Manuel. Optimizing human learning. *arXiv preprint arXiv:1712.01856*, 2017.

Xiong, Xiaolu, Zhao, Siyuan, Van Inwegen, Eric, and Beck, Joseph. Going deeper with deep knowledge tracing. In *EDM*, pp. 545–550, 2016.