Option Tracing: Beyond Binary Knowledge Tracing

Aritra Ghosh
University of Massachusetts Amherst
Amherst, MA 01002
`arighosh@cs.umass.edu`

Andrew S. Lan
University of Massachusetts Amherst
Amherst, MA 01002
`andrewlan@cs.umass.edu`

Abstract

This paper details our solutions to Tasks 1&2 of the NeurIPS 2020 Education Challenge.[1] Knowledge tracing, a family of methods to estimate each student's mastery levels on skills/knowledge components from their past responses to assessment questions, is useful for progress monitoring, personalization, and helping teachers to deliver personalized and targeted feedback to students to improve their learning outcomes. One key limitation of current knowledge tracing methods is that they can only estimate an *overall* knowledge level of a student since they analyze only the binary-valued correctness of student responses. We adapt a series of popular knowledge tracing methods to the task of option prediction in multiple choice questions. Experimental results show that our method performs well on both option prediction and correctness prediction.

1 Introduction

Knowledge tracing (KT) [3], i.e., estimating student mastery levels on knowledge component/skill/concept they are learning from their past responses to assessment questions/items and using them to predict their future performance, is a key task in learning analytics. Many different KT methods have been developed, ranging from hidden Markov model-based Bayesian knowledge tracing [8, 10, 14], factor analysis-based models [1, 11], especially the item Difficulty, student ability, skill, and student skill practice history (DAS3H) model [2], to deep learning-based models [5, 12]. These methods have enjoyed various degrees of success; the former two types of KT methods exhibit excellent interpretability while deep learning-based methods trade off interpretability for excellent predictive accuracy on students' future performance. However, one key limitation of these binary KT methods is that they use the binary-valued indicator of whether a student responds to a question correctly or not, rather than their exact response, to estimate their *overall* mastery level on each knowledge component. However, not all incorrect responses are equal: past research has found that the number of different ways student answers can be incorrect can be large: in step-by-step solutions to open-ended mathematics problems, there are often more than ten incorrect ways to solve a problem [9], each corresponding to a unique cause of error; in short-answer questions, only $30-60\%$ of incorrect answers generated by students are anticipated by teachers or numerical simulations [4, 13]. The Eedi dataset provided by the NeurIPS 2020 Education Challenge that include the exact options students select on multiple choice questions (MCQs) provides us with an opportunity to partially address these limitations; despite not open-ended, certain incorrect distractor options in well-designed MCQs can provide valuable insights into a student's *error mode* and the *source* of the error.

2 Problem Setup

Each student's performance record consists of a sequence of responses to questions assigned at a series of discrete time steps. For student i at time step t, we denote the combination of the question $q_t^i \in \mathbb{N}^+$

1st NeurIPS Education Challenge (NeurIPS 2020), Vancouver, Canada.

[1]Our code is available at `https://github.com/arghosh/NeurIPSEducation2020`.

that they answered, the set of subjects $\{s_{t,j}^i\}_{j=1}^{n_t^i}$ this question covers, their binary-valued response correctness $r_t^i \in \{0, 1\}$ (1 corresponds to a correct response, which means the student selected the correct option of the question), the option $y_t^i \in \{A, B, C, D\}$ they chose, and the correct option $c_t^i \in \{A, B, C, D\}$ to this question as a tuple, $(q_t^i, \{s_{t,j}^i\}_{j=1}^{n_t^i}, r_t^i, y_t^i, c_t^i)$, where $s_{t,j}^i \in \mathbb{N}^+$ denotes the index of the j^{th} subject, $j \in 1, \dots, n_t^i$ since each question can be tagged with multiple subjects. The datasets also contains the timestamp of each interaction. We have several additional features (quiz id, group id, student demographic information) associated with each tuple. To remove excessive notational clutter, we chose to omit these features in our modeling descriptions. However, we always use these features in our experiments; the experimental section details our approach to incorporate them. In the challenge setup, we associate a flag variable $f_t^i \in \{0, 1\}$ with each time step, where 1 represents that the timestep is part of the challenge training set and available to use. This variable help us to mask the data when we compute the training loss. Given $(q_t^i, \{s_{t,j}^i\}_{j=1}^{n_t^i}, r_t^i, y_t^i, c_t^i, f_t^i = 1)$ and $(q_t^i, \{s_{t,j}^i\}_{j=1}^{n_t^i}, c_t^i, f_t^i = 0)$, the task is to predict $y_{t'}^{i'}$ (and $r_{t'}^{i'}$) for $(t', i') \in \{(t, i) : f_t^i = 0\}$.

3 Methodology

Before delving into the individual methods, we start with a set of unified modules that apply to all methods discussed in this paper. The question embedding module $\mathcal{E}_q : q \to \mathbb{R}^d$ transforms the question index q_t^i to a d-dimensional, learnable real-valued vector in $\mathbb{R}^d$. Similarly, the response embedding module $\mathcal{E}_r : r \to \mathbb{R}^d$ transforms the response correctness r_t^i to $\mathbb{R}^d$ and the option embedding module $\mathcal{E}_o : \{A, B, C, D\} \to \mathbb{R}^d$ transforms the correct option c_t^i and the chosen option y_t^i to vectors in $\mathbb{R}^d$. We do not use separate embeddings for every question-option (q, o) pair since that leads to severe overfitting in our experiments; instead, the $2d$-dimensional embedding for (q, o) is obtained using $[\mathcal{E}_q(q) \oplus \mathcal{E}_o(o)]$ where $\oplus$ is the concatenation operator. The subject embedding module $\mathcal{E}_s : s \to \mathbb{R}^d$ transforms the subject index to $\mathbb{R}^d$. Since each question may be tagged with several subjects, we define the final subject embedding as $\mathcal{E}_s(\{s_{t,j}^i\}_{j=1}^{n_t^i}) = \sum_{j=1}^{n_t^i} \mathcal{E}_s(s_{t,j}^i)$. Some of the methods (such as NCF) use a user embedding module $\mathcal{E}_u : i \to \mathbb{R}^d$ that transforms the student index to $\mathbb{R}^d$. For simplicity, we use the same d-dimensional vector for all embedding modules; however, the dimensions of each module can be different. We train all model parameters, denoted as Θ, which contains the embeddings listed here and other model parameters specific to each individual method, by maximizing the log-likelihood of the selected options (or responses),

$$\underset{\Theta}{\operatorname{argmax}} \sum_{i=1}^{S} \sum_{t=1}^{Q_i} \{ \sum_{o \in \{A,B,C,D\}} \mathbb{1}[y_t^i = o] \log \Pr(o; \Theta) \quad \text{or} \sum_{c \in \{0,1\}} \mathbb{1}[r_t^i = c] \log \Pr(c; \Theta) \},$$

where $\mathbb{1}$ is the indicator function and S and Q_i denote the number of students and the number of questions student i responded to that are observed, respectively. Since the options (responses) are categorical (binary), the resulting loss function corresponds to the common cross-entropy loss (binary cross-entropy loss) [6].

3.1 Model

NCF. Neural collaboartive filtering (NCF) is one of the most popular CF algorithms for user-item interaction datasets. In the option prediction task, students correspond to users and questions corresponds to items. The input for NCF at time step t for student i, $\mathbf{x}_t^i$, is given by

$$\mathbf{x}_t^i = [\mathcal{E}_q(q_t^i) \oplus \mathcal{E}_u(i) \oplus \mathcal{E}_s(\{s_t^i\})].$$

Predictive probabilities $p(y_t^i = o)$ over four options are calculated using the softmax function [6],

$$\mathbf{z}_t^i = \text{NN}(\mathbf{x}_t^i) \in \mathbb{R}^4, \ \Pr(y_t^i = o \in \{A, B, C, D\} | \mathbf{x}_t^i) = [\text{Softmax}(\mathbf{z}_t^i)]_o, \ \hat{y}_t^i = \text{argmax}_{o \in \{A,B,C,D\}} [\mathbf{z}_t^i]_o,$$

Predictive probabilities $p(r_t^i = c)$ over two responses are calculated using the sigmoid function [6],

$$\mathbf{z}_t^i = \text{NN}(\mathbf{x}_t^i) \in \mathbb{R}^1, \quad \Pr(r_t^i = 1 | \mathbf{x}_t^i) = \text{Sigmoid}(\mathbf{z}_t^i), \quad \hat{r}_t^i = \mathbb{1}[\mathbf{z}_t^i \geq 0.5],$$

where $\text{NN}(\cdot)$ denotes a feed-forward, fully-connected neural network and $[]_o$ refers to the o^{th} entry of a vector. In NCF, the model parameters are the weights and biases in the feed-forward neural network, $\text{NN}(\cdot)$. This prediction module is shared by the subsequent methods.

NCF+ DAS3H. The NCF model uses embeddings for students that are static over time and independent of their current learning context, corresponding to the static student ability and question difficulty parameters in IRT models. The DAS3H model augments IRT by summarizing student's learning context using some hand-crafted features such as the number of correct past responses and the number of incorrect past responses on questions in the same subject as the current question. Motivated by DAS3H, we augment the NCF model with DAS3H-like features as

$$\mathbf{x}_t^i = [\mathcal{E}_q(q_t^i) \oplus \mathcal{E}_u(i) \oplus \mathcal{E}_s(\{s_t\}) \oplus \mathbf{W}^T \text{DAS3H Features}],$$

where $\mathbf{W} \in R^{16 \times d}$ is a weight matrix mapping the DAS3H features into the input vector space. Since in the challenge setup we also have access to a student's future responses when we are predicting their current response, we include two additional features for the total number of future correct attempts and future incorrect attempts. Moreover, since questions may be tagged with multiple subjects, we compute these count where set of subjects are partially (4 features) or fully matched (4 features) between the current question and the past/future questions, resulting in a total of 8 features. We compute additional 8 features for interactions (past and future) that are attempted in a shorted time periods (≤ 0.1 day). Similar to [2], we apply $\log$ to these counts. The model parameters for NCF+DAS3H include the additional parameter matrix $\mathbf{W}$ in addition to the NCF method.

LSTM. The main drawback of NCF is that the student embedding is static and not updated as students respond to more questions and their knowledge states evolve. Recurrent neural networks, and in particular long short-term memory network (LSTM)-based approaches are capable of modeling evolving knowledge or hidden states [12]. However, we cannot directly use methods such as DKT in the CF setup since the student's responses at some time steps in their past are not observed. Therefore, we use the following method to handle evolving knowledge states using recurrent networks with missing observations. The input to the LSTM at each time step is given by

$$\mathbf{x}_t = [\mathcal{E}_q(q_t^i) \oplus \mathcal{E}_o(c_t^i) \oplus \mathcal{E}_s(\{s_t^i\}) \oplus \big(\mathcal{E}_o(y_t^i) \odot f_t^i\big) \oplus \big(\mathcal{E}_l(r_t^i) \odot f_t^i\big)], \tag{1}$$

where $\odot$ denotes the element-wise multiplication between two vectors. We mask the option embeddings and response correctness embeddings using the flag variable for time steps where we do not observe them; we still use the question embedding as input since we know the student responded to the question. Then, the student's latent knowledge states at the next time step are updated using the LSTM module as $\mathbf{h}_{t+1} = \text{LSTM}(\mathbf{h}_t, \mathbf{x}_t)$. The output to the prediction module is computed using,

$$\mathbf{z}_t^i = \text{NN}([\mathbf{h}_t^i \oplus \mathcal{E}_q(q_t^i) \oplus \mathcal{E}_o(c_t^i) \oplus \mathcal{E}_s(\{s_t^i\})]) \in \mathbb{R}^{4 \text{ or } 1}. \tag{2}$$

The parameters in the LSTM model include the input and transition weight matrices and bias vectors.

Bi-LSTM. Similar to our extension to the NCF method using features of both a student's past and future responses, we can extend the base LSTM model to a bi-directional LSTM (Bi-LSTM) model [7]' to learn student's knowledge states using information on both the past and the future. Here, we compute two latent knowledge states using two separate LSTM modules, the forward state $\overrightarrow{\mathbf{h}}_t$ that summarizes the student's past response history and the backward state $\overleftarrow{\mathbf{h}}_t$ that summarizes the student's future response history at time step t as

$$\overrightarrow{\mathbf{h}}_{t+1} = \text{Forward LSTM}(\overrightarrow{\mathbf{h}}_t, \mathbf{x}_t),\ \overleftarrow{\mathbf{h}}_{t-1} = \text{Backward LSTM}(\overleftarrow{\mathbf{h}}_t, \mathbf{x}_t).$$

The final latent knowledge state is the concatenation of the two states as $\mathbf{h}_t = [\overrightarrow{\mathbf{h}}_t \oplus \overleftarrow{\mathbf{h}}_t]$. The parameters in the Bi-LSTM model include two sets of parameters for the forward and backward LSTMs.

Bi-LSTM+DAS3H. Similarly, we also include the DAS3H features in the Bi-LSTM model. We concatenate DAS3H features in the output module as,

$$\mathbf{z}_t^i = \text{NN}([\mathbf{h}_t^i \oplus \mathcal{E}_q(q_t^i) \oplus \mathcal{E}_o(c_t^i) \oplus \mathcal{E}_s(\{s_t^i\}) \oplus \mathbf{W}^T \text{DAS3H Features}]) \in \mathbb{R}^{4 \text{ or } 1}.$$

Bi-LSTM+DAS3H+ Attention. We also observe that students respond to questions from the same quiz together within a short period of time. Since it is reasonable to assume that the knowledge states of students do not change significantly in a short period of time, we explored using attention modules to take other questions in the same quiz into account when predicting the student's response to a

Model	Validation Split	Response Prediction			Option Prediction		
		Local	Public	Private	Local	Public	Private
NCF	20%	74.39	74.68	74.69	65.26	65.36	-
NCF+DAS3H	20%	75.76	-	-	65.81	-	-
LSTM	20%	75.15	-	-	66.04	-	-
Bi-LSTM	20%	76.31	76.26	-	66.71	66.70	-
Bi-LSTM+DAS3H	20%	76.45	-	-	66.78	-	-
Bi-LSTM+DAS3H	5%	76.88	76.89	-	67.30	-	-
Bi-LSTM+DAS3H +Attention	5%	76.98	76.92	76.94	67.31	67.33	67.38

Table 1: Accuracy (in %) of all methods on a local test dataset (part of the challenge's training set) and the challenge public and private leaderboard test datasets.

question. The queries and the keys for each question are computed using linear layers on the quiz embeddings. The values are computed using a linear layer on the option and response correctness embeddings as

$$\text{Query}_t^i = \mathbf{W}_Q \mathcal{E}_{qz}(\text{quiz-id}_t^i),\ \text{Key}_t^i = \mathbf{W}_K \mathcal{E}_{qz}(\text{quiz-id}_t^i),\ \text{Value}_t^i = \mathbf{W}_V \mathcal{E}_r(r_t^i),$$
$$\mathbf{g}_t^i = \text{Masked Attention}_{\tau, \tau \neq t, f_\tau^i = 1}\Big(\text{Query}_t^i, \text{Key}_\tau^i, \text{Value}_\tau^i\Big).$$

We use the masked attention mechanism where at each time index, we only attend to all the training time steps except the current one. The computed attention values $\mathbf{g}_t^i$ are concatenated into the $\mathbf{z}_t^i$ vector as

$$\mathbf{z}_t^i = \text{NN}([\mathbf{g}_t^i \oplus \mathbf{h}_t^i \oplus \mathcal{E}_q(q_t^i) \oplus \mathcal{E}_o(c_t^i) \oplus \mathcal{E}_s(\{s_t^i\}) \oplus \mathbf{W}^T \text{DAS3H Features}]) \in \mathbb{R}^{4 \text{ or } 1}.$$

4 Experimental Results and Discussion

In this section, we discuss our local experimental results, public and private leaderboard results. Since the dataset is large, we initially experimented with a validation set that is 20% of the training set; however, we submitted predictions using only 5% as the validation set since this setting leads to better results. We select question embedding dimensions from $\{16,24,32\}$. We select dropout rates from $\{0.1,0.15, 0.2\}$. We set hidden dimension as 128 for recurrent neural network-based models. For NCF, we set the student embedding dimension to 128. We also used group ID, quiz ID, time difference, and confidence values; we use embedding layers (for categorical) and linear layers (for continuous) for these features and simply concatenate them with $\mathbf{x}_t$. For the response correctness prediction task, we simply replace the final softmax layer of the option predictor module (NN : $\cdot \to [0,1]^4$) with a sigmoid layer (NN : $\cdot \to [0,1]$) for all models; the resulting loss function corresponds to standard binary cross entropy loss.

Table 1 lists the performance of all methods on predicting unobserved options and response correctness. We observe that with static embeddings, NCF performs the worst among all the methods for both tasks. DAS3H features alleviate the problem of static embeddings to some extent On the response prediction task, NCF+DAS3H improves over NCF by Accuracy/AUC by $\sim 2\%$; however, the performance improvement in the option prediction task is marginal $\sim 0.5\%$. This observation suggests that DAS3H features are more helpful for the response prediction task than the option prediction task, which is not surprising since these features count only the correctness of past and future responses, hence providing little additional information on which distractor option a student would pick if they cannot select the correct option. On the other hand, recurrent neural network-based approaches such as LSTM perform significantly better. Moreover, since Bi-LSTM uses information from both the past and the future, it is able to outperform LSTM, especially on the response prediction task (by around 2%). Furthermore, additional features on top of the Bi-LSTM model, especially DAS3H, lead to small performance improvements over Bi-LSTM (around 0.1% on both tasks). Using attention modules provides an additional 0.1% improvement on the response prediction task.

Acknowledgement

This work is supported by the NSF under grant no. IIS-1917713.

References

[1] H. Cen, K. Koedinger, and B. Junker. Learning factors analysis–A general method for cognitive model evaluation and improvement. In *Proc. International Conference on Intelligent Tutoring Systems*, pages 164–175, June 2006.

[2] B. Choffin, F. Popineau, Y. Bourda, and J.-J. Vie. Das3h: Modeling student learning and forgetting for pptimally scheduling distributed practice of skills. In *Proc. International Conference on Educational Data Mining*, pages 29–38, July 2019.

[3] A. Corbett and J. Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-adapted Interaction*, 4(4):253–278, Dec. 1994.

[4] J. A. Erickson, A. F. Botelho, S. McAteer, A. Varatharaj, and N. T. Heffernan. The automated grading of student open responses in mathematics. In *Proceedings of the Tenth International Conference on Learning Analytics & Knowledge*, pages 615–624, 2020.

[5] A. Ghosh, N. Heffernan, and A. S. Lan. Context-aware attentive knowledge tracing. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2330–2339, 2020.

[6] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[7] A. Graves, A.-r. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. IEEE, 2013.

[8] M. Khajah, Y. Huang, J. González-Brenes, M. Mozer, and P. Brusilovsky. Integrating knowledge tracing and item response theory: A tale of two frameworks. In *Proc. International Workshop on Personalization Approaches in Learning Environments*, volume 1181, pages 7–15, Jan. 2014.

[9] A. S. Lan, D. Vats, A. E. Waters, and R. G. Baraniuk. Mathematical language processing: Automatic grading and feedback for open response mathematical questions. In *Proc. ACM Conference on Learning at Scale (L@S)*, pages 167–176, Mar. 2015.

[10] Z. A. Pardos and N. T. Heffernan. Modeling individualization in a Bayesian networks implementation of knowledge tracing. In *Proc. 18th Intl. Conf. on User Modeling, Adaptation, and Personalization*, pages 255–266, June 2010.

[11] P. Pavlik Jr, H. Cen, and K. Koedinger. Performance factors analysis–A new alternative to knowledge tracing. In *Proc. International Conference on Artificial Intelligence in Education*, pages 531–538, June 2009.

[12] C. Piech, J. Bassen, J. Huang, S. Ganguli, M. Sahami, L. J. Guibas, and J. Sohl-Dickstein. Deep knowledge tracing. In *Proc. Conference on Advances in Neural Information Processing Systems*, pages 505–513, Dec. 2015.

[13] D. A. Selent. *Creating Systems and Applying Large-Scale Methods to Improve Student Remediation in Online Tutoring Systems in Real-time and at Scale*. PhD thesis, Worcester Polytechnic Institute, 2017.

[14] M. V. Yudelson, K. R. Koedinger, and G. J. Gordon. Individualized bayesian knowledge tracing models. In *International conference on artificial intelligence in education*, pages 171–180. Springer, 2013.