


Implicit Heterogeneous Features Embedding in Deep Knowledge Tracing

Haiqin Yang¹  · Lap Pong Cheung²

Received: 10 July 2017 / Accepted: 31 October 2017 / Published online: 15 December 2017
© Springer Science+Business Media, LLC, part of Springer Nature 2017

Abstract Deep recurrent neural networks have been successfully applied to knowledge tracing, namely, deep knowledge tracing (DKT), which aims to automatically trace students' knowledge states by mining their exercise performance data. Two main issues exist in the current DKT models: First, the complexity of the DKT models increases the tension of psychological interpretation. Second, the input of existing DKT models is only the exercise tags representing via one-hot encoding. The correlation between the hidden knowledge components and students' responses to the exercises heavily relies on training the DKT models. The existing rich and informative features are excluded in the training, which may yield sub-optimal performance. To utilize the information embedded in these features, researchers have proposed a manual method to pre-process the features, i.e., discretizing them based on the inner characteristics of individual features. However, the proposed method requires many feature engineering efforts and is infeasible when the selected features are huge. To tackle the above issues, we design an automatic system to embed the heterogeneous features implicitly and effectively into the original DKT model. More specifically, we apply tree-based classifiers to predict

whether the student can correctly answer the exercise given the heterogeneous features, an effective way to capture how the student deviates from others in the exercise. The predicted response and the true response are then encoded into a 4-bit one-hot encoding and concatenated with the original one-hot encoding features on the exercise tags to train a long short-term memory (LSTM) model, which can output the probability that a student will answer the exercise correctly on the corresponding exercise. We conduct a thorough evaluation on two educational datasets and demonstrate the merits and observations of our proposal.

Keywords Recurrent neural networks · Knowledge tracing · Tree-based classifiers

Introduction

In the past forty years, several major paradigm shifts have appeared in the area of cognitive science and machine learning [5]. Now, deep learning reaches the new peak of the interest [32]. Deep learning, as the resurgence of neural networks, has attracted many research devotions due to its outstanding performance in various real-world applications and actual deployment in state-of-the-art systems for speech recognition [17], image classification [13, 30, 42], text classification [23], generating image captions [47], and even playing Go [41].

The effectiveness of deep learning is the result of its ability to automatically learn the representation from the input features without handcraft. The “deep” in deep learning refers to multiple layers of neurons (or feature representation) between the input and the output of the neural networks [16, 32]. The success of deep learning starts from deep belief nets [19] and quickly extends to other neural networks,

✉ Haiqin Yang
hxyang@ieee.org
Lap Pong Cheung
lpcheung@link.cuhk.edu.hk

¹ Department of Computing, Hang Seng Management College, Shatin, Hong Kong

² Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, Hong Kong

such as convolutional neural networks and recurrent neural networks [16, 30].

Recurrent neural networks (RNNs) are one of the effective tools in cognitive computing and machine learning to model dynamic temporal behaviors for sequences of inputs [40, 50]. Recently, deep RNNs such as the long short-term memory (LSTM) and the gated recurrent unit (GRU) have been intensively investigated and proved their effectiveness in solving various real-world applications such as speech recognition, language modeling, and knowledge tracing [8, 20, 34, 38].

Knowledge tracing is one of the significant research topics in educational data mining (EDM) [26, 31]. The goal is to capture students' knowledge states over time so that we can estimate their learning progress of mastering the required knowledge components [2, 9, 10]. Inferring students' knowledge states allows us to adapt to different learning progress and to recommend suitable personalized exercises or necessary teaching materials according to students' needs [1, 12, 44]. Recently, massive online open course (MOOC) platforms, such as Coursera, Edx, and Khan Academy, have provided high-quality open access online courses and attracted a large amount of enrolled users worldwide to enrol these courses [11]. The abundance of the data generated in these platforms enables researchers to investigate and monitor the learning process of students [26, 28, 31].

In the literature, the proposed methods for knowledge tracing can be divided into two main streams: One is Bayesian Knowledge Tracing (BKT) [3, 10, 36], which applies a Hidden Markov Model (HMM) to model the knowledge components. The hidden states are updated according to each student's responses to the exercises. Another approach is Deep Knowledge Tracing (DKT) [38, 49], which utilizes deep recurrent neural networks to discover the hidden structure of the correlation of exercises by analyzing students' responses to the exercises. In the debut, it is shown that DKT can achieve 25% gain in the area under the ROC curve (AUC) score over BKT [38]. More variants of DKT models are further investigated in [25, 48, 52].

There are still two main issues in the previously proposed DKT models: First, the complexity of the DKT models increases the tension of psychological interpretation of the models. How to propose a model with sufficient psychological interpretation is favorable in cognition science [43, 51]. Second, the input of DKT models is only a one-hot encoding of the exercise tags [38]. It excludes many rich and informative features, such as the exercise title, the number of attempts to answer, and the duration time of answers. These features are heterogeneous and exhibit different characteristics of students' learning procedure. They not only provide additional information on exercises but also capture students' learning procedure. Analyzing and utilizing

them properly will help to trace students' learning states. Currently, researchers mainly focus on incorporating different types of features in the learning. The proposed features capture different aspects of students' learning procedure, such as measuring the effect of students' individual characteristics, assessing the effect of help in a tutor system, controlling the difficulty level of exercises, and measuring the effect of subskills [24, 53]. In [53], a manual method is proposed to analyze the features and to select appropriate feature subsets. The selected features are then discretized based on the statistics of the features and the semantic meaning is learned via an autoencoder. The manual feature engineer effort offers further improvement in knowledge tracing but is still restricted in two aspects: First, they require sufficient domain knowledge to understand the data. This may introduce bias when practitioners cannot fully explore the data. Second, they are infeasible to extend the method to huge feature size.

To tackle the above issues, we propose an automatic and intelligent approach to integrate the heterogeneous features into the DKT model. More specifically, we conduct a pre-processing step via the tree-based classifiers due to their effectiveness and interpretation power [39]. We then apply the tree-based classifiers to predict whether a student can answer an exercise correctly given the heterogeneous features. After that, we encode the predicted response and the true response into binary codes and concatenated them with the original one-hot encoding features as the input to train a LSTM model. Although the pre-processing step is simple, it can provide additional information of students' learning behaviors, especially, how a student deviates from others in the learning process.

We highlight the contributions of this article in the following:

- We have proposed an automatic and effective way to pre-process the heterogeneous features based on tree-based classifiers. The splitting features can provide us insight into the characteristics of students' learning behaviors.
- We present a systematic framework to incorporate the learned response, the true response, and the original one-hot encoding features to train an LSTM model. The output can produce the predicted probability whether a student will answer the next exercise correctly. The learned response given the heterogeneous features allows us to exploit students' learning behaviors.
- We conduct a thorough evaluation on two educational datasets and demonstrate the effectiveness and merits of our proposal.

The rest of the paper is organized as follows: In “[Related Work](#),” we review the related work in this paper, including techniques for knowledge tracing and the tree-based

classifiers we adopt. In “**Methods**,” we detail the overall architecture of our proposal, especially, how the heterogeneous features are learned and incorporated. In “**Experiments**,” we present the educational datasets, the experimental setup, and the experimental results with detailed explanation. Finally, in “**Conclusions**,” we conclude the whole paper with some remarks.

Related Work

Knowledge Tracing

Knowledge tracing is an important research topic in EDM, aiming to capture the students’ knowledge learning states based on their performance on the exercises. BKT is a dominant approach in the field, where the knowledge components are denoted as a series of binary variables in the hidden states modeled by a HMM [3, 10, 36]. The hidden states are updated according to each student’s responses to the given exercises. Researchers then extend the HMM model to explore different latent factors [15, 27, 37].

DKT [38] is a breakthrough to leverage a vanilla RNN or an LSTM model to solve this task. In the debut, it is shown that DKT can achieve 25% gain in the AUC score over BKT. Though later, some researchers argue that, with suitable extensions, BKT can achieve comparable performance with DKT [26]. Due to the good performance of DKT, variants of DKT models are accordingly proposed [48, 52, 53]. For example, a memory-augmented neural network replaces the LSTM in the original DKT to capture the long-term dependence and extended to utilize the key-value mechanism to store the concept representation and students’ understanding state of each concept, respectively [52].

Some researchers also try to include heterogeneous features to improve the model performance of knowledge tracing [24, 53]. The existing publications show that by employing these heterogeneous features properly, they indeed can further improve the model performance. However, the existing work contains the deficiency of handcraft or not fitting in the DKT architecture. This motivates us to further explore the heterogeneous features and to exploit them in the DKT architecture.

Tree-Based Classifiers

Decision trees are one type of the most popular data mining algorithms for classification and decision-making [39]. They utilize the entropy to split features and have triggered a family of feature discretization and feature selection techniques [14, 29]. The processes of feature selection and feature discretization embedded in the training procedure via decision trees have shown that they can largely reduce

the engineering effort [45]. Moreover, the learned features are meaningful and interpretable, which motivates us to include them in our proposal.

Among various decision trees, Classification and Regression Trees (CART) [39] have exhibited several significant advantages: First, they can handle both numerical and categorical features. Second, they can handle outliers properly [46] and avoid overfitting [35]. Hence, we apply CART in [7]. After observing the power of CART, we decide to further explore other tree-based classifiers such as random forest [4] and the Gradient Boosted Decision Tree (GBDT) [18]. These two methods are both ensemble classifiers, where random forest consists of weak decision trees to reduce the overall performance variance and the GBDT is a linear combination of weak learners greedily to improve the performance of the entire ensemble. Due to their power of retaining robustness and improving the predictive performance of solving classification applications [4, 18], we apply these two methods in the rest.

Methods

Figure 1 illustrates the overall architecture of our proposed model, Deep Knowledge Tracing with tree-based classifiers, where CART is illustrated as the representative tree-based classifier. It can be replaced by other tree-based classifiers, e.g., random forest or GBDT. In Fig. 1, the bottom part is the pre-processing procedure on the heterogeneous features. These features are learned by CART to predict whether a student will answer the exercise correctly and output the predicted response. The predicted response and the true response are encoded into a 4-bit one-hot encoding. They are concatenated with the original one-hot encoding on the exercise tag as a new input. This new input is fed into an LSTM [20] to learn the similarity of exercises and trace the knowledge components mastered by the students. It is noted that Fig. 1 shows a vanilla RNN for simplicity, where we deploy an LSTM in our evaluation.

Input and Output

Consider a specific student practicing an exercise at the t th time stamp, let e_t and a_t be the exercise tag and the heterogeneous features, respectively. The notation $c_t = 1$ implies that the student will answer the exercise correctly while $c_t = 0$ for an incorrect answer. As shown in Fig. 1, CART will take a_t as the input and tries to predict whether the student will correctly answer the exercise given these heterogeneous features. The corresponding predicted response is then denoted by a'_t .

All features are represented into the binary representation denoted by $O(\cdot, \cdot)$, where $O(e_t, c_t) \in \{0, 1\}^M$ is the

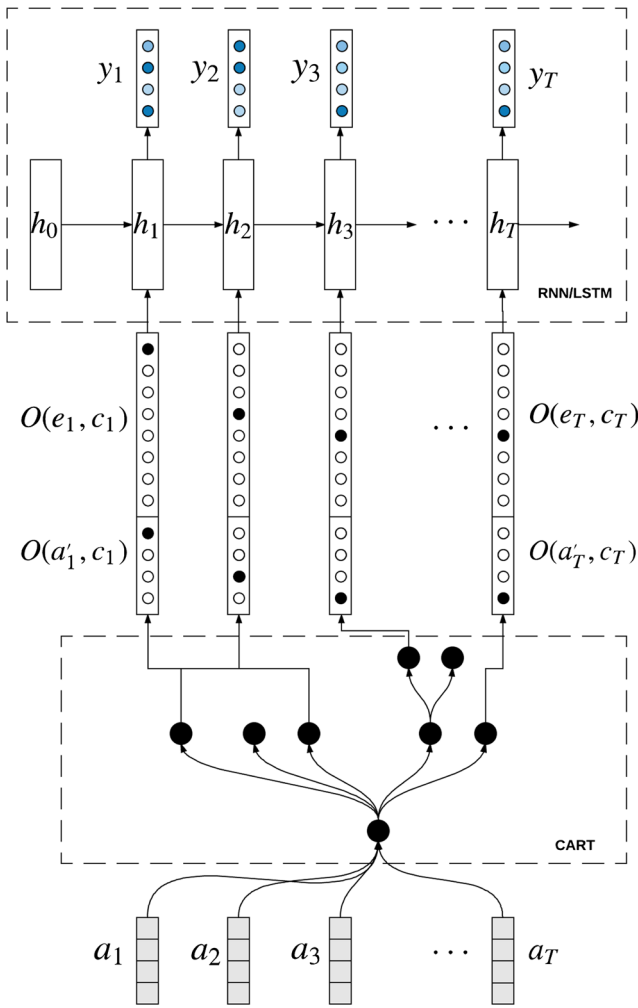


Fig. 1 The architecture of our proposal with the 4-bit one-hot encoding for responses consists of three parts: (1) heterogeneous features learned via tree-based classifiers, (2) feature concatenation, and (3) model training and prediction by an RNN/LSTM. The solid large black nodes indicate the splitting of features on different sub-branches of the trees. The dots in the black color, the white color, and the gray color in a vector indicate the values of 1, 0, and the probability of the prediction, respectively

original one-hot encoding for an exercise tag with the number of exercises being M and $O(a'_t, c_t)$ is the 4-bit one-hot encoding. Hence, the total size of $O(e_t, c_t)$ and $O(a'_t, c_t)$ is $2M + 4$. For $O(e_t, c_t)$, all elements are zeros except that 1 will be denoted at the i th index when the answer of the i th

exercise is correct; otherwise, 1 will be placed at the $i + M$ -th index. $O(a'_t, c_t)$ is constructed by the predicted response learned by the tree-based classifiers and the true response, which is defined in Table 1.

After concatenating $O(e_t, c_t)$ and $O(a'_t, c_t)$, we feed it as a new input of LSTM x_t to train the corresponding model and output a vector $y_t \in \mathbb{R}^M$ for predicting the probability that whether a student will answer the question correctly. In the level of RNN/LSTM in Fig. 1, different color grades in the nodes of y_t represent different levels of the probability, where dark colors represent higher probabilities while light colors represent lower probabilities.

Models

Tree-Based Classifiers

We apply the following tree-based classifiers, CART [39], random forest [4], and GBDT [18], to automatically partition the feature space and output the predicted response whether a student will correctly answer an exercise. We briefly introduce these three models in the following.

At each node, CART continuously conducts binary partitioning to group the interaction of the same class by maximizing the gini index or information gain. Here, we take information gain as an example in the following formulation. Given a set S at a node contains training data $a_t \in \mathbb{R}^n$ and the corresponding labels $c_t \in \{0, 1\}$, CART partitions the data into two subsets

$$S_l = \{(a_t, c_t) | a_{t,j} < t\}, \quad \text{and} \quad S_r = S \setminus S_l$$

where j is the splitting variable and t is the threshold determined by minimizing the information gain defined as follows:

$$(j^*, t^*) = \arg \min_{j,t} G(S, j, t) \triangleq \frac{|S_l|}{|S|} H(S_l) + \frac{|S_r|}{|S|} H(S_r),$$

where $|\cdot|$ defines the Cardinality of the set, i.e., the number of elements in the set. $H(\cdot)$ defines the impurity measured by the cross entropy, and $G(\cdot)$ denotes the gini index or information gain. For a region R with N observations, the cross entropy H is defined by

$$H(X) = - \sum_k p_k \log(p_k), \quad \text{where} \quad p_k = \frac{1}{N} \sum_{a_t \in R} I(c_t = k)$$

Table 1 Explanation of the notation of $O(a'_t, c_t)$

Symbol	Meaning
1000	A true positive prediction: a correctly answered exercise is predicted as a correctly answered exercise.
0010	A false positive prediction: a wrongly answered exercise is predicted as a correctly answered exercise.
0100	A false negative prediction: a correctly answered exercise is predicted as an incorrectly answered exercise.
0001	A true negative prediction: a wrongly answered exercise is also predicted as an incorrectly answered exercise.

In binary classification, k is the label set, which can be selected from $\{0, 1\}$, or $\{-1, +1\}$.

By minimizing the cross entropy, CART learns a set of classification rules. At time t , the heterogeneous feature a_t is fed into the root of CART and follows the path assigned by the classification rules until getting a predicted response a'_t .

Random forest learns an ensemble of decision trees by growing a bag of trees with the bootstrap sample and variable subsets [4]. Suppose there are B trees in the bag, for each tree, a bootstrap sample \mathbf{Z}^* of size N is drawn. At each node of the corresponding tree T_b , a subset of variables is selected at random and is split according to the same criteria of CART until the maximum node size is reached. The class prediction $\mathbf{hat}C^B(a_t) = \text{majority vote}\{a_t^b\}_{b=1}^B$, where a_t^b is the class prediction of a particular tree b .

GBDT is also a tree-based ensemble method and achieves better performance by reducing bias rather than variance [18]. The ensemble is learned by adding a CART iteratively which minimizes the following objective function:

$$H^{(t)} = \sum_{i=1}^n l(y_i, \hat{y}_i^{(t)}) + \sum_{i=1}^t \Omega(f_i)$$

where n is the number of training samples, $l(\cdot)$ is the loss function, and $\Omega(f_i)$ is the regularization term. $\hat{y}_i^{(t)}$ is the predicted value for the i -th sample at the t -th step and is defined by: $\hat{y}_i^{(0)} = 0$, $\hat{y}_i^{(t)} = \sum_{k=1}^t f_k(a_i) = \hat{y}_i^{(t-1)} + f_t(a_i)$, f_k is the CART learned at the k -th step and $f_k(a_i)$ is the corresponding predicted value on the feature a_i .

It is noted that the information of heterogeneous features is therefore implicitly captured by a'_t via the predicted response, which embeds the information of how a student deviating from others in the exercises. The personalized information is then utilized in the DTK models.

Recurrent Neural Networks

A recurrent neural network is a neural network that simulates a discrete-time dynamical system that has an input \mathbf{x}_t , an output \mathbf{y}_t , and a hidden state \mathbf{h}_t , where t represents time. The dynamical system is defined by

$$\mathbf{h}_t = f_h(\mathbf{x}_t, \mathbf{h}_{t-1}) \tag{1}$$

$$\mathbf{y}_t = f_o(\mathbf{h}_t), \tag{2}$$

where f_h and f_o are a state transition function and an output function, respectively. Each function is parameterized by a set of parameters: θ_h and θ_o .

LSTM is proven to be an efficient RNN in modeling long-term dependency through a collection of cells and gates. The cell states are controlled by gates to decide whether to store or remove the information, which facilitates complex interaction of current input and history. The

hidden state and output are updated by the following set of equations:

$$i_t = \sigma(\mathbf{W}_{xi}\mathbf{x}_t + \mathbf{W}_{hi}\mathbf{h}_{t-1} + \mathbf{W}_{ci}c_{t-1} + b_i) \tag{3}$$

$$f_t = \sigma(\mathbf{W}_{xf}\mathbf{x}_t + \mathbf{W}_{hf}\mathbf{h}_{t-1} + \mathbf{W}_{cf}c_{t-1} + b_f) \tag{4}$$

$$c_t = f_t c_{t-1} + i_t \tanh(\mathbf{W}_{xc}\mathbf{x}_t + \mathbf{W}_{hc}\mathbf{h}_{t-1} + b_c) \tag{5}$$

$$o_t = \sigma(\mathbf{W}_{xo}\mathbf{x}_t + \mathbf{W}_{ho}\mathbf{h}_{t-1} + \mathbf{W}_{co}c_{t-1} + b_o) \tag{6}$$

$$\mathbf{h}_t = o_t \tanh(c_t) \tag{7}$$

where σ is the sigmoid function. The components of LSTM, denoted as i, f, c , and o , are input gate, forget gate, cell activation vector, and output gate, respectively.

In training an RNN/LSTM, we are given a set of N training sequences $D = \{(\mathbf{x}_i^{(n)}, \mathbf{y}_i^{(n)})\}_{i=1}^{T_n}$, where $n = 1, \dots, N$, and estimate the parameters by minimizing the following objective function:

$$J(\theta) \propto \sum_{n=1}^N \sum_{t=1}^{T_n} \mathcal{L}(\mathbf{y}_t^{(n)}, f_o(\mathbf{h}_t^{(n)})), \tag{8}$$

where θ defines all variables of \mathbf{W} 's and b 's defined in Eqs. (3)–(7). $\mathcal{L}(\mathbf{a}, \mathbf{b})$ is a predefined divergence measure between \mathbf{a} and \mathbf{b} , such as Euclidean distance or cross-entropy.

The input $\mathbf{x}_t = [O(e_t, c_t), O(a'_t, c_t)]$, capturing students' exercise performance, is fed into an LSTM to learn the hidden structure of the sequence of exercises, which represents the knowledge components. The hidden cell \mathbf{h}_t is then passed to a fully connected layer via a sigmoid activation function to get the output \mathbf{y}_t , which denotes the predicted probability of whether a student will correctly answer the next exercise. In this setting, we will predict the probability for all M exercises because we do not know which one is the next exercise.

Prediction

In the test, the average loss is computed by the binary cross-entropy defined as follows:

$$L = \frac{1}{N} \sum_{n=1}^N \sum_{t=t_0^n}^{t_0^n+T^n} \mathbf{c}_{t+1}^n \log \hat{\mathbf{y}}_t^n + (1 - \mathbf{c}_{t+1}^n) \log(1 - \hat{\mathbf{y}}_t^n),$$

where N is the number of students, t_0^n is the starting index for the n th student in the test set, and T^n is the number of exercises for the student. The predicted value $\hat{\mathbf{y}}_t^n$ is the inner product of predicted output and the one-hot encoding of the exercises conducted by the student n , i.e., $\hat{\mathbf{y}}_t^n = \mathbf{y}_t^n \top O(\mathbf{e}_{t+1}^n)$ because $\hat{\mathbf{y}}_t^n$ can output the corresponding predicted probability of whether the student n can answer the question correctly in the next time stamp.

Experiments

In the experiments, we address the following issues:

- 1) What is the performance of our proposal compared to the baseline methods?
- 2) What is the effect of the encoding scheme?
- 3) What is the importance of the features learned?
- 4) What is the effect of tree-based classifiers to DKT and the visualized trees?

“**Model Comparison**”–“**Effect of Tree-Based Classifiers and Tree Results**” will answer the above questions accordingly.

Datasets

In the following, we conduct experiments on two popular educational datasets collected from the computer-based online learning platforms [7, 53]. The datasets are:

- **ASSISTments** 2009-2010¹ [53]: The dataset consists of 4,151 students exercising on 124 knowledge components with 332,343 interactions (records). It is also called the mastery learning data because a student is considered mastering a skill when meeting certain criteria.
- **Junyi** academy² [6]: The dataset is crawled from a Chinese e-learning platform established on the basis of the open-source code released by the Khan Academy. The dataset contains students’ exercises in mathematics. We select 1,000 most active students from the exercise log, which yields 666 knowledge components and 971,402 records.

Table 2 summarizes the basic statistics of the datasets, including the number of users, skills (knowledge components), records (exercise interactions), and the used heterogeneous features. In the following, **ASSISTments** and **Junyi** are bold to denote the corresponding dataset. In **ASSISTments**, we use the following 12 features:

- **original**: a binary feature records whether the exercise is the main problem or a scaffolding problem, i.e., whether the original problem is broken into several steps.
- **attempt_count**: a numerical value records the number of attempts (times) a student tries to answer the exercise.
- **ms_first_response**: a numerical value records the time in milliseconds between the start time and the first action from the student, e.g., asking for the hints or entering an answer.

¹<https://sites.google.com/site/assistmentsdata/home/assignment-2009-2010-data/skill-builder-data-2009-2010>

²<https://pslcdatashop.web.cmu.edu/DatasetInfo?datasetId=1198>

Table 2 Summary of the datasets

Summary	ASSISTments	Junyi
No. of users	4,151	1,000
No. of skills	124	666
No. of records	332,343	971,402
No. of features	12	10

- **answer_text**: a categorical feature records whether the answer is entered by the student or the value is selected in a multiple choice.
- **assistments_position**: the position of the assignment on the class assignments page.
- **type (problem_set_type)**: the organization of contents in the problem set. It has three classes:
 - Linear: Student completes all problems in a pre-determined order.
 - Random: Student completes all problems, but each student is presented with the problems in a different random order.
 - Mastery: Random order, and student must “master” the problem set by getting a certain number of questions correct in a row before being able to continue.
- **hint_count**: an integer records the number of the hints a student requests in practicing the exercise.
- **hint_total**: an integer records the number of possible hints on the problem. Note that each problem has a different number of hints.
- **overlap_time**: a numerical value records the time in milliseconds for the student to complete the problem.
- **first_action**: a categorical feature records the first action the student performs to the problem: 0 for attempting to solve it, 1 for asking for the hint, 2 for scaffolding, and 3 for doing nothing.
- **opportunity**: an integer records the number of opportunities the student has to practice the skill.
- **tutor mode**: a categorical feature indicating whether the exercise is in the tutor mode, the test mode, the pre-test mode, or the post-test mode.

In **Junyi**, there are ten heterogeneous features:

- **problem number**: an integer records how many times the student practices the exercise. For example, the value is 1 if the student tries to answer the exercise at the first time.
- **topic mode**: a binary feature records whether the student is assigned this exercise by clicking the topic icon.
- **suggested**: a binary feature records whether the exercise is suggested by the system according to prerequisite relationships on the knowledge map.

- **review mode**: a binary feature records whether the exercise is done by the student after he/she earns proficiency.
- **time**: a numerical value records the time (in seconds) of a student spending on the exercise.
- **time taken**: a numerical value records the total number of seconds the student spends on this exercise.
- **attempt counts**: an integer records how many times the student attempts to answer the problem.
- **hints used**: a binary feature records whether the student requests hints.
- **count hints**: an integer records how many times the student requests hints.
- **earn proficiency**: a binary feature records whether the student reaches proficiency. Please refer to [21] for the algorithm of computing proficiency.

Both datasets also contain other non-numerical and non-categorical features, which are removed in the experiment. Detailed explanation about the features can be referred to the following two links:

- <https://sites.google.com/site/assistmentsdata/how-to-interpret> and
- <https://pslcdatashop.web.cmu.edu/Files?datasetId=1198>.

Model Comparison

A fivefold student level cross-validation is conducted in the test. The results are evaluated by the area under the ROC curve (AUC) and R^2 , two standard metrics for evaluating the predicted performance [22, 38, 53]. The following models with different feature processing are compared:

- **Deep Knowledge Tracing (DKT)** [38]: the input feature is the one-hot encoding of the exercise tags.
- **DKT with Feature Engineering (DKT-FE)** [53]: Feature engineering has been conducted by manually selecting a subset of heterogeneous features and discretizing them by a certain pre-determined criterion while reducing the dimensionality of the input via autoencoder. The learned feature is concatenated with

the one-hot encoding of the exercise tags as the input.

- **DKT without Feature Engineering (DKT-W)**: The selected heterogeneous features are the same as those of DKT-FE but without any further feature processing. The selected feature is directly concatenated with the one-hot encoding of the exercise tags as the input.
- **DKT with Decision Trees (DKT-CART)**: The selected heterogeneous feature is learned by CART to output the predicted response, which is represented by a 2-bit binary code, and concatenated with the 2-bit binary code of the true response and the one-hot encoding as the input of the LSTM.
- **DKT with Random Forest (DKT-RF)**: The selected heterogeneous feature is learned by the random forest to predict the corresponding response. The setting of the input is the same as that of DKT-CART.
- **DKT with GBDT (DKT-GBDT)**: The selected heterogeneous feature is learned by the corresponding response. The setting of the input is the same as that of DKT-CART.

For the LSTM, we set the hidden dimension to 200 and train it via the stochastic gradient descent on the size of a mini-batch being 5. Other parameters are set as default in the Tensorflow. For different tree-based classifiers, the parameters are set as default in the Python toolbox, scikit-learn.

Table 3 reports the results of all four compared methods. From the results, we have the following observations:

- Our proposed DKT with tree-based classifiers attains significantly better performance over other methods in terms of both the AUC and R^2 metrics on both datasets. Especially, our proposed DKT-CART attains 13% gain over DKT in the R^2 metric.
- For models of the DKT with tree-based classifiers, DKT-GBDT attains the best performance in both datasets among all three compared methods, while DKT-RF gets the second best performance. The results show that ensemble methods can further improve the model performance in these two datasets.

Table 3 Experimental results on the compared models. The predicted and true responses are encoded into 4-bits in the proposed models

Model	ASSISTments		Junyi	
	AUC (%)	R^2	AUC (%)	R^2
DKT	73.8 ± 0.7	0.161 ± 0.010	72.5 ± 0.4	0.076 ± 0.014
DKT-FE	73.1 ± 1.0	0.163 ± 0.010	68.8 ± 0.5	0.039 ± 0.004
DKT-W	60.9 ± 0.2	0.010 ± 0.012	70.0 ± 0.6	0.052 ± 0.005
DKT-CART	74.2 ± 0.7	0.172 ± 0.014	72.9 ± 0.5	0.090 ± 0.007
DKT-RF	74.4 ± 0.7	0.171 ± 0.014	73.2 ± 0.8	0.092 ± 0.025
DKT-GBDT	74.7 ± 0.5	0.181 ± 0.010	73.3 ± 0.5	0.093 ± 0.010

The Italic entries in the tables are the highest outputs

- An interesting observation is that including heterogeneous features without appropriately pre-processing degrades the performance of DKT. We conjecture this may be due to the introduction of noise, which intervenes DKT to extract the similarity between exercises.
- The degrading effect of DKT-W is highly dependent on the size of the dataset. The size of the Junyi dataset is much larger than that of ASSISTments and it may help to relieve the effect of training the LSTM.
- The performance of DKT-FE is a slightly poor than that of DKT-W in the Junyi dataset. The reason is that we adopt the same criterion to process the feature as the ASSISTments data shown in [53]. The provided criterion is not extensible to the new Junyi dataset.
- Overall, the experimental results show that including additional features may improve the prediction accuracy, but it requires properly pre-processing.

Effect of Encoding Scheme

One issue is that the predicted and true responses can be encoded into 2-bit instead of 4-bit as the designed setting of one-hot encoding of the exercise tags. To test the effect of these two settings, we change the inputs and conduct the experiments accordingly.

Tables 4 and 5 report the results of the compared results with respect to the number of encoded response units, respectively. We can observe that

- In **ASSISTments**, DKT-RF and DKT-GBDT achieve better performance than DKT-CART in both the AUC and R^2 metrics. It shows that ensemble classifiers can further improve the model performance.
- In **Junyi**, the performance is not significantly different. DKT-CART with 2-bit units and DKT-GBDT with 4-bit units attain the best performance in the AUC metric, while DKT-CART earns a little better than DKT-GBDT in terms of the R^2 metric.

Generally speaking, 4-bit units encoding can get better or at least comparable performance than the scheme of

Table 4 Experimental results of the DKT model with different tree-based classifiers on **ASSISTments** with respect to the number of encoded response units

Model	4 units		2 units	
	AUC (%)	R^2	AUC (%)	R^2
DKT-CART	74.2 ± 0.7	0.172 ± 0.014	72.9 ± 0.7	0.156 ± 0.014
DKT-RF	74.4 ± 0.7	0.171 ± 0.014	73.3 ± 0.6	0.160 ± 0.008
DKT-GBDT	74.7 ± 0.5	0.181 ± 0.010	74.4 ± 0.6	0.178 ± 0.010

The Italic entries in the tables are the highest outputs

Table 5 Experimental results of the DKT model with different tree-based classifiers on **Junyi** with respect to the number of encoded response bits

Decision Tree	4 units		2 units	
	AUC (%)	R^2	AUC (%)	R^2
DKT-CART	72.9 ± 0.5	0.090 ± 0.007	73.3 ± 0.5	0.097 ± 0.007
DKT-RF	73.2 ± 0.8	0.092 ± 0.025	73.1 ± 0.5	0.092 ± 0.003
DKT-GBDT	73.3 ± 0.5	0.093 ± 0.010	73.1 ± 0.8	0.087 ± 0.015

The Italic entries in the tables are the highest outputs

2-bit units encoding. We hypothesize that it may be the consequence of “division of labor.” As the input x_t is the concatenation of two one-hot encoding vectors, it selects and adds two columns in the input-to-hidden weight matrix, which contributes to the updating of hidden layer. If the scheme of the 4-bit unit encoding is applied, the last four columns in the input-to-hidden weight matrix would be dedicated to learn how the cross-effect of the predicted response and the true response to the contribution of knowledge tracing. Meanwhile, the original one-hot encoding features are assigned to simply learn the effect of accumulating proficiency through exercises. In contrast, for the scheme of the 2-bit unit encoding, no column in the weight matrix is assigned to learn the cross-effect. The original one-hot encoding features have to learn not only the accumulation of proficiency but also the cross-effect, which increases the learning complexity.

Importance of the Features

Tree-based classifiers have the power to evaluate the importance of features. In [4, 33], the importance of a variable can be measured by mean decrease impurity (MDI). That is, the importance of a variable X_m for predicting Y is measured by adding up the weighted impurity decreases $p(t)\Delta i(s_t, t)$ for all nodes t where X_m is involved, averaged over all N_T trees:

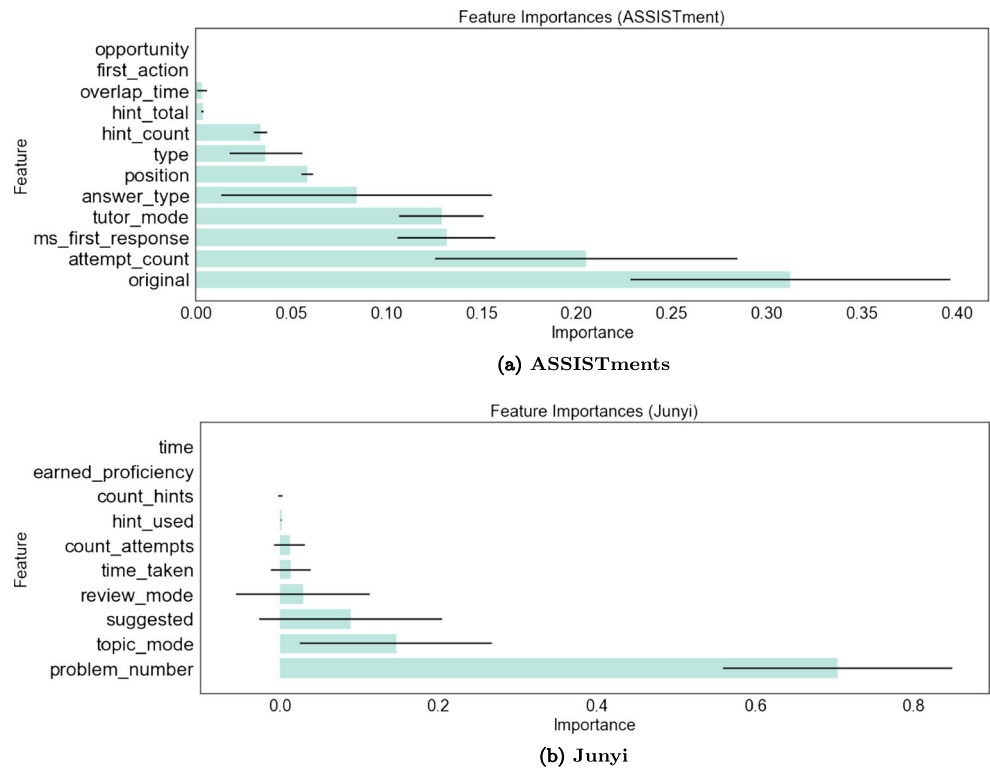
$$Importance(X_m) = \frac{1}{N_T} \sum_T \sum_{t \in T: v(s_t)=X_m} p(t)\Delta i(s_t, t)$$

and $\Delta i(s, t) = i(t) - p_L i(t_L) - p_R i(t_R)$, where $p(t)$ is the proportions $\frac{N_t}{N}$ of samples reaching t and p_L, p_R are $\frac{N_{tL}}{N_t}, \frac{N_{tR}}{N_t}$, the proportions of samples in the left and right node, respectively. $v(s_t)$ is the variable involved in split s_t . $i(t)$ can be any impurity measure.

Figure 2 shows the importance of the features in random forest for **ASSISTments** and **Junyi**, respectively. We can observe that

- Figure 2a shows that the features of **ASSISTments** are grouped into five sectors based on their importance.

Fig. 2 Feature importance measured on both test datasets. The importance of variables are evaluated by mean decrease impurity. **a ASSISTments.** **b Junyi**



The feature, original, is the most important with a mean over 0.3. Following it are the features of attempt_count, ms_first_response, tutor_mode, and answer_type, which get a mean weight around 0.12. The third sector consists of the features of position, type, and hint_count, which get a mean weight of 0.05. The fourth sector consists of the features of hint_total and overlap_time. The final sector consists of the features of first_action and opportunity, which show no importance for the classification.

- Figure 2b shows that the features of **Junyi** are grouped into four sectors based on their importance. The feature of problem_number gains an important weight over 0.7. The second sector consists of the features of topic_mode and suggested which exhibits a mean weight around 0.1. The third sector consists of the features of review_mode, time_taken, and count_attempts, whose mean weights are less than 0.1. The rest four features, hint_used, count_hints, earned_proficiency, and time, are in the final sector and contain negligible weights.

Overall, the importance of the features can provide us guidance to understand the data and extract meaningful information to improve model performance.

Effect of Tree-Based Classifiers and Tree Results

We also record the performance of the tree-based classifiers predicting the correctness of the student’s answer to the current exercise in Tables 6 and 7 for **ASSISTments** and **Junyi**, respectively. The results show that the adopted tree-based classifiers usually attain good performance on the prediction at the current state. The ensemble classifiers can get better performance than CART and accordingly achieve better performance in the corresponding DKT models.

Another advantage of tree-based classifiers is its interpreting ability for the results. The decision tree can be visualized such that teachers and researchers can extract the latent factors which affect the probability of correct prediction. Figure 3 shows parts of the tree learned by CART on

Table 6 Prediction accuracy of different tree-based classifiers given the heterogeneous features on **ASSISTments**

Classifier	AUC (%)	R ²
CART	88.8 ± 3.7	0.526 ± 0.14
RF	<i>94.1 ± 2.4</i>	<i>0.544 ± 0.13</i>
GBDT	93.56 ± 2.4	0.463 ± 0.08

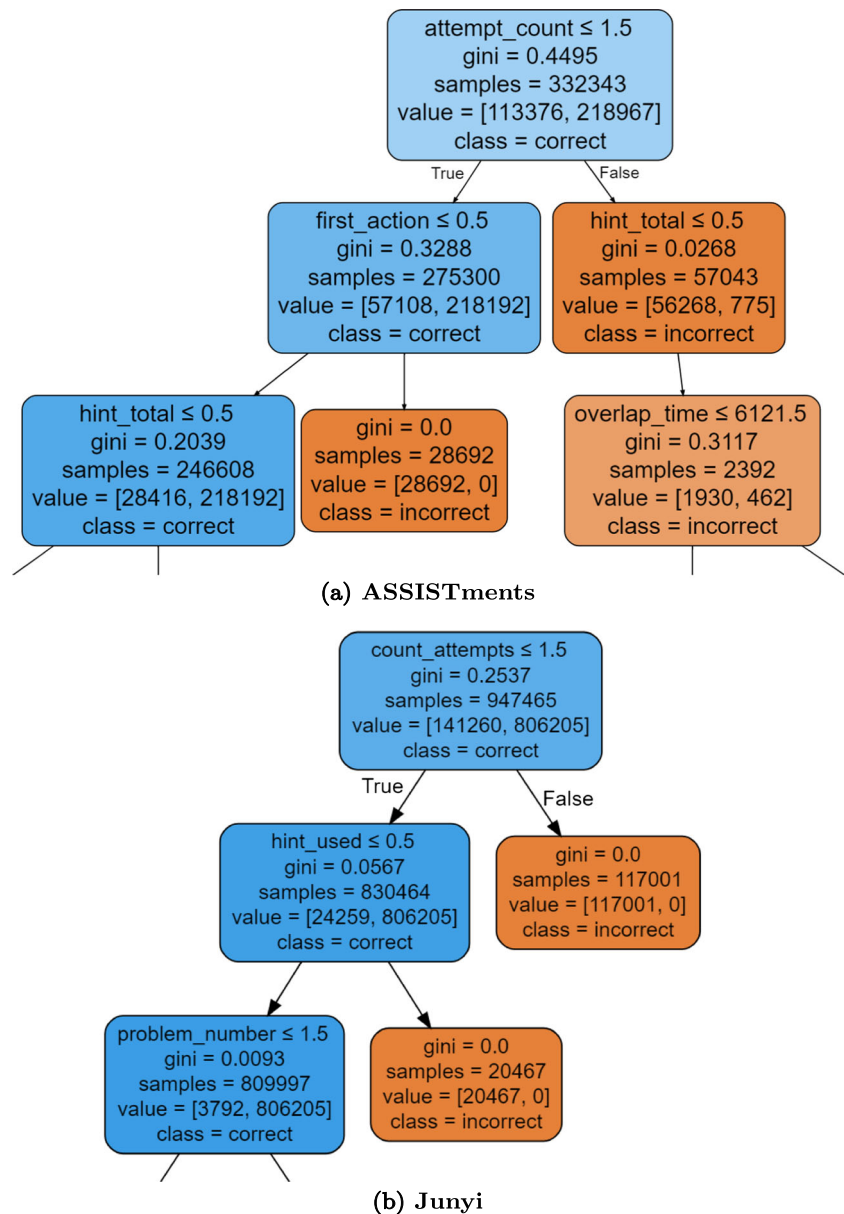
The Italic entries in the tables are the highest outputs

Table 7 Prediction accuracy of different tree-based classifiers given the heterogeneous features on **Junyi**

Classifier	AUC (%)	R ²
CART	99.0 ± 0.0	0.965 ± 0.0
RF	<i>99.4 ± 0.0</i>	<i>0.968 ± 0.0</i>
GBDT	99.2 ± 0.0	0.968 ± 0.00

The Italic entries in the tables are the highest outputs

Fig. 3 Parts of the trees learned by CART on ASSISTments and Junyi. The color of a block indicates the majority class, i.e., the class with more training samples, in that node: the blue color denotes that the sample is correctly assigned to the majority class and the red color for incorrect assignment. The light of the color implies the value of the gini coefficient, a lighter one for a larger gini coefficient. In each block, the first row denotes the selected feature and its splitting threshold. “gini” stands for the gini coefficient. “samples” means the total number of samples being assigned and classified in that node. “value” denotes the number of samples in each class. The last row “class” indicates whether the current block is in the correct state



both datasets. The learned trees can then be further analyzed to understand the importance of the features.

Conclusions

We have proposed an effective method to pre-process the heterogeneous features and integrate the learned feature implicitly to the original deep knowledge tracing model. The pre-processing step is conducted by tree-based classifiers, i.e., CART, random forest, and gradient boosting decision tree, to output the predicted response that a student will correctly answer the current exercise given the heterogeneous features. This allows us to capture students' behaviors on the exercises and to provide an good initialization

to the DKT model. Our experiments on two educational datasets demonstrate the effectiveness and merits of our proposal.

Some interesting future work can be considered. For example, we do not fully utilize the importance of the features. How to include such information in the RNN models is a significant research topic. The current setting only predicts exercises in a fixed set. It is valuable to explore the current model to provide the personalized recommendation for students to select appropriate exercises and to conduct selective practice.

Funding Information The work described in this paper was partially supported by the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. UGC/IDS14/16).

Compliance with Ethical Standards

Ethical Approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

- Agrawal R. Data-driven education: Some opportunities and challenges. In: EDM; 2016. p. 2.
- Ayers E, Nugent R, Dean N. A comparison of student skill knowledge estimates. In: EDM; 2009. p. 1–10.
- Baker RS, Corbett AT, Alevan V. More accurate student modeling through contextual estimation of slip and guess probabilities in Bayesian knowledge tracing. In: Proceedings of the 9th International Conference on Intelligent Tutoring Systems, ITS 2008, Montreal, Canada, June 23–27; 2008. p. 406–415.
- Breiman L. Random forests. *Mach Learn*. 2001;45(1):5–32.
- Cambria E, Hussain A. Sentic computing. *Cogn Comput*. 2015; 7(2):183–5.
- Chang H, Hsu H, Chen K. Modeling exercise relationships in e-learning: a unified approach. In: EDM; 2015. p. 532–535.
- Cheung LP, Yang H. Heterogeneous features integration in deep knowledge tracing. In: ICONIP; 2017.
- Chung J, Gulcehre C, Cho K, Bengio Y. Gated feedback recurrent neural networks. In: ICML; 2015. p. 2067–2075.
- Corbett AT, Anderson JR. Knowledge tracing: modelling the acquisition of procedural knowledge. *User Model User-adapt Interact*. 1995;4(4):253–78.
- Corbett AT, Anderson JR. Knowledge tracing: modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction*. 1994.
- Czerniewicz L, Deacon A, Glover M, Walji S. MOOC—making and open educational practices. *J Comput High Educ*. 2017;29(1):81–97.
- Desmarais MC, Villarreal A, Gagnon M. Adaptive test design with a naive bayes framework. In: EDM; 2008. p. 48–56.
- Gao F, Zhang Y, Wang J, Sun J, Yang E, Hussain A. Visual attention model based vehicle target detection in synthetic aperture radar images: a novel approach. *Cogn Comput*. 2015;7(4):434–44.
- Garcia S, Luengo J, Saez JA, Lopez V, Herrera F. A survey of discretization techniques Taxonomy and empirical analysis in supervised learning. *IEEE Trans Knowl Data Eng*. 2013;25(4):734–50.
- Gong Y, Beck JE, Heffernan NT. Comparing knowledge tracing and performance factor analysis by using multiple model fitting procedures. In: Proceedings of the 10th International Conference on Intelligent Tutoring Systems, ITS 2010, Part I, Pittsburgh, PA, USA, June 14–18; 2010. p. 35–44.
- Goodfellow IJ, Bengio Y, Courville AC. *Deep Learning. Adaptive computation and machine learning*. MIT Press. 2016.
- Graves A, Mohamed A, Hinton GE. Speech recognition with deep recurrent neural networks. In: *IEEE ICASSP*; 2013. p. 6645–6649.
- Hastie T, Tibshirani R, Friedman J. *The elements of statistical learning: data mining, inference, and prediction*, 2nd ed. Berlin: Springer; 2009.
- Hinton GE, Osindero S, Teh YW. A fast learning algorithm for deep belief nets. *Neural Comput*. 2006;18(7):1527–54.
- Hochreiter S, Schmidhuber J. Long short-term memory. *Neural Comput*. 1997;9(8):1735–80.
- Hu D. How Khan Academy is using machine learning to assess student mastery. 2011.
- Hu J, Yang H, Lyu MR, King I, So AM-C. Online nonlinear AUC maximization for imbalanced data sets. *IEEE Trans Neural Netw Learning Syst*. 2017.
- Hu Z, Zhang Z, Yang H, Chen Q, Zuo D. A deep learning approach for predicting the quality of online health expert question-answering services. *J Biomed Inform*. 2017;71:241–53.
- Huang Y, González-brenes JP, Brusilovsky P. General features in knowledge tracing to model multiple subskills, temporal item response theory, and expert knowledge. In: EDM; 2014. p. 84–91.
- Huang Y, Guerra J, Brusilovsky P. A data-driven framework of modeling skill combinations for deeper knowledge tracing. In: EDM; 2016. p. 593–594.
- Khajah M, Lindsey RV, Mozer M. How deep is knowledge tracing? In: EDM; 2016.
- Khajah M, Wing R, Lindsey RV, Mozer M. Integrating latent-factor and knowledge-tracing models to predict individual differences in learning. In: EDM; 2014. p. 99–106.
- Koedinger KR, Cunningham K, Skogsholm A, Leber B. An open repository and analysis tools for fine-grained, longitudinal learner data. In: EDM; 2008. p. 157–166.
- Kotsiantis S, techniques D, Kanellopoulos. Discretization a recent survey. *GESTS International Transactions on Computer Science and Engineering*. 2006;32(1):47–58.
- Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. *Commun ACM*. 2017;60(6):84–90.
- Labutov I, Studer C. Calibrated self-assessment. In: EDM; 2016.
- LeCun Y, Bengio Y, Hinton GE. Deep learning. *Nature*. 2015;521(7553):436–44.
- Louppe G, Wehenkel L, Suter A, Geurts P. Understanding variable importances in forests of randomized trees. In: NIPS; 2013. p. 431–439.
- Mikolov T, Karafiát M, Burget L, Cernocký J, Khudanpur S. Recurrent neural network based language model. In: *INTER-SPEECH*; 2010. p. 1045–1048.
- Mingers J. An empirical comparison of pruning methods for decision tree induction. *Mach Learn*. 1989;4(2):227–43.
- Pardos ZA, Heffernan NT. Modeling individualization in a Bayesian networks implementation of knowledge tracing. In: Proceedings of the 18th International Conference on User Modeling, Adaptation, and Personalization, UMAP 2010, Big Island, HI, USA, June 20–24; 2010. p. 255–266.
- Pavlik JrPI, Cen H, Koedinger KR. Performance factors analysis—a new alternative to knowledge tracing. *Online Submission*. 2009.
- Piech C, Bassen J, Huang J, Ganguli S, Sahami M, Guibas LJ, Sohl-Dickstein J. Deep knowledge tracing. In: NIPS; 2015. p. 505–513.
- Quinlan JR. *C4.5: programs for machine learning*. Amsterdam: Elsevier; 2014.
- Schmidhuber J. Deep learning in neural networks: an overview. *Neural Netw*. 2015;61:85–117.
- Silver D, Huang A, Maddison CJ, Guez A, Sifre L, van den Driessche G, Schrittwieser J, Antonoglou I, Panneershelvam V, Lanctot M, Dieleman S, Grewe D, Nham J, Kalchbrenner N, Sutskever I, Lillicrap TP, Leach M, Kavukcuoglu K, Graepel T, Hassabis D. Mastering the game of go with deep neural networks and tree search. *Nature*. 2016;529(7587):484–9.
- Spratling MW. A hierarchical predictive coding model of object recognition in natural images. *Cogn Comput*. 2017;9(2):151–67.
- Sun R. Anatomy of the mind: a quick overview. *Cogn Comput*. 2017;9(1):1–4.
- Sweeney M, Lester J, Rangwala H, Johri A. Next-term student performance prediction: a recommender systems approach. In: EDM; 2016. p. 7.
- Tang J, Alelyani S, Liu H. Feature selection for classification A review. In: *Data classification: algorithms and applications*; 2014. p. 37.

46. Timofeev R. Classification and regression trees (CART) theory and applications. Berlin: PhD thesis, Humboldt University; 2004.
47. Vinyals O, Toshev A, Bengio S, Erhan D. Show and tell: lessons learned from the 2015 MSCOCO image captioning challenge. *IEEE Trans Pattern Anal Mach Intell.* 2017;39(4):652–63.
48. Wang L, Sy A, Liu L, Piech C. Deep knowledge tracing on programming exercises. In: *L@S; 2017.* p. 201–204.
49. Xiong X, Zhao S, Inwegen EV, Beck J. Going deeper with deep knowledge tracing. In: *EDM; 2016.* p. 545–550.
50. Xu C, Li P. Dynamics in four-neuron bidirectional associative memory networks with inertia and multiple delays. *Cogn Comput.* 2016;8(1):78–104.
51. Yang H, Ling G, Su Y, Lyu MR, King I. Boosting response aware model-based collaborative filtering. *IEEE Trans Knowl Data Eng.* 2015;27(8):2064–77.
52. Zhang J, Shi X, King I, Yeung D. Dynamic key-value memory networks for knowledge tracing. In: *WWW; 2017.* p. 765–774.
53. Zhang L, Xiong X, Zhao S, Botelho A, Heffernan NT. Incorporating rich features into deep knowledge tracing. In: *L@S; 2017.* p. 169–172.