Deep Knowledge Tracing on Skills with Small Datasets

Ange Tato and Roger Nkambou(✉)

Université du Québec à Montréal, Montréal, Canada
{nyamen_tato.angea_drienne,nkambou.roger}@uqam.ca

Abstract. Deep Knowledge Tracing (DKT), as well as other machine learning approaches, is biased toward data used during the training step. Thus, for problems where we have few amounts of data for training, the generalization power will be low, and the models will tend to work well on classes containing many samples and poorly on those with few. This situation is quite common in educational data where some skills are very difficult to master while others are very easy. As a result, there will be less data on students who correctly answered questions related to difficult skills, but also on those who provided incorrect answers to questions related to easy skills. In those cases, the DKT is unable to correctly predict the student's answers to questions associated with these skills. To improve DKT performance under these conditions, we have developed a two-fold approach. Firstly, the loss function is modified so that some skills are masked to force the model's attention on those that are difficult to generalize. Secondly, to cope with the limited amount of data on some skills, we proposed a hybrid architecture that integrates a priori (expert) knowledge with DKT through an attentional mechanism. The resulting model accurately tracks student Knowledge in the Logic-Muse Intelligent Tutoring System (ITS), compared to the traditional Bayesian Knowledge Tracing (BKT) and the original DKT.

Keywords: Deep Knowledge Tracing · Bayesian Knowledge Tracing · Knowledge Tracing

1 Introduction

Modeling students' knowledge is a fundamental step when building intelligent tutoring systems (ITS). Knowledge Tracing, a popular approach to modeling learner knowledge, aims at modeling how students' knowledge evolves during learning [7]. There exist many solutions to estimate that probability such as the Bayesian Knowledge Tracing (BKT) and the Deep Knowledge Tracing (DKT).

BKT [7] is a special case of the Hidden Markov Model where student knowledge is represented as a set of binary variables. Observations are also binary: a student gets a problem either right or wrong [29]. However, there is a certain probability (G, the Guess parameter) that the student will give a correct response. Correspondingly, a student who does know a skill generally will give a

S. Crossley and E. Popescu (Eds.): ITS 2022, LNCS 13284, pp. 123–135, 2022.
https://doi.org/10.1007/978-3-031-09680-8_12

correct response, but there is a certain probability (S, the Slip parameter) that the student will give an incorrect response. The standard BKT model is thus defined by four parameters: initial knowledge, learning rate (learning parameters), slip, and guess (mediating parameters). It has been successfully used in a variety of systems including computer programming [11], reading skills [4], logical reasoning [24] etc. Using a Bayesian network sometimes implies manually defining apriori probabilities and manually labeling student interactions with relevant concepts. Also, the binary response data used to model knowledge, observations and transitions impose a limit on the kinds of exercises that can be modeled. DKT has been proposed as a good alternative to overcome BKT limits.

Deep learning has been successfully applied in many domains including images recognition [9], Natural Language Processing [2,6] and more recently in education for modeling student knowledge. DKT [20] uses an LSTM (Long Short Term Memory) to predict student performance based on the pattern of their sequential responses. DKT observes knowledge at both the skill level, and the problem level, observing the correctness of each problem. At any time step, the input layer of the DKT is the student performance on a single problem of the skill that the student is currently working on. In other words, the skill and correctness of each item are used to predict the correctness of the next item, given that problem's skill [31]. Rather than constructing a separate model for each skill as BKT does, DKT models all skills jointly [12,20]. It has been shown that DKT can robustly predict whether or not a student will solve a particular problem correctly given the accuracy of historic solutions [26,31]. However many recent works have pointed out some issues with the DKT such as: the model only considers the knowledge components of the problems and correctness as input, neglecting the breadth of other features collected by computer-based learning platforms [31]. This problem was solved by incorporating more features into the input of the model and by incorporating an auto-encoder network layer to convert the input into a low dimensional feature vector. Other issues were pointed out by Chun-Kit et al. [28] which are (1) the model fails to reconstruct the observed input; As a result, even when a student performs well on a skill component, the prediction of that skill mastery level decreases instead, and vice versa; (2) the predicted performance for skills across time-steps is not consistent. As a solution, they augmented the loss function with regularization terms that correspond to 'reconstruction' and 'waviness'. This solution is similar to our first contribution.

Skills with limited data denotes skills that are very difficult to master (there are few data on students that have mastered these skills) and skills that are very easy to master (there are few data on students that have not mastered these skills). Like other machine learning techniques, the DKT is biased towards the data seen during the training phase due to its data-driven approach. Therefore, the generalization performance of the model depends on the training data. For problems (or skills) that are difficult to master, which means a rare occurrence of correct answers, the DKT is unable to accurately predict the student performance on questions associated with those skills. In the same vein, for skills

that are easy to master, which means a rare occurrence of incorrect answers, the DKT also fails to accurately predict the student performance on questions associated with those skills. In machine learning domain, this problem is known as the class imbalance problem [10] where there are fewer occurrences of data for a certain class which results in sub-optimal performance.

In the educational field, a priori expert knowledge is usually available and generally used to build systems such as Intelligent Tutoring Systems (ITS). Expert knowledge can be available through books or previously built models (such as rules-based models). We believe that this a priori expert knowledge, sometimes acquired over decades of intense research, cannot be dismissed and ignored. Sometimes, a priori expert knowledge can be available but is not always sufficiently accurate. Nevertheless, even inaccurate models can provide useful information that should not be dismissed [30]. In general, employing a fully data-driven approach to train deep neural networks requires the acquisition of a huge amount of data, which might not always be practical or realistic due to economic reasons or the complexity of the process it entails. We show that combining a priori expert knowledge and data-driven methods using the attentional mechanism constitutes a suitable approach towards the design of hybrid deep learning architecture.

In this paper, we put forth an approach that uses attentional mechanism [27] and capitalizes on the availability of expert knowledge (through a Bayesian Network built by experts) to overcome the problem of having few data when training machine deep learning models. We also propose to leverage the problem of skills with limited data by using a custom loss function for the DKT, where we mask skills with many samples and give weight to skills with few samples. The main contributions of this work can be summarized as follows: (1) An extension that improves the original DKT in the prediction of skills with limited data; (2) The incorporation of a priori knowledge (when available) using attention mechanism in a deep learning architecture. We applied the proposed solution to the prediction of the logical reasoning performance of students.

2 Brief Review of the Deep Knowledge Tracing

DKT takes as input sequences of exercise-performance (e_t, p_t) pairs presented one trial at a time. The model then predicts the knowledge state, based on the current hidden state. The hidden layer of the LSTM represents the latent encoding of knowledge state, based on the current input and previous latent encoding of knowledge state. It represents the latent knowledge state of a student, resulted from his past learning trajectory.

To train the model, the exercise-performance needs to be converted into a sequence of fixed length input vectors x_t. x_t is a one-hot encoding of (e_t, p_t) that represents the combination of which exercise (skill involved) was answered and the real answer given by the student. For student s with a sequence of exercice-performance of length T, the DKT model maps the inputs $(x_1, x_2, ..., x_T)$ to the output y_t which is a vector of length equals to the number of skills. Each entry

of y_t represents the predicted probability that the student will correctly answer exercises from that particular skill. The training objective is the negative log-likelihood of the observed sequence of student responses under the model. The loss function is as follows [20]:

$$L = \sum_t \ell(y_t^\top \delta(e_{t+1}), p_{t+1}) \tag{1}$$

$\delta(e_{t+1})$ is the one-hot encoding of which exercise is answered at time $t+1$. ℓ is the binary cross entropy.

3 Penalization of Loss Function

The performance prediction on skills with little data can be compared to unbalanced problems in machine learning. There are multiple strategies to deal with class imbalance such as resampling the data by under-sampling the majority class or oversampling the minority class [19]. However, over-sampling can easily introduce undesirable noise with overfitting risks; on the other hand, under-sampling is often preferred but may remove valuable information, which we can't afford because of the few amounts of data. Another well-known strategy is cost-sensitive learning, which assigns higher misclassification costs to the minority class than to the majority class [23]. Our proposed solution falls into the category of cost-sensitive learning, where we assigned a higher cost for skills with few samples. In other words, during the training, we force the model to pay more attention to floor/ceiling skills.

The main idea is to treat the loss as a weighted average where the weights are specified by parameters λ_i with $i \in [0,n]$ where n is the number of parts to add to the original loss. We added a regularization term in the loss function which corresponds to the application of a mask to the original loss to ignore skills with many samples. The result of the mask has 2 parts: the very difficult skills and the very easy skills. Each part of the mask is multiplied by regularization parameters (or weights) λ_1 and λ_2 respectively. These factors are the penalties applied to the model. Penalizing the DKT model imposes an additional cost when making prediction mistakes on the minority class during training. These penalties bias the model to pay more attention to the minority class (correct answers on skills difficult to master and incorrect answers on skills easy to master). Now we will explain how we compute the new loss.

If we were to evaluate a DKT model for the prediction of a knowledge state on only one skill k, the loss function would be written as follows:

$$L_k = \sum_t \ell(y_{k,t}, p_{k,t+1}) \tag{2}$$

where $y_{k,t}$ is a vector of length equals to the number of skills with 0 on all the entries except for the entry k. It represents the predicted probability at time t that a student would correctly answer a problem related to that specific skill

(k) at time $t+1$. $p_{k,t+1}$ denotes the real answer (performance) given at time $t+1$ by a student on a problem related to the skill k. For the model to be able to make good predictions on skills with few data, we first extracted those skills using statistics (skills with a very small number of correct answers and skills with a very high number of correct answers). We could have also used the DKT itself (by running the DKT and then identifying skills where the precision and recall are low for each of the correct/incorrect classes). We then apply a mask (this is easily done with any programming language) on the loss function whose purpose is to hide skills with many samples to only keep what we want the model to focus on. We then run the model with a new loss function:

$$L = \sum_t \ell(y_t^T \delta(e_{t+1}), p_{t+1}) + \lambda_1 \sum_t \sum_{kF} \ell(y_{kF,t}, p_{kF,t+1,1}) + \lambda_2 \sum_t \sum_{kC} \ell(y_{kC,t}, p_{kC,t+1,0}) \tag{3}$$

Parameters λ_1 and $\lambda_2 >= 0$ are weights that we apply to the mask and kF, kC denote respectively all skills that are difficult to master (floor skills) and all skills that are easy to master (ceiling skills). The digit 1 in $p_{kF,t+1,1}$ denotes correct answers and digit 0 in $p_{kC,t+1,0}$ denotes incorrect answers . If $\lambda = 0$, the model becomes the original DKT. The more the value λ is high, the more the model will be biased towards skills with few samples (floor/ceiling skills). The choice of the value of λ is thus important. $y_{k,t}$ and $p_{k,t+1}$ are the vectors y_t and p_{t+1} respectively, where we only keep values related to the skill k. Thus $p_{kF,t,1}$ refers to a vector of length equals to the number of skills with 0 on all the entries except for entries kF and where the real answers given are correct at time $t+1$. $\delta(e_{t+1})$ is the one-hot encoding of which exercise is answered at time $t+1$. The new loss provides another way to balance the data.

4 Combining a Bayesian Network with the DKT

BN is a graphical model used to model process under uncertainty by representing relationships between variables in terms of a probability distribution [21]. BN allows inferring the probability of mastering a skill from a specific response pattern [18]. The structure and the parameter or probability distributions are provided by experts or learned using algorithms such as Expectation-Maximization. In this work, we only consider BNs that are built by experts. BNs have been successfully used to model knowledge state of learners [15,16,24] or learner affect [22]. There are many contexts where a lot of data or expert knowledge (e.g. medicine) are available. How can the DKT benefit from that? We want to take advantage of expert knowledge when available. We also suppose that the model accuracy could increase as expert knowledge is not biased towards rare data, which can be very helpful in the case of few amounts of data.

The inner architecture of a neural network makes it difficult to incorporate domain knowledge into the learning process [13]. Our solution is to force the model to pay attention to what the expert knowledge says about the current

input x. The goal is not to incorporate how the expert knowledge is processed, but rather its final prediction about the input. Since attention [27] is a memory-access mechanism, it fits well in this context where we want the model to have access to the expert knowledge during learning [1], as a memory. Thus, the model will pay attention to what the expert knowledge says before taking any decision. Attention-based recurrent networks have been successfully applied to a wide variety of tasks, such as machine translation [3], handwriting synthesis [8], speech recognition [5], etc. By integrating the expert knowledge (here a BN) in the DKT using the attention mechanism, the model iteratively processes the apriori knowledge by selecting relevant content at every step. In the attention mechanism presented by Luong et al. [14] (specifically the global attentional model), the attentional vector is computed from the target hidden state h_t and the input hidden state. Instead of the input hidden state, we will have the data coming from expert knowledge which will be used to compute the context vector C_t (that we will call expert-side context vector) (see Fig 2 in [14]). Thus, given the hidden state y_t (the prediction) of the DKT, and the expert-side context vector c_t^e, we employ a concatenation layer to combine the information from both vectors to produce the attentional hidden state a_t as follows:

$$a_t = tanh(W_c[c_t^e; y_t]) \tag{4}$$

The attentional vector a_t along with the expert prediction e and the y_t are then fed through a Dense layer to produce the predictive knowledge state y'_t. Now, the expert-side context is computed as follows:

$$\begin{aligned} score(e_k, y_t) &= e_k \cdot y_t \cdot W_a + b \\ \alpha_{t,k} &= \frac{exp^{score(e_k, y_t)}}{\sum_{j=1}^{s} exp^{score(e_j, y_t)}} \\ c_t^e &= \sum_k \alpha_{t,k} \cdot e \end{aligned} \tag{5}$$

where $1 <= k <= s$, e is the current knowledge state predicted by the expert, y_t is the current knowledge state predicted by the DKT and s is the number of skills. The parameter e represents a vector of length equals to the number of skills where each entry represents the predicted probability that the student will answer correctly to exercises from that particular skill, given by the BN. e_k is of size 1 and W_a, W_c, W_s, y_t, e and a_t are of size s the number of skills. Figure 1 shows in detail this global process.

5 Experiments

Our goal is to create an accurate learner model in an ITS called Logic-Muse. The current learner model implemented in Logic-Muse uses a BN [hidden] built from expert knowledge.

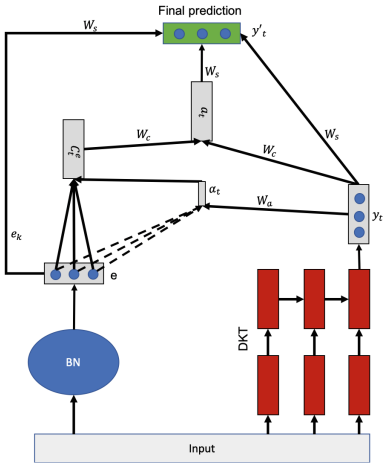

Fig. 1. Global attentional hybrid model—at each time step t, the model infers an alignment weight vector $\alpha_{t,k}$ based on the current predicted knowledge y_t and all entries of the expert knowledge vector. c_t^e is then computed as the weighted average, according to $\alpha_{t,k}$, over each of the entries of the expert knowledge vector.

5.1 Logic Muse

Logic-Muse is a web-based Intelligent Tutoring System that helps learners improve logical reasoning skills. Logic-Muse includes a learning environment that uses various meta-structures to provide reasoning activities on various contents [hidden]. The expert model implements logical reasoning skills and knowledge as well as related reasoning mechanisms (syntactic and semantic rules of the given logical system). The model of (valid and invalid) inference rules are encoded as production rules, and the semantic memory of the target logic is encoded in a formal OWL ontology and connected to the inference rules. The first version of Logic-Muse focuses on propositional logic. Logic-Muse learner model goal is to represent, update and predict the learner's state of knowledge based on her/his interaction with the system. It has multiple aspects including the cognitive part that essentially represents the learner's knowledge state (mastery of the reasoning skills in each of the six reasoning situations that have been identified thanks to the experts). The cognitive state is generated from the learner's behavior during his interactions with the system, that is, it is inferred by the system from the information available. It is supported by a Bayesian network [hidden] based on domain knowledge, where influence relationships between nodes (reasoning skills) as well as prior probabilities are provided by the experts. Some nodes are directly connected to the reasoning activities such as exercises. The skills involved in the BN are those put forward by the mental models' theory to reason in conformity to the logical rules. There are 16 skills directly observable (linked to the exercises) and 12 latent skills. There is a total of 48 exercises.

5.2 Dataset

294 participants participated in this study. They all completed the 48 logical reasoning exercises. In our dataset, each line of data represents each participant (a total of 294 data and a sequence length of 48). The amount of data is very few to train a deep learning model. However, combined with expert knowledge, we will see a substantial difference in the results. The exercises were encoded using skills that are directly observable, which means that the questions related to the same skill are encoded with the same Id (1∼16). The skills with few data are determined by a comparison of the average of correct answers obtained for each skill. In Table 1, we averaged all the answers on each skill. The skills difficult to master (floor) are those with the lowest average value and the skills easy to master (ceiling) are those with the highest average. The second and the last part of our new loss function involve those skills. Since the LSTM only accept a fixed length of vectors as the input, we used one-hot encoding to convert student performance into a fixed length of vectors whose all elements are 0 except for a single 1. The single 1 in the vector indicates two things: which skill was answered and if the skill was answered correctly.

Table 1. Distribution of responses over skills—Skills difficult to master (Average < 0.4) and skills easy to master (Average > 0.9) are in **bold**.

Skills	N	Average	Standard dev
MppFd	294	**0,9456**	0,16078
MppMd	294	0,898	0,23726
MppCcf	294	**0,907**	0,2394
MppA	294	**0,9615**	0,16066
MttFd	294	0,8435	0,26646
MttMd	294	0,7925	0,29985
MttCcf	294	0,7494	0,33326
MttA	294	0,8401	0,28974
AcMa	294	0,424	0,38072
AcFa	294	**0,3039**	0,3652
AcCcf	294	**0,3345**	0,40801
AcA	294	**0,2823**	0,41038
DaMa	294	0,407	0,37389
DaFa	294	**0,3027**	0,35081
DaCcf	294	**0,381**	0,40662
DaA	294	**0,305**	0,42077

5.3 Results

To assess our proposed solutions, we ran 3 models: the DKT, the DKT where we applied a mask to the loss function (DKTm), and the DKTm with apriori knowledge (DKTm+BN). We used 20% of the data for testing and 15% for validation. The BN alone gave 65% of global accuracy. The result is evaluated using the $F1score$ metric on each skill (treated as 2 classes - correct and incorrect answers) being predicted and the overall accuracy. The models were evaluated in 20 different experiments and the final results were averaged. In all our experiments, we set λ_1 and $\lambda_2 = 0.10$. Our implementation of the DKTm+BN model in Tensorflow using Keras backend was inspired by the implementation[1] done by Khajah et al. [12]. Our code is also available on GitHub[2] for further research.

The results (for skills that are difficult to master) are shown in Figs. 2 and 3. As expected, the new DKTm (Accuracy = 0.8 outperforms the original DKT (Accuracy = 0.74) on all the skills being predicted. Furthermore, the DKTm enhanced with BN (Accuracy = 0.82 outperforms all other models on predicting skills with little data (good answers on skills difficult to master). For skills that are easy to master (e.g. **MPP**), all the models always predict that students will give correct answers (F1score of incorrect answers is 0 for DKT and almost 0 for the other models) even after applying the weighted loss. This is because, on the 294 data, we have for example only 6 incorrect answers for the **MPP_FFD** skill. We tested the models with high values for λ_2 and we got values of f1score equal to around 0.6 for correct answers and around 0.7 for incorrect answers. This result can be satisfying in other contexts but in the context of logical reasoning where it is established that the **MPP** is a skill that is always well mastered, 0.6 as an f1score for correctly predicting a correct answer is not acceptable. That is why we kept $\lambda_2 = 0.10$. However, the solution stays valid for data where the ratio r = number of correct answers/ number of questions answered or r = number of incorrect answers/ number of questions answered on a skill is not too small (as in this case) and is less than 0.5. For skills that are difficult to master (Figs. 2 and 3), there is a huge difference between the DKT and the other models. The DKT is unable to track correct answers on skills difficult to master (see Fig. 3). This behavior cannot be accepted since the knowledge tracing of students who perform well on those skills will fail. Thus it is important to make sure that the final model is accurate for all the skills. For the prediction of wrong answers on skills difficult to master (see Fig. 3), we can notice that the DKTm and the DKTm+BN still behave better than DKT which means that the penalty added to the loss function does not affect the predictive capacity of the original DKT.

We noticed that predictions are not sometimes consistent with the reality as other works have also highlighted [28]. The model fails to reconstruct the observed input. As a result, even when a student performs well on a skill, the prediction of that skill's mastery level decreases instead, and vice versa. Also, the predicted performance across time steps is not consistent. When a student gives

[1] https://github.com/mmkhajah/dkt.

[2] https://github.com/angetato/Deep-Knowledge-Tracing-On-Skills-With-Limited-Data.

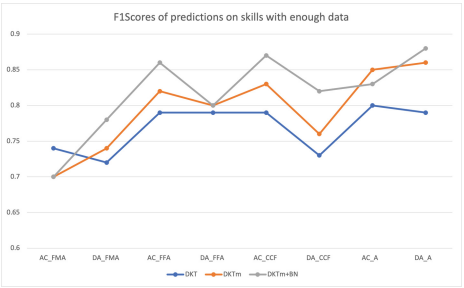

Fig. 2. The DKT, the DKTm and the DKTm+BN on skills that are difficult to master—We repeated the experiments 20 times. For each skill, we computed the value of the f1score for the prediction of incorrect answers (enough data).

correct answers to a skill k, the DKT does not sometimes update the current state of knowledge on that skill (the skill stays low or is updated very slowly). The problem can be addressed by adding regularization terms to the loss function of the original DKT as suggested by [28]. It can also be partially solved when adding the a priori knowledge as we noticed during our experiments. However, we stay confident in the fact that if the a priori knowledge is more accurate (which is not our case as the accuracy of the BN is 0.65) we will get better results.

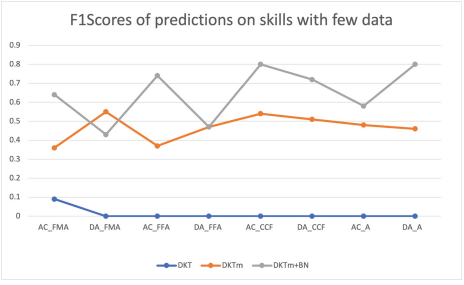

Fig. 3. Prediction score of correct answers on skills that are difficult to master (limited data) for the three models: DKT, DKTm and DKTm+BN.

The idea of using the attention mechanism to incorporate expert knowledge into NN is novel and can be used in other domains such as text classification or in medicine where there is a lot of expert knowledge available. For example, we could think of a classifier using a neural network combined with a rule-based system playing the role of expert knowledge.

6 Conclusion

In this paper, we proposed two simple, effective, and intuitive techniques to improve the DKT on the prediction of floor and ceiling skills which are skills that are very difficult and easy to master respectively. The first technique consists of applying a penalty to the loss function, for making incorrect predictions on skills with few samples. The second solution aims at incorporating a BN (expert knowledge) in the DKT using the attention mechanism. At the same time, we introduced a new way of using the attention mechanism, to allow neural networks to take into account expert knowledge (when available) in their training and decision process. We tested the solution on a dataset that is unbalanced. The results showed that the DKT is unable to accurately track skills with limited data, compare to the DKTm and the DKTm+BN.

During the experiments, we noticed that, for skills that are very easy to master, all the 3 models were unable to track incorrect answers, since the ratio r = incorrect answers/total of questions answered was very low. Even with a penalty, we were not able to significantly improve the DKT model on the skills involved. However, with the combination of the BN, the results were noticeable. In this paper, we have set the regularization parameters λ_1 and λ_2 with a fixed value but for future work, we will do a grid search to find the best values.

We are aware that the lack of data might be a bias to our results since deep learning architectures perform better on larger datasets. However, the floor/ceiling skills problem can still occur even with a large dataset. Thus, we believe that the solutions we have proposed can work perfectly on larger datasets. We will do further experiments on the integration of expert knowledge into neural network architectures. We also plan to test our techniques on larger and or public datasets such as ASSISTments dataset.

References

1. Ange, T., Roger, N., Aude, D.: Hybrid deep neural networks to predict socio-moral reasoning skills. In: Proceedings of the 12th International Conference on Educational Data Mining, pp. 623–626 (2019)
2. Ange, T., Roger, N., Aude, D., Claude, F.: Semi-supervised multimodal deep learning model for polarity detection in arguments. In: 2018 International Joint Conference on Neural Networks (IJCNN), pp. 1–8. IEEE (2018)
3. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 (2014)
4. Beck, J.E., Chang, K.: Identifiability: a fundamental problem of student modeling. In: Conati, C., McCoy, K., Paliouras, G. (eds.) UM 2007. LNCS (LNAI), vol. 4511, pp. 137–146. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-73078-1_17
5. Chorowski, J.K., Bahdanau, D., Serdyuk, D., Cho, K., Bengio, Y.: Attention-based models for speech recognition. In: Advances in Neural Information Processing Systems, pp. 577–585 (2015)
6. Collobert, R., Weston, J.: A unified architecture for natural language processing: deep neural networks with multitask learning. In: Proceedings of the 25th International Conference on Machine learning, pp. 160–167. ACM (2008)

7. Corbett, A.T., Anderson, J.R.: Knowledge tracing: modeling the acquisition of procedural knowledge. User Model. User-Adap. Inter. **4**(4), 253–278 (1994)
8. Graves, A.: Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850 (2013)
9. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
10. Huang, C., Li, Y., Change Loy, C., Tang, X.: Learning deep representation for imbalanced classification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5375–5384 (2016)
11. Kasurinen, J., Nikula, U.: Estimating programming knowledge with bayesian knowledge tracing. In: ACM SIGCSE Bulletin, vol. 41, pp. 313–317. ACM (2009)
12. Khajah, M., Lindsey, R.V., Mozer, M.C.: How deep is knowledge tracing? arXiv preprint arXiv:1604.02416 (2016)
13. Lu, H., Setiono, R., Liu, H.: Effective data mining using neural networks. IEEE Trans. Knowl. Data Eng. **8**(6), 957–961 (1996)
14. Luong, M.T., Pham, H., Manning, C.D.: Effective approaches to attention-based neural machine translation. arXiv preprint arXiv:1508.04025 (2015)
15. Martin, J., VanLehn, K.: Student assessment using bayesian nets. Int. J. Hum Comput Stud. **42**(6), 575–591 (1995)
16. Nguyen, L., Do, P.: Combination of bayesian network and overlay model in user modeling. In: Allen, G., Nabrzyski, J., Seidel, E., van Albada, G.D., Dongarra, J., Sloot, P.M.A. (eds.) ICCS 2009. LNCS, vol. 5545, pp. 5–14. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01973-9_2
17. Nkambou, R., Brisson, J., Kenfack, C., Robert, S., Kissok, P., Tato, A.: Towards an intelligent tutoring system for logical reasoning in multiple contexts. In: Conole, G., Klobučar, T., Rensing, C., Konert, J., Lavoué, É. (eds.) EC-TEL 2015. LNCS, vol. 9307, pp. 460–466. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24258-3_40
18. Nkambou, R., Mizoguchi, R., Bourdeau, J.: Advances in Intelligent Tutoring Systems, vol. 308. Springer, Berlin (2010). https://doi.org/10.1007/978-3-642-14363-2
19. Oquab, M., Bottou, L., Laptev, I., Sivic, J.: Learning and transferring mid-level image representations using convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1717–1724 (2014)
20. Piech, C., et al.: Deep knowledge tracing. In: Advances in Neural Information Processing Systems, pp. 505–513 (2015)
21. Russell, S.J., Norvig, P.: Artificial Intelligence: a Modern Approach. Pearson Education Limited, Malaysia (2016)
22. Sabourin, J., Mott, B., Lester, J.C.: Modeling learner affect with theoretically grounded dynamic bayesian networks. In: D'Mello, S., Graesser, A., Schuller, B., Martin, J.-C. (eds.) ACII 2011. LNCS, vol. 6974, pp. 286–295. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-24600-5_32
23. Shen, W., Wang, X., Wang, Y., Bai, X., Zhang, Z.: Deepcontour: a deep convolutional feature learned by positive-sharing loss for contour detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3982–3991 (2015)

24. Tato, A., Nkambou, R., Brisson, J., Kenfack, C., Robert, S., Kissok, P.: A bayesian network for the cognitive diagnosis of deductive reasoning. In: Verbert, K., Sharples, M., Klobučar, T. (eds.) EC-TEL 2016. LNCS, vol. 9891, pp. 627–631. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45153-4_78
25. Tato, A., Nkambou, R., Brisson, J., Robert, S.: Predicting learner's deductive reasoning skills using a bayesian network. In: André, E., Baker, R., Hu, X., Rodrigo, M.M.T., du Boulay, B. (eds.) AIED 2017. LNCS (LNAI), vol. 10331, pp. 381–392. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-61425-0_32
26. Wang, L., Sy, A., Liu, L., Piech, C.: Deep knowledge tracing on programming exercises. In: Proceedings of the Fourth (2017) ACM Conference on Learning@ Scale, pp. 201–204. ACM (2017)
27. Xu, K., et al.: Show, attend and tell: Neural image caption generation with visual attention. In: International Conference on Machine Learning, pp. 2048–2057 (2015)
28. Yeung, C.K., Yeung, D.Y.: Addressing two problems in deep knowledge tracing via prediction-consistent regularization. arXiv preprint arXiv:1806.02180 (2018)
29. Yudelson, M.V., Koedinger, K.R., Gordon, G.J.: Individualized bayesian knowledge tracing models. In: Lane, H.C., Yacef, K., Mostow, J., Pavlik, P. (eds.) AIED 2013. LNCS (LNAI), vol. 7926, pp. 171–180. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39112-5_18
30. Zappone, A., Di Renzo, M., Debbah, M., Lam, T.T., Qian, X.: Model-aided wireless artificial intelligence: Embedding expert knowledge in deep neural networks towards wireless systems optimization. arXiv preprint arXiv:1808.01672 (2018)
31. Zhang, L., Xiong, X., Zhao, S., Botelho, A., Heffernan, N.T.: Incorporating rich features into deep knowledge tracing. In: Proceedings of the Fourth (2017) ACM Conference on Learning@ Scale, pp. 169–172. ACM (2017)