Deep Knowledge Tracing and Dynamic Student Classification for Knowledge Tracing

Sein Minn[1], Yi Yu[2], Michel C. Desmarais[1], Feida Zhu [3], Jill-Jênn Vie[4]

[1]Department of Computer Engineering, Polytechnique Montreal, Canada
[2]Digital Content and Media Sciences Research Division, National Institute of Informatics, Japan
[3]School of Information Systems, Singapore Management University, Singapore
[4]RIKEN Center of Advanced Intelligence Project, Japan

Abstract—In Intelligent Tutoring System (ITS), tracing the student's knowledge state during learning has been studied for several decades in order to provide more supportive learning instructions. In this paper, we propose a novel model for knowledge tracing that i) captures students' learning ability and dynamically assigns students into distinct groups with similar ability at regular time intervals, and ii) combines this information with a Recurrent Neural Network architecture known as Deep Knowledge Tracing. Experimental results confirm that the proposed model is significantly better at predicting student performance than well known state-of-the-art techniques for student modelling.

Index Terms—Student model, Deep knowledge tracing, K-means clustering, RNNs, LSTMs

[1]

I. INTRODUCTION

ITS is an active field of research that aims to provide personalized instructions to students. Early work dates back to the late 1970s. A wide array of Artificial Intelligence and Knowledge Representation techniques have been explored, of which we can mention rule-based and Bayesian representation of student knowledge and misconceptions, skills modeling with logistic regression in Item Response Theory, case-based reasoning, and, more recently reinforcement learning and deep learning [1], [2]. One can even argue that most of the main techniques found in Artificial Intelligence and Data Mining have found their way into the field of ITS, and in particular for the problem of knowledge tracing, which aims to model the student's state of mastery of conceptual or procedural knowledge from observed performance on tasks [3].

In this paper we propose a novel model for knowledge tracing, Deep Knowledge Tracing with Dynamic Student Classification (DKT-DSC). At each time interval, the model first assigns a student into a distinct group of students that share similar learning ability. This information is then fed to a Recurrent Neural Network (RNN), known as the DKT architecture [4] for predicting student's performance from data. We can consider the student classification as a long-term memory of the student's ability as input to the RNN improves knowledge tracing with DKT, which is among the state-of-the-art approach to knowledge tracing.

---
[1]This work is available at https://github.com/simon-tan/DKT-DSC.git

The rest of this paper is organized as follows. Section II reviews related work on student modelling techniques. Section III presents the proposed DKT-DSC model. Section IV describes the datasets used in our experiments. Experimental results are shown in Section V and finally Section VI concludes this paper and discusses future avenues of research.

II. RELATED WORK

We review here four of the best known state-of-the-art student modelling methods for estimating student's performance, either for their predominance in psychometrics (IRT) or Educational Data Mining (BKT), or because they are best performers (PFA, DKT). See [5] for a general review.

A. Item Response Theory (IRT)

IRT assumes the student knowledge state is static and represented by her proficiency when completing an assessment during an exam [6], [7], [8], [9]. IRT models a single skill and assumes the test items are *unidimensional*. It assigns student i with a static proficiency θ_i. Each item j has its own difficulty β_j. The main idea of IRT is estimating a probability that student i answers item j correctly by using student's ability and item's difficulty. The widely used one-parameter version of IRT, known as the Rasch model, is

$$p_j(\theta_i) = \frac{1}{1 + e^{-(\theta_i - \beta_j)}}. \qquad (1)$$

Recently, Wilson [6] proposed an IRT model that outperforms state-of-the-art knowledge tracing models. In which, maximum a posteriori (MAP) estimates of θ_i and β_j are computed using the Newton-Raphson method.

B. Bayesian Knowledge Tracing (BKT)

BKT was introduced for knowledge tracing within a learning environment for which the assumption on static knowledge states is dropped [3], [10]. It also assumes a single skill is tested per item, but this assumption is relaxed in later work on BKT. Standard BKT estimate of student's knowledge about a skill is continually updated with four probabilities: [$P(L_0)$ initial probability of mastery, $P(T)$ transitioning from non-mastery to mastery, $P(G)$ guessing and $P(S)$ slipping], once the student gives her response at each time:

$$P(L_n|Correct) = \frac{P(L_{n-1})(1 - P(S))}{P(L_{n-1})(1 - P(S)) + (1 - P(L_{n-1}))P(G)} \quad (2)$$

$$P(L_n|Incorrect) = \frac{P(L_{n-1})P(S)}{P(L_{n-1})P(S) + (1 - P(L_{n-1}))(1 - P(G))} \quad (3)$$

$$P(L_n) = P(L_{n-1}|Action) + (1 - P(L_{n-1}|Action))P(T) \quad (4)$$

There have been various extensions of BKT in the last decades [11], [12].

C. Performance Factor Analysis (PFA)

PFA, which was proposed as an alternative to BKT, also relaxes the static knowledge assumption and models multiple skills simultaneously [13] with its basic structure. It defines the probability of success to an item j by student i as:

$$P(m_{i,j}) = 1/(1 + e^{-\ell_{i,j}}) \quad (5)$$

$$\ell_{i,j} = \beta_j + \sum_{k \in KCs} (\gamma_k s_{ik} + \rho_k f_{ik}), \quad (6)$$

where β_k is the bias for the skill k, and γ_k and ρ_k represent the learning gain per success and failure attempt to skill k, respectively. S_{ik} is the number of successful attempts and f_{ik} is the number of failure attempts made by student i on skill k [13].

D. Deep Knowledge Tracing (DKT)

DKT was introduced in [4]. It uses a Long Short-Term Memory (LSTM)[14] to represent the latent knowledge space of students dynamically. The increase in student's knowledge through an assignment can be inferred by utilizing the history of student's previous performance. DKT uses large numbers of artificial neurons for representing latent knowledge state along with a temporal dynamic structure and allows a model to learn the latent knowledge state from data. It is defined by the following equations:

$$h_t = \tanh(W_{hx}x_t + W_{hh}h_{t-1} + b_h), \quad (7)$$

$$y_t = \sigma(W_{yh}h_t + b_y). \quad (8)$$

In DKT, both tanh and the sigmoid function are applied element wise and parameterized by an input weight matrix W_{hx}, recurrent weight matrix W_{hh}, initial state h_0, and readout weight matrix W_{yh}. Biases for latent and readout units are represented by b_h and b_y.

III. DEEP KNOWLEDGE TRACING WITH DYNAMIC STUDENT CLASSIFICATION

Human learning is a process that involves practice: we become proficient through practice. However, learning is also affected by the individual's ability to learn, or to become proficient with more or less practice. We refer to the ability to become proficient with little practice as the learning ability. Based on that notion, we proposed a model Deep Knowledge Tracing with Dynamic Student Classification (DKT-DSC), that

assesses a student's learning ability and assign her into a distinct group of students with similar ability, and then the model invokes an RNN to trace her knowledge in each distinct group at different time intervals. It can trace the performance of students based on their learning ability, reassessed regularly over time.

A. Dynamic assessment of student's learning ability and grouping

Dividing students into distinct groups with similar learning ability, according to their previous performance on various contents in a learning system, has been explored in several research works in the field of education [15], [16] for providing more adaptive instructions to each group of students with similar ability. Dynamic assessment of student learning ability at each time interval is performed by clustering based on the assessment of their previous performance history before the start of next time interval.

1) Time interval: Time interval is a segment containing a number of student's attempts to answer questions in the system. In this perspective, a tick of time is a single first attempt to a question or exercise.

2) Segmenting students' attempt sequence: segmentation of each student response sequence into multiple time intervals serves two purposes: 1) To reduce computational burden and memory space allocation for learning throughout a long sequence. 2) To re-assess a student' learning ability after each time interval and assign her into a group which she belongs to for the next time interval dynamically.

Fig. 1. Segmentation of a student's attempt sequence.

Fig. 1 illustrates an example of dividing a 24-attempt response sequence of a student into 5 segments (time intervals) where a segment represents a time interval in which that student answered 6 problems in the system. When the student stopped interacting with system, it is represented with -1 in the last time interval. The number of attempts made by each student varies based on the number of questions they answered during the interaction with system.

3) Long-term skills encoding for clustering: Student are grouped according to their learning ability profile: the skills or knowledge they acquired. Data for assessing student's learning ability is available from previous attempts on test items or exercises corresponding to a specific skill.

The learning ability profile is encoded as a vector of length the number of skills, and updated after each time interval by using all previous attempts on each skill. The differences between success and failure ratios on each skill of student's

previous attempts are transformed into a data vector for clustering student i at time interval z as follows:

$$Correct(x_j)_{1:z} = \sum_{t=1}^{z} \frac{(x_{jt} == 1)}{|N_{jt}|}, \qquad (9)$$

$$Incorrect(x_j)_{1:z} = \sum_{t=1}^{z} \frac{(x_{jt} == 0)}{|N_{jt}|}, \qquad (10)$$

$$R(x_j)_{1:z} = Correct(x_j)_{1:z} - Incorrect(x_j)_{1:z}, \qquad (11)$$

$$d_{1:z}^{i} = (R(x_1)_{1:z}, R(x_2)_{1:z}, ..., R(x_n)_{1:z}), \qquad (12)$$

in which $Correct(x_j)_{1:z}$ and $Incorrect(x_j)_{1:z}$ represent the ratios of skill x_j being correctly answered or incorrectly, by student i on n number of skills $(x_1, x_2, .., x_n)$ from time interval 1 to current time interval z. $|N_{jt}|$ is the total number of practices of skill x_j up to time interval t. $R(x_j)_{1:z}$ represents the difference between how much student i performs on skill j, correctly or incorrectly, for time interval 1 to z and $d_{1:z}^{i} \in D$ represents a vector containing the learning ability profile of student i on each skill from time interval 1 until z. Each student may have a different number of total time intervals in the lifetime of their interactions with the system (see Fig. 3).

4) K-means Clustering: Assigning students into a group with similar ability at each time interval is performed by k-means clustering on data D [17], [18]. At the time of the clustering training phase, we find the centroids for each student group without considering the time interval index. Once it has been computed, the centroid of each group will not change any more during the whole clustering process. After that, we assign students (in both training and testing data) into distinct groups at each time interval (see Fig 2).
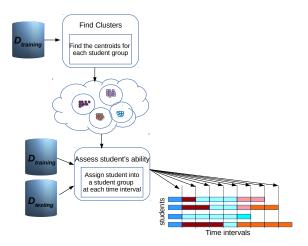

Fig. 2. Clustering students at each time interval.

When we find the group which student i belongs to at time interval z, we use the learning ability profile data points $d_{1:z-1}^{i}$ because we are not supposed to know the current attempts of student i at time interval z. After learning the centroids of all

K clusters, each student at each time interval Seg_z is assigned into the nearest cluster C_c by the following equation:

$$Cluster(Stu_i, Seg_z) = \arg\min_C \sum_{c=1}^{K} \sum_{d_{1:z-1}^{i} \in C_c} ||d_{1:z-1}^{i} - \mu_c||^2 \qquad (13)$$

where μ_c is the mean of points in a cluster set C_c (a group of students), and ability profile data $d_{1:z-1}^{i}$ represents the previous performance data of student i from time interval 1 to $z-1$.
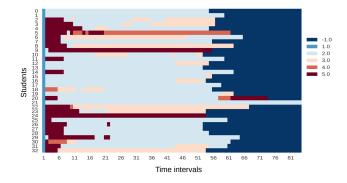

Fig. 3. Evolution of students' learning ability over each time interval (each time interval contains 20 attempts) throughout their interactions.

Figure 3 illustrates the data of 33 students' learning abilities based on their previous performance and the evolution over time intervals. Dark blue (-1) means students do not have any attempt by the time when they quit the system. Group 1 is for the first time interval of every student and the rest of the groups $(2, .., 5)$ are assigned by the k-means clustering method at each time interval z by using previous performance data $d_{1:z-1}^{i}$.

B. Deep knowledge tracing

DKT-DSC incorporates student's learning ability to the DKT for better individualization of the system, by assigning a student into a group of students with similar ability dynamically. It relaxes the assumption that all students have the same ability and that students' ability is consistent over time. In fact, student's ability is evolving continuously and some students may learn faster than others.

In the standard DKT, x_t is a one-hot encoding vector of the student interaction tuple $x_t = \{s_t, a_t\}$ that represents the combination of the skills s_t practiced, and of a_t which indicates if the answer is correct. But DKT-DSC also requires $x_t = \{s_t, a_t\}$ additionally with c_t which is a group or cluster $Cluster(Stu_i, Seg_z)$ indicating Stu_i's ability at current time interval Seg_z . In the hidden layers, the last node of each time interval is served as first node h_0 for next time interval when we segment the response sequence into multiple time intervals. The output y_t is a vector of same length as the number of problems. Thus, the probability of the next problem answered correctly at c_t of Seg_z can be obtained from y_t. In that respect, Eq. 7 and 8 are still valid for DKT-DSC. The

output y_t of both DKT and DKT-DSC is the same which provides the predicted probability for a particular problem.
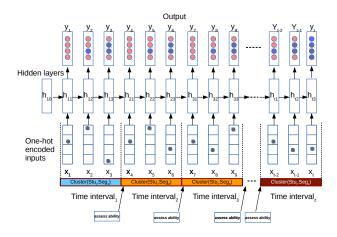

Fig. 4. DKT-DSC prediction in each time interval (each segment) is associated with a distinct group (cluster) throughout interactions of a student with the system.

Figure 4 illustrates how DKT-DSC model has been adapted by incorporating student's learning ability as distinct group information at each time interval (each segment) to improve individualization in knowledge tracing. The colour at each time interval at the input layer represents which group a student belongs to at that time interval according to her learning ability. Note that without incorporating student's ability, DKT-DSC model is the same as the standard DKT model.

By adding this cluster information $Cluster(Stu_i, Seg_z)$ of what group the student belongs to, we ensure that these high-level statistics are still available to the model for making its predictions throughout the whole academic year. This is what the DKT model does, treating all students in same way without considering their learning abilities. On the contrary, DKT-DSC uses clustering to find a group of students with similar ability by using their ability profile data at different time intervals. Tracing student's knowledge in each different group can provide more effectiveness in student's performance prediction.

Finally, we summarize the characteristics of each model in this paper in Table I.

TABLE I
COMPARISON OF DIFFERENT MODELS

	IRT	PFA	BKT	DKT	DKT-DSC
Use of student's ability	Yes	No	No	No	Yes
Use of item difficulty	Yes	No	No	No	No
Use of single skill	Yes	No	Yes	Yes	Yes
Use of multiple skill	No	Yes	No	No	No
Learn on ordered sequence	No	No	Yes	Yes	Yes

IV. DATASETS

In order to validate the proposed model, we tested it on four public datasets from two distinct tutoring scenarios in which students interact with a computer-based learning system in educational settings.

- The ASSISTment system[2] is an online tutoring system that was first created in 2004 which engages middle and high-school students with scaffolded hints in their math problems. If students working on ASSISTments answer a problem correctly, they are given a new problem. If they answer it incorrectly, they are provided with a small tutoring session where they must answer a few questions that break the problem down into steps. Datasets are: ASSISTments 2009-2010 skill builder data set, ASSIST-ments 2012-2013, ASSISTments 2014-2015.
  In all datasets, problems are usually tagged with just one skill, but a rare few may be associated with two or three skills. It typically depends on the structure given by the content creator. Some researchers separate a record with multiple skills into multiple single skill records by duplicating. Wilson [6] claimed that this type of data processing can artificially boost prediction results significantly, because these duplicate rows can be accounted for approximately 25% of the records in the Assitment09 dataset for DKT models. So we removed duplicate and multiple-skill repeated records in all datasets for the fairness of comparison.
- KDD Cup: The PSLC DataShop released several data sets derived from Carnegie Learning's Cognitive Tutor. Algebra 2005-2006 [19] is a development dataset released during the KDD Cup 2010 competition[3]. In this dataset, the problems are associated with multiple skills. So we regard a subset of multiple skills as a new skill [20].

We evaluate models described above with four datasets from two separate real world tutors. The experimental results show how the models perform across different datasets. Only the first correct attempts to original problems are considered in our experiment.

To the best of our knowledge, these are the largest publicly available knowledge tracing datasets.

TABLE II
OVERVIEW OF DATASETS

Dataset	Number of			Description
	Skills	Students	Records	
ASSISTments	123	4,163	278,607	2009-2010 [21]
	198	28,834	2,506,769	2012-2013 [22]
	100	19,840	683,801	2014-2015 [20]
Cognitive Tutor	437	574	808,775	KDD Cup 2010 [19]

V. EXPERIMENTAL STUDY

DKT-DSC is extended from the original DKT algorithm, and is combined with the k-means clustering method with a Euclidean distance. Ten iterations were made in the training stage. DKT-DSC and DKT share the same loss function, with 200 fully-connected hidden nodes for each hidden layer. For speeding up the training process, mini-batch stochastic

---

[2]https://sites.google.com/site/assistmentsdata
[3]https://pslcdatashop.web.cmu.edu/KDDCup/downloads.jsp

gradient descent is used to minimize the loss function. The batch size for our implementation is 32, corresponding to 32 split sequences from each student. We train the model with a learning rate of 0.01 and dropout is also applied for avoiding overfitting [23].

In our experiment, 5 fold cross-validations are used to make predictions on both datasets. Each fold involves randomly splitting each dataset into 80% training data and 20% test data at the student level. So both training and test datasets contain response records from different students. Training for clustering is performed only using data from students in the training dataset. We use EM to train BKT and the limit of iterations is set to 200. We learn models for each skill and make predictions separately, then the results for each skill are averaged. For DKT and DKT-DSC, we set the number of epochs to 100. All these models are trained and tested on the same sets of data. Next response of a student is predicted by using current and previous response sequence in chronological order.

We compare our model with state-of-the-art models: IRT [6], BKT [11], PFA [13], DKT [4]. But we do not compare with other variant models, because those are more or less similar and do not show significant difference in performance. For IRT, we apply the code from Knewton [6] and the code for DKT is from WPI [20]. For DKT, we use the same setting of parameters as DKT-DSC and also apply segmentation for a fair comparison. Predicted sequences of student performance by each model are tabulated and evaluated in terms of Area Under the Curve (AUC) and Root mean squared error (RMSE). AUC provides a robust metric where the value to predict is binary, as it is the case of our datasets. An AUC of 0.50 represents the score achieved by random guess. We set AUC 0.61 of BKT as a baseline in our experiment.

TABLE III
AUC RESULT FOR ALL DATASETS

Datasets	Model				
	BKT	IRT	PFA	DKT	DKT-DSC
ASSISTments09	0.67	0.75	0.70	0.73	**0.91**
ASSISTments12	0.61	0.74	0.67	0.72	**0.87**
ASSISTments14	0.64	0.67	0.69	0.72	**0.87**
Cognitive Tutor	0.61	**0.81**	0.76	0.79	**0.81**

In Table III, DKT-DSC performs significantly better than state-of-the-art models in all datasets. On the ASSISTments09 dataset, compared with the standard DKT which has an AUC of 0.73, our DKT-DSC model achieves an AUC of 0.92, which represents a significant gain of 26%. On the ASSISTments12 dataset with 2.5 million records, the result shows 11% increase, AUC 0.80 in DKT-DSC compared with AUC 0.72 in the original DKT. In the latest ASSISTments14 dataset, DKT-DSC achieves an improvement of 19% over the original DKT. In the Cognitive Tutor dataset, DKT-DSC also achieves about 1% gain with AUC=0.81 while the original DKT has AUC=0.79. As for other algorithms, IRT also provides a slight improvement over the original DKT in all datasets but

DKT-DSC performs significantly better than both DKT and IRT. Note that Problem ID is not provided in the original ASSISTments14 dataset. So we use Skill ID as Problem ID for the IRT model, and that is why IRT only gets a AUC of 0.67. In all models described above, only the IRT model learns the problem difficulty while all other models only rely on skills.

TABLE IV
RMSE RESULT FOR ALL DATASETS

Datasets	Model				
	BKT	IRT	PFA	DKT	DKT-DSC
ASSISTments09	0.46	0.44	0.45	0.45	**0.33**
ASSISTments12	0.51	0.44	0.44	0.43	**0.35**
ASSISTments14	0.51	0.44	0.42	0.42	**0.35**
Cognitive Tutor	0.47	0.37	0.39	**0.36**	**0.36**

In Table IV, when we compare the models in terms of RMSE, BKT is 0.46 in ASSISTments09, 0.51 in ASSISTments12 and 0.47 in Cognitive Tutor. RMSE results of DKT-DSC in all dataset are under 0.40 while that of all other models are no less than 0.42 (except IRT, PFA and DKT in the Cognitive Tutor dataset). According to these results, DKT-DSC outperforms in all ASSISTments datasets and shows a slightly better performance in the Cognitive Tutor dataset. All of the above experiments are conducted on the time interval containing 20 attempts and 8 clusters (groups of students).

TABLE V
AUC OF DKT-DSC WITH DIFFERENT LENGTHS OF TIME INTERVAL

Time interval	Datasets			
	Ass09	Ass12	Ass15	KDD
20	**0.91**	**0.81**	**0.86**	0.81
30	0.88	0.80	0.82	0.81
50	0.87	0.80	0.78	0.81
100	0.82	0.77	0.73	**0.82**

Segmentation of student responses into fixed-time intervals is applied for DKT-DSC (on 4 groups of students) and tested with each time interval containing 20, 30, 50, and 100 attempts in this experiment. The performance of DKT-DSC is described in Table V. DKT-DSC performs better when each time interval contains 100 attempts in KDD dataset with 574 (joint) skills. It can be considered as a small dataset according to the numbers of skills contained in it. So it may inefficiently identify the student's group because of the sparsity of data. When it contains sufficient amount of attempts in each time interval, it shows a better performance.

TABLE VI
AUC OF DKT-DSC WITH DIFFERENT NUMBER OF GROUPS OF STUDENTS

# clusters	Datasets			
	Ass09	Ass12	Ass15	KDD
2	0.88	0.78	0.83	0.79
4	**0.91**	0.81	0.86	0.80
6	**0.91**	0.84	0.86	0.80
8	**0.91**	**0.87**	**0.87**	**0.81**

As well as with different lengths of time interval, various number of clusters provide different performances as described in Table VI. According to experimental results, 8 clusters with 20 number of attempts in each time interval is the best parameter for DKT-DSC.

VI. Conclusion and Future Work

In this paper, we proposed a new model, DKT-DSC that assesses student's learning ability at each time interval and dynamically assigns a student into a distinct group of students with the same ability. A student's knowledge is traced based on the group which she belongs to at each time interval. Experiments with four datasets show that the proposed model performs statistically and significantly better than state-of-the-art models. DKT assumes all students have the same learning ability and only tracks the improvement of knowledge in a skill sequence without considering difference between abilities of each student and learning rate. In comparison, our model improves over DKT by capturing the student's ability over time. Assessing student's ability in this way gives the model critical information in the prediction of student performance in their next time interval and tracing their knowledge where abilities of the students evolve dynamically. We individualize the input vector by taking both student's ability and practicing skill into account. Instead of using the skill level alone, incorporating student's ability in terms of group information in DKT-DSC yields an improvement in the prediction of performance. Dynamically assessing student's ability at each time interval plays the critical role and helps the DKT-DSC model capture more variance in the data, leading to more accurate predictions.

In our future work, we will adapt this model to problems with multiple associated subskills in the system and apply it in the recommendation of problems with multiple associated skills. Problems for practice should be recommended according to the knowledge level and the ability a student possesses. The significant gain obtained by DKT-DSC can make a difference in current knowledge tracing respect. Further investigation on the potential application of DKT-DSC to other content recommendations (movies and other commercial products) will also be considered.

VII. Acknowledgements

References

[1] J. S. Brown and R. R. Burton, "Diagnostic models for procedural bugs in basic mathematical skills," *Cognitive science*, vol. 2, no. 2, pp. 155–192, 1978. I

[2] M. C. Polson and J. J. Richardson, *Foundations of intelligent tutoring systems*. Psychology Press, 2013. I

[3] A. T. Corbett and J. R. Anderson, "Knowledge tracing: Modeling the acquisition of procedural knowledge," *User modeling and user-adapted interaction*, vol. 4, no. 4, pp. 253–278, 1994. I, II-B

[4] C. Piech, J. Bassen, J. Huang, S. Ganguli, M. Sahami, L. J. Guibas, and J. Sohl-Dickstein, "Deep knowledge tracing," in *Advances in Neural Information Processing Systems*, 2015, pp. 505–513. I, II-D, V

[5] M. C. Desmarais and R. S. Baker, "A review of recent advances in learner and skill modeling in intelligent learning environments," *User Modeling and User-Adapted Interaction*, vol. 22, no. 1-2, pp. 9–38, 2012. II

[6] K. H. Wilson, Y. Karklin, B. Han, and C. Ekanadham, "Back to the basics: Bayesian extensions of irt outperform neural networks for proficiency estimation," *arXiv preprint arXiv:1604.02336*, 2016. II-A, II-A, IV, V

[7] W. J. van der Linden and R. K. Hambleton, *Handbook of modern item response theory*. Springer Science & Business Media, 2013. II-A

[8] J. González-Brenes, Y. Huang, and P. Brusilovsky, "General features in knowledge tracing to model multiple subskills, temporal item response theory, and expert knowledge," in *The 7th International Conference on Educational Data Mining*. University of Pittsburgh, 2014, pp. 84–91. II-A

[9] C. Ekanadham and Y. Karklin, "T-skirt: Online estimation of student proficiency in an adaptive learning system," *arXiv preprint arXiv:1702.04282*, 2017. II-A

[10] R. S. d Baker, A. T. Corbett, and V. Aleven, "More accurate student modeling through contextual estimation of slip and guess probabilities in bayesian knowledge tracing," in *International Conference on Intelligent Tutoring Systems*. Springer, 2008, pp. 406–415. II-B

[11] R. S. Baker, A. T. Corbett, and V. Aleven, "More accurate student modeling through contextual estimation of slip and guess probabilities in bayesian knowledge tracing," *Lecture Notes in Computer Science*, vol. 5091, pp. 406–415, 2008. II-B, V

[12] Z. Pardos and N. Heffernan, "Kt-idem: introducing item difficulty to the knowledge tracing model," *User Modeling, Adaption and Personalization*, pp. 243–254, 2011. II-B

[13] P. I. Pavlik Jr, H. Cen, and K. R. Koedinger, "Performance factors analysis–a new alternative to knowledge tracing." *Online Submission*, 2009. II-C, II-C, V

[14] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997. II-D

[15] A. Merceron and K. Yacef, "Clustering students to help evaluate learning," in *Technology Enhanced Learning*. Springer, 2005, pp. 31–42. III-A

[16] S. Trivedi, Z. A. Pardos, and N. T. Heffernan, "Clustering students to generate an ensemble to improve standard test score predictions," in *International Conference on Artificial Intelligence in Education*. Springer, 2011, pp. 377–384. III-A

[17] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 14. Oakland, CA, USA, 1967, pp. 281–297. III-A4

[18] G. Ball and I. Hall Dj, "A novel method of data analysis and pattern classification. isodata, a novel method of data analysis and pattern classification. tch. report 5ri, project 5533," 1965. III-A4

[19] A. Corbett, "Cognitive computer tutors: Solving the two-sigma problem," *User Modeling 2001*, pp. 137–147, 2001. IV, II

[20] X. Xiong, S. Zhao, E. Van Inwegen, and J. Beck, "Going deeper with deep knowledge tracing." in *EDM*, 2016, pp. 545–550. IV, II, V

[21] L. Razzaq, M. Feng, G. Nuzzo-Jones, N. Heffernan, K. Koedinger, B. Junker, S. Ritter, A. Knight, C. Aniszczyk, S. Choksey *et al.*, "The assistment project: Blending assessment and assisting," in *Proceedings of the 12th Annual Conference on Artificial Intelligence in Education*, 2005, pp. 555–562. II

[22] M. Feng, N. Heffernan, and K. Koedinger, "Addressing the assessment challenge with an online system that tutors as it assesses," *User Modeling and User-Adapted Interaction*, vol. 19, no. 3, pp. 243–266, 2009. II

[23] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014. V