Impact of Methodological Choices on the Evaluation of Student Models

Tomáš Effenberger and Radek Pelánek

Masaryk University Brno
Czech Republic
tomas.effenberger@mail.muni.cz, pelanek@fi.muni.cz

Abstract. The evaluation of student models involves many methodological decisions, e.g., the choice of performance metric, data filtering, and cross-validation setting. Such issues may seem like technical details, and they do not get much attention in published research. Nevertheless, their impact on experiments can be significant. We report experiments with six models for predicting problem-solving times in four introductory programming exercises. Our focus is not on these models per se but rather on the methodological choices necessary for performing these experiments. The results show, particularly, the importance of the choice of performance metric, including details of its computation and presentation.

1 Introduction

Student modeling [4, 12] is at the core of many techniques in the field of artificial intelligence in education. A key element of research and development of student modeling techniques is the comparison of several alternative models. Such comparisons are used to choose models (and their hyper-parameters) to be used in real-life systems, to judge the merit of newly proposed techniques, and to determine priorities for future research.

Results of model comparison can be influenced by methodological decisions made in the experimental setting of the comparison, e.g., the exact manner of dividing data into training and testing set [13], the choice of metrics [10], or the treatment of outliers. These issues do not get much attention unless suspiciously good results are reported, e.g., as in the case of deep knowledge tracing paper [15], which reported significant improvement in predictive accuracy, prompting several research groups to probe the results and to identify several methodological problems in the evaluation [8, 16, 17].

The importance of methodological choices was recently discussed in [13], but using mostly simplified examples and simulations. We perform an exploration of the impact of methodological choices using real data. We use data from introductory programming exercises, where students are expected to construct a program to solve a given problem. We compare models that predict the problem-solving time for the next item. We use data from four types of exercises with different characteristics—this allows us to explore the generalizability of our observations.

For the data, we perform a comparison of six student models. Our focus is not on these models (which typically get attention in reported results), but rather on the methodological choices done in the experimental setting and on the impact of these choices on results. We explicitly describe the choices that need to be made and show specific examples that illustrate how these choices influence the results of model comparison. Our results highlight the importance of the choice of performance metric, including details of its computation, processing, and reporting.

2 Setting

To analyze the impact of methodological choices, we measure the performance of six student models for predicting the time to solve the next problem in four programming exercises. Predicting problem-solving times is less explored than predicting binary success, yet it is a more informative measure of performance for problems that take more than just a few seconds to solve [14]. As the problem-solving times are usually approximately log-normal [14], our models and evaluations work with the log-transformed times (denoted 'log-time').

2.1 Data

We use data from four introductory programming exercises, each containing 70–100 items divided into 8–12 levels. Table 1 provides an overview of these exercises. In the Arrows exercise, students place commands (usually directions to follow) directly into the grid with the game world. In the Robot exercise [5] and Turtle graphics [2], students create programs using a block-based programming interface [1]. In the last exercise, students write Python code to solve problems with numbers, strings, and lists. In all cases, the problems require at most 25 lines of code and are solved in between 10 seconds and 5 minutes by most of the students.

2.2 Student Models

In all experiments, we compare the following student models for predicting problem-solving times. All these models can be first fitted offline and then evaluated online on previously unseen students.

Table 1. Programming exercises and data used in experiments.

Exercise	Items	Students	Successful attempts	Median time
Arrows	94	13,000	182,000	32 sec
Robot	85	10,800	146,000	51 sec
Turtle	77	10,100	87,000	81 sec
Python	73	1,400	17,000	174 sec

1. *Item average (I-Avg)*: a baseline model predicting average log-time for a given item.
2. *Student-item average (SI-Avg)*: a simple model predicting item average time reduced by a naive estimate of the student's skill. The skill is computed from the previous student's attempts as the average deviation between the observed log-time and the item average log-time. To avoid overfitting, the estimate is regularized by adding five pseudo-observations of zero deviations.
3. *t-IRT*: a one-parameter item response theory model (1PL IRT) adapted for problem-solving times [14]. The model has the same set of parameters as the SI-Avg, but now they are optimized to minimize RMSE (with L2 penalty), using regularized linear least squares regression. The skill, which is assumed to be constant, is in the online evaluation phase estimated using the regularized mean deviation in the same way as in the SI-Avg model.
4. *t-AFM*: an additive factors model [3] adapted for problem-solving times. The Q-matrix is constructed from the levels in each exercise. Three additional modifications to the standard AFM were necessary for a reasonable performance: a difficulty parameter for each item, log-transformation of the practice opportunities counts (only solved attempts are considered), and an online estimate of the prior skill, using the same regularized mean deviation as for the SI-Avg and t-IRT.
5. *Elo*: a model based on the Elo rating system adapted for problem-solving times [11]. It tracks a single skill for each student and a single difficulty for each item. After each observed attempt, the skill and the difficulty are updated in proportion to the prediction error. In contrast to SI-Avg and t-IRT, the Elo model assumes changing skill, which is reflected by holding the learning rate for the estimate of the student's skill constant. On the other hand, the difficulties are assumed not to change over time, so their learning rate is inversely proportional to the number of observations.
6. *Random forest (RF)*: a generic machine learning model utilizing ensemble of decision trees, with the following features: item and level (using one-hot-encoding scheme), problem-solving time on recent items (using exponential moving average), and the numbers of items the student had already solved under and above several time thresholds, both in total and in the individual levels.

To select reasonable hyper-parameters for the models (e.g., the number of pseudo-observations for the online estimate of the prior skill, or the number of trees and the maximum depth for the Random forest), we used a subset of data from the Robot exercise (first 50,000 attempts). These data were not used for the subsequent experiments, and the hyper-parameters were not modified for the other exercises.

2.3 Evaluation Approach

To explore the impact of a set of methodological choices, we compare the results of student models evaluation using these choices. For each exercise and each

set of methodological choices, we use the following evaluation approach, which corresponds to a common practice in the evaluation of student models.

First, we apply data preprocessing choices, such as filtering of students with few attempts and capping observed solving times. Then we perform student-level k-fold cross-validation [13], i.e., all attempts of a single student are all assigned to one of the k folds (we use $k = 10$). For each fold and each model, we fit the model parameters on a training set ($k - 1$ folds) and then evaluate the performance of the model on a testing set (the remaining fold). The evaluation phase is online, i.e., the models can update their parameters (e.g., the skill of a student) after each observed attempt. The performance of models is measured by comparing the predicted and observed problem-solving times, using Root Mean Square Error (RMSE) as the default performance metric. Finally, we report the mean value and the standard deviation of the metric across folds and also the average rank of the model according to the metric.

We evaluate the impact of several methodological choices and their interactions: the choice of predictive accuracy metric and the details of the computation and reporting of the metric, division of data into training and testing sets, filtering of the data, and treatment of outliers. When reporting the observed results, we face the trade-off between conciseness and representativeness. Often, we illustrate the impact of a given choice on a single exercise; when we do so, we always report to which extent the trends observed in this exercise generalize to the other three and provide the same plots for the other exercises as a supplementary material available at `github.com/adaptive-learning/aied2020`.

3 Metrics

Although student models can be evaluated and compared from many perspectives [7], the primary criterion used to compare models is the predictive accuracy. The predictive accuracy is quantified by a performance metric [10], i.e., a function that takes a vector of predictions and a vector of observations and produces a scalar value. The choice of a metric used for model comparison involves quite a large number of (often under-reported) decisions.

3.1 Normalization and Stability of Results

We start by a discussion of the processing and presentation of results since it also influences the presentation of our results in the rest of the paper. To check the stability of model comparisons, it is useful to have not just a single value of a metric but to run repeated experiments and study the stability of results. A straightforward approach is to perform k-fold cross-validation and report the mean value of a metric and its standard deviation.

Such a presentation can be, however, misleading. Fig. 1 provides a specific illustration. The left part of the figure shows the basic approach to evaluation where we compare the values of RMSE directly. This presentation shows that the results for individual models overlap to a large degree; we could be tempted to

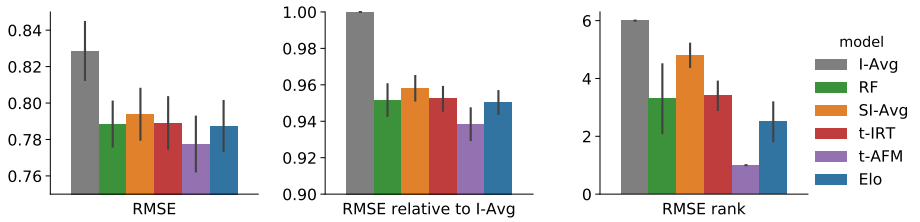

Fig. 1. Comparison of student models for the Robot exercise using RMSE (left), RMSE relative to the baseline (center), and order of the models according to RMSE (right). The vertical bars show standard deviations computed from 10 cross-validation folds.

conclude from this that the accuracy of the studied models is not very different. However, it may be that the observed variability is due to variability across folds, not due to the variability of models predictive ability.

The variability caused by data can be, for example, due to the presence of unmotivated students with chaotic and hard to predict behavior and their uneven distribution across folds. To reduce this variability, we can normalize the metric value. In Fig. 1 we report two types of such normalization: A) RMSE relative to a baseline model (per fold), B) RMSE rank among compared models (per fold). As Fig. 1 shows, these normalizations give quite a different picture concerning how consistent are the differences in model performance. Consider, for instance, comparison of SI-Avg and t-AFM. While the distributions of their RMSEs across folds largely overlap, exploring their ranks reveals that t-AFM has consistently better performance than SI-Avg. This is not an isolated case; for all four exercises, there are some pairs of models whose distributions of RMSEs largely overlap, while the distributions of the ranks do not.

We do not claim that the normalized approaches are better. It may be that one model is consistently better (which is highlighted by the rank approach), but the differences are consistently small and thus practically not important. Reporting both the absolute and normalized RMSEs gives a fuller picture than using just one of the approaches alone.

In this paper, we usually report both the absolute RMSEs and the ranks. The ranks often provide more insight into the impact of methodological choices, since they are more robust to the noise within the folds, and this makes the differences between the models more salient. Additionally, the ranking approach allows us to study the impact of different metrics, which we look at next.

3.2 RMSE versus MAE

There is a large number of metrics, particularly for the case of models predicting probabilities [10]. Our default choice, RMSE, is a commonly used metric. For the case of predicting continuous values (as is the case of the used logarithm of time), another natural choice is Mean Absolute Error (MAE). To explore the potential impact of metric choice, we compare these two metrics.

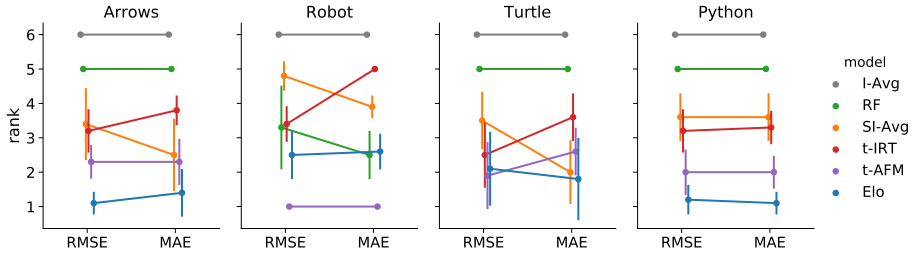

Fig. 2. Comparison of model orderings under RMSE and MAE metrics in four programming exercises. The vertical bars show standard deviations of the ranks computed from 10 cross-validation folds.

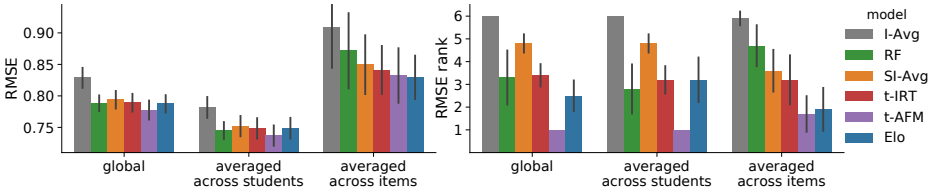

Fig. 3. Comparison of models for the Robot exercise using RMSE averaged either globally, across students, or across items. The vertical bars show standard deviations of the ranks computed from 10 cross-validation folds.

Fig. 2 shows the results of this comparison across our four datasets. The figure visualizes the ranking of models and shows that the results are mostly stable with respect to the choice of metric. However, there are cases where the choice of metric influences results. Particularly, there is a mostly consistent trend with respect to SI-Avg and t-IRT models: t-IRT achieves better results for the RMSE metric, whereas SI-AVG is better for the MAE metric.

3.3 Averaging

Another decision is the approach to the averaging in the computation [13]. We can use either global computation (treat all observations equally), averaging across students (compute RMSE per student and then compute an average), or averaging across items (compute RMSE per item and then compute an average). These can produce different results, particularly when the distribution of answers is skewed across items or students. For all four datasets we use, that is indeed the case. Fig. 3 shows an example of the Robot exercise, where the impact of the averaging is the most pronounced. In this exercise, the averaging across items leads to considerably higher values of RMSE and even to some changes in the ordering of the models.

To get better insight, we can disaggregate RMSE into individual levels (groups of items of similar difficulty) or populations (e.g., groups of students according to their activity or performance). Fig. 4 shows an example of such per-level RMSE

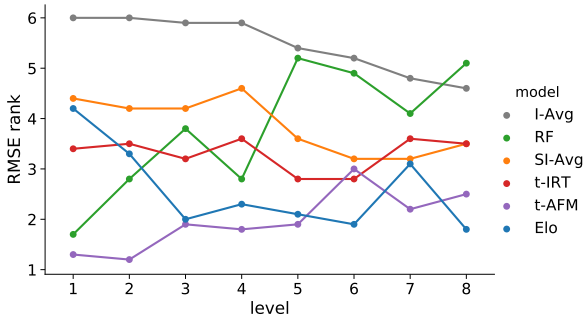

Fig. 4. Average ranks of student models in individual levels of the Robot exercise.

decomposition for the Robot exercise. Note, particularly, the performance of the Random forest model; it is one of the best models in the initial levels, while one of the worst in the advanced levels. Since students mostly solved items in the first few levels, the global averaging (shown in Fig. 3) favors this model.

We expected that the benefits of the more complex models (like Random Forest and t-AFM) would manifest especially in the last levels, where the complex models can make use of richer students' history. However, the results, for all four programming datasets, show that the trends are exactly opposite: the mean ranks of all models get closer to each other in higher levels. Probably, the skew of the data leads the complex models to overfocus on the first few levels at the expense of the less solved last advanced levels; furthermore, while the models can benefit from more data about the students, they might be seriously hampered by less data for the items.

4 Data Processing

Another set of methodological choices concerns the processing of data: Do we perform some data filtering? How do we treat outliers? How exactly do we divide data into a training and testing set?

4.1 Filtering and Outliers

The data from learning systems are noisy, e.g., due to off-task behavior, guessing, or cheating. In order to reduce the impact of this noise on the results of experiments, it may be meaningful to perform some data preprocessing, for example:

- filtering students with small activity (rationale: students with small activity are often just experimenting with the system, and thus there is higher chance that their behavior is noisy),
- filtering items with small activity (rationale: models do not have enough information to provide good predictions for such items),

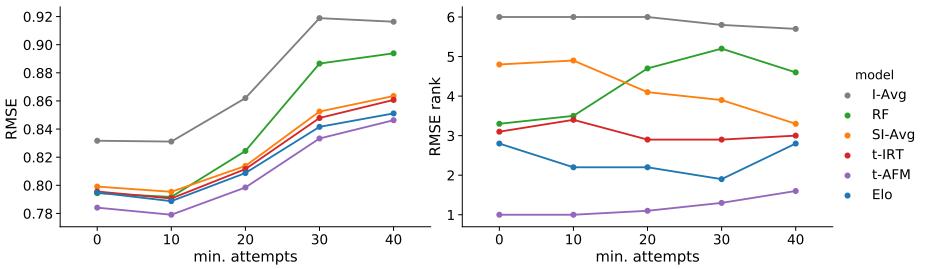

Fig. 5. Average RMSE and RMSE ranks of student models when students with few attempts are filtered (Robot exercise). The filtering of students with at least 10, 20, 30, and 40 attempts results in keeping 86%, 47%, 22%, and 13% of the original attempts.

– removing or capping outliers, i.e., too high problem-solving times (rationale: very high problem-solving times are often caused by some disruption in solving activity, not by poor student skill or high item difficulty).

In some cases, the choice of a filtering threshold can have a pronounced effect on the absolute RMSE, even much higher than the differences in RMSE caused by using a different model. This is illustrated in Fig. 5 for the case of filtering students in the Robot exercise. We observed similar trends in all four datasets and for other data preprocessing choices: high impact on the absolute values of RMSE, but usually a negligible impact on the ranking, unless the thresholds are rather extreme.

As in the case of disaggregating RMSE per level, our initial intuition about the relative merits of the filtering for the simple and complex models was incorrect: we expected the complex models to benefit more from severe filtering since the remaining students have a long history that the complex models can utilize, while the simple models cannot. Nevertheless, both the absolute RMSE and ranks of the complex models actually increase with more severe filtering since the filtering results in an increased proportion of the data for the items with few attempts and less training data overall, which is a more significant issue for the models with many parameters.

4.2 Data Division for Cross-Validation

Reported comparisons of student models often use "k-fold cross-validation" without further specification of the division of data into folds. Since the data from learning systems have an internal structure (mapping to items and students, temporal sequences), there are many ways in which the division of data can be performed [13].

To explore the impact of this choice, we compare two natural choices: student-level cross-validation and time-series cross-validation. In the student-level k-fold cross-validation, all attempts of a single student are assigned randomly to one of the k folds. The relative order of the attempts is preserved (there is no shuffling), and all groups contain approximately the same number of students. This

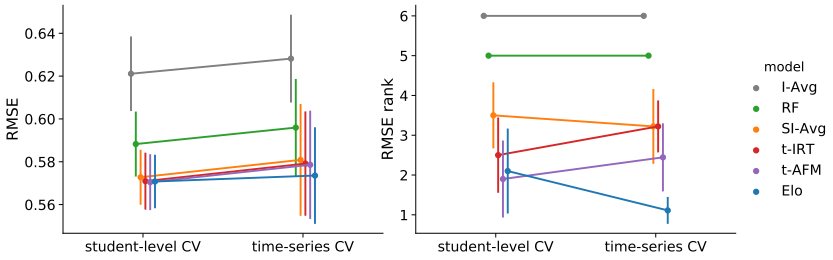

Fig. 6. Comparison of models for the Turtle exercise using either student-level or time-series cross-validation strategy.

cross-validation strategy ensures that the models cannot use future attempts of a given student to predict her past, but it does not prevent them from using future attempts of the *other* students. In contrast, the time-series cross-validation creates the folds strictly by time, always using only the preceding folds for the training.

Our analysis shows that the choice of cross-validation strategy can influence the results of experiments. Fig. 6 shows the results for the Turtle dataset: with respect to ranking, the Elo model is a clear winner when the time-series cross-validation is used, whereas in the case of student-level cross-validation, it has similar results as other models. An analysis of the dataset shows that it contains a temporal pattern—the performance of consecutive students on each item is correlated. Although this correlation is quite weak and the Elo model is not explicitly designed to exploit it, the presence of this pattern is sufficient to impact the results of the comparison.

In the other exercises, especially in the Arrows and Python, the impact of the cross-validation strategy is rather small. However, in these two exercises, the Elo model is already the best model even when the student-level cross-validation is used. In the other five models, the students in the test set cannot influence the predictions for the subsequent students, so these models cannot exploit the described temporal pattern.

5 Discussion

The comparison of student models involves many methodological choices, which can influence the results of the evaluation. This situation is not unique to student modeling; similar problems are well-known in other fields, e.g., Gelman and Loker [6] discuss statistical analysis of experiments with examples from social science.

In this work, we highlight and explore methodological choices that are typically encountered in the evaluation of student models. Insufficient attention to these details poses several risks:

– Possibility of "fishing" for choices that present a researcher's favorite technique (e.g., a newly proposed method) in a favorable light.

– Missing of potentially interesting results due to some arbitrary methodological choice that masks important differences between models.
– Misleading comparisons of models, which were evaluated by slightly different methodologies (differing in details that are undocumented or over which authors gloss over).

The basic step to mitigating these risks is the awareness of the available choices and their explicit and clear description in research papers. Some kind of preregistration procedure [9] can further strengthen the credentials of student model comparisons.

Typically, researchers in student modeling and developers of adaptive learning systems are interested primarily in student models and do not want to spend much time exploring methodological choices. Experiments are not inherently unstable—many decisions have only a small impact on results. It is thus useful to know which choices deserve most focus. This, of course, depends to a large degree on a particular setting and it is unlikely that some completely universal guidelines can be found. However, reporting experience from a variety of comparisons should lead to a set of reasonable recommendations.

We have performed our experiments in the domain of problem-solving activities and for the student models predicting problem-solving times. In this setting, the results show the importance of the choice of a performance metric and of the details of its processing and presentation. Specifically, our results show that there are large differences between the presentation of results of cross-validation across folds in terms of the absolute value of metrics, relative values (normalized by baseline performance per fold), and rankings of performance per fold. On the other hand, filtering of data and treatment of outliers have a relatively small impact on the ranking of models (for reasonable choices of thresholds).

Our results also clearly illustrate that the absolute values of performance metrics depend on details of the evaluation methodology and properties of a specific dataset. The differences in metric values are typically larger among different evaluation settings than among different models. Consequently, it is very dangerous to compare metric values to results reported in research papers even when using the same dataset (as done, for example, in the deep knowledge tracing paper [15]). Comparisons make sense only when we are absolutely sure that the computation of metric values is done in exactly the same way. Since there are many subtle choices that influence metric values, this can be in practice best done by comparing only models that use the same implementation of an evaluation framework.

Supplementary Materials

For all the presented plots, we provide their analogues with all four exercises as supplementary materials at `github.com/adaptive-learning/aied2020`. The numbering of the supplementary plots corresponds to the numbering in the paper.

References

1. Bau, D., Gray, J., Kelleher, C., Sheldon, J., Turbak, F.: Learnable programming: Blocks and beyond. Communications of the ACM **60**(6), 72–80 (2017)
2. Caspersen, M.E., Christensen, H.B.: Here, there and everywhere – on the recurring use of turtle graphics in cs1. In: ACM International Conference Proceeding Series. vol. 8, pp. 34–40 (2000)
3. Cen, H., Koedinger, K., Junker, B.: Learning factors analysis–a general method for cognitive model evaluation and improvement. In: Proc. of Intelligent Tutoring Systems. pp. 164–175. Springer (2006)
4. Desmarais, M.C., Baker, R.S.: A review of recent advances in learner and skill modeling in intelligent learning environments. User Modeling and User-Adapted Interaction **22**(1-2), 9–38 (2012)
5. Effenberger, T., Pelánek, R.: Towards making block-based programming activities adaptive. In: Proceedings of Learning at Scale. p. 13. ACM (2018)
6. Gelman, A., Loken, E.: The garden of forking paths: Why multiple comparisons can be a problem, even when there is no fishing expedition or p-hacking and the research hypothesis was posited ahead of time. Department of Statistics, Columbia University (2013)
7. Huang, Y., González-Brenes, J.P., Kumar, R., Brusilovsky, P.: A framework for multifaceted evaluation of student models. International Educational Data Mining Society (2015)
8. Khajah, M., Lindsey, R.V., Mozer, M.C.: How deep is knowledge tracing? In: Proceedings of Educational Data Mining (2016)
9. Nosek, B.A., Ebersole, C.R., DeHaven, A.C., Mellor, D.T.: The preregistration revolution. Proceedings of the National Academy of Sciences **115**(11), 2600–2606 (2018)
10. Pelánek, R.: Metrics for evaluation of student models. Journal of Educational Data Mining **7**(2) (2015)
11. Pelánek, R.: Applications of the elo rating system in adaptive educational systems. Computers & Education **98**, 169–179 (2016)
12. Pelánek, R.: Bayesian knowledge tracing, logistic models, and beyond: an overview of learner modeling techniques. User Modeling and User-Adapted Interaction **27**(3), 313–350 (2017)
13. Pelánek, R.: The details matter: methodological nuances in the evaluation of student models. User Modeling and User-Adapted Interaction **28**, 207–235 (2018). https://doi.org/10.1007/s11257-018-9204-y
14. Pelánek, R., Jarušek, P.: Student modeling based on problem solving times. International Journal of Artificial Intelligence in Education pp. 1–27 (2015)
15. Piech, C., Bassen, J., Huang, J., Ganguli, S., Sahami, M., Guibas, L.J., Sohl-Dickstein, J.: Deep knowledge tracing. In: Advances in Neural Information Processing Systems. pp. 505–513 (2015)
16. Wilson, K.H., Xiong, X., Khajah, M., Lindsey, R.V., Zhao, S., Karklin, Y., Van Inwegen, E.G., Han, B., Ekanadham, C., Beck, J.E., et al.: Estimating student proficiency: Deep learning is not the panacea. In: Proceedings of Neural Information Processing Systems, Workshop on Machine Learning for Education (2016)
17. Xiong, X., Zhao, S., Van Inwegen, E., Beck, J.: Going deeper with deep knowledge tracing. In: Proceedings of Educational Data Mining. pp. 545–550 (2016)