Parameterized Exercises in Java Programming: using Knowledge Structure for Performance Prediction

Shaghayegh Sahebi[1], Yun Huang[1], and Peter Brusilovsky[1,2]

[1] Intelligent Systems Program, University of Pittsburgh
[2] School of Information Sciences, University of Pittsburgh
{shs106,yuh43,peterb}@pitt.edu

Abstract. In this paper, we study the effect of using domain knowledge structure on predicting student performance with parameterized Java programming exercises. Domain knowledge structure defines connections between elementary knowledge items. While known to be beneficial in general, it has not been used to predict performance. We compare five different approaches for this purpose: Bayesian Knowledge Tracing (BKT), Performance Factor Analysis (PFA), and three dimensional Bayesian Probabilistic Tensor Factorization (3D-BPTF), that are not able to take into account knowledge structure; and four-dimensional Bayesian Probabilistic Tensor Factorization (4D-BPTF) and Feature-Aware Student Knowledge Tracing (FAST), that can take into account knowledge structure. We approach the problem using both topic-level and question-level Knowledge Components (KCs) and test the methods on a dataset of parameterized questions. Our work is the first in the field that models students' behavior in a four dimensional tensor. Our experiments show that, when having only the knowledge-item-level information, all of the models work similarly in predicting student performance, but adding the topic-level information that integrates knowledge items changes the performance of these models in different directions.

1 Introduction

Parameterized questions and exercises have recently emerged as an important tool for online assessment and learning. A parameterized question is essentially a template for the question, created by an author. At presentation time, the template is instantiated with randomly generated parameters. Parameterized questions are considerably harder to implement than traditional “static" questions since they need some special infrastructure to dynamically check the correctness of student answers. However, the benefits offered by this technology make this additional investment worthwhile. During assessment, they can be used to produce online individualized assessments for large classes minimizing cheating problems. Even more importantly, in the self-assessment context, the same question can be used again and again with different parameters, allowing every student to achieve understanding and mastery. These properties of parameterized exercises made them very attractive for the large-scale online systems. In turn, it made platforms that supported parameterized questions such as LON-CAPA [9] or edX very popular for college-offered online learning and MOOCs.

Due to the complexity of parameterized assessment, the majority of work in this field was done in physics and other math-related domains. However, this technology is becoming more and more popular in the domain of programming. In particular, Brusilovsky and Sosnovsky [1] developed and explored the QuizPACK platform for C-programming and Hsiao et. al explored the similar QuizJET [5] platform for Java programming. In all these domains, the results of many studies have confirmed the value of parameterized questions as e-learning tools [6, 10, 1, 5]. At the same time, Hsiao et al.'s research on parameterized exercises in the programming domain [5] demonstrated that the ability to try the same question again and again is not always beneficial, especially for students who are not good at managing their learning. The analysis of a large number of student logs revealed some considerable number of unproductive repetitions. For example, we can observe many cases where students repeatedly try to correctly solve the same exercise with different parameters (which is easy for them at the repetition) instead of focusing on new, more challenging questions. We can also observe repetitive failed attempts to solve the same exercise for which the students are apparently not ready, instead of focusing on simpler exercises and missing knowledge.

We believe that this unproductive practice could be avoided if a personalized e-learning system featuring parameterized exercises can predict the success of students' future problem-solving attempts in the same way a recommender system can predict, for example, whether a user would or would not like a new movie. The ability to predict student performance in the context of solving parameterized exercises could enable the system to intercept non-productive behavior and recommend a more efficient learning path.

Predicting Student Performance (PSP) is a popular topic in the area of cognitive tutors. In this category of systems each problem or problem-solving step (item) is associated with specific units of knowledge (Knowledge Components or KCs) to be mastered. Observing students' past successes and failures, a cognitive modeling system attempts to model student mastery for each unit of knowledge. The traditional approach for cognitive modeling is Bayesian Knowledge Tracing (BKT) [2], which employs a two-state dynamic Bayesian network estimating the latent cognitive state (student knowledge) from students' performance. More recently Performance Factor Analysis (PFA) [12] emerged as an alternative approach for cognitive modeling and performance prediction. PFA takes into account the effects of the initial difficulty of the KCs and prior successes and failures of a student on the KCs associated with the current item.

In addition, many relevant ideas have been explored in the field of collaborative recommender systems [11]. While collaborative filtering approaches were designed to predict user taste, not user performance, technically they are resolved to predict a score for unknown items based on the past experiences of users. We can consider users of a collaborative filtering system as students, items as skills/questions/steps in solving the problem, and user rating as the predicted value representing the student's success or failure. In recent years, more modern approaches, such as matrix factorization [8] and tensor factorization [7] have been used in recommender systems. There are several works applying factorization techniques to student modeling, such as Thai et. al's three dimensional tensor factorization [13]. However, none of these works are focused on predicting user performance in parameterized questions at the question level nor have they used four-dimensional tensors to model students' behavior.

The problem of PSP in the context of solving parameterized problems, however, is harder than predicting solving regular "solve-once" problems. Traditional modeling approaches both from cognitive tutors and collaborative recommendation areas are not fully adequate for the parameterized problem case since they can't distinguish repeated attempts to solve the same problem from solving a new problem related to the same skills. As a result, we cannot name any attempts to explore approaches for predicting success in solving parameterized exercises. While there are some works focused on performance prediction in classes with parameterized exercises, they focus on a much coarser level of prediction, such as PSP in the whole class [10].

This paper attempts to bridge this gap by exploring a range of techniques for performance prediction in the challenging context of parameterized exercises for Java programming. The main idea of our work is that the quality of PSP in parameterized problems could be increased by using additional data, such as domain knowledge structure, that are not typically considered by traditional approaches. Current prediction approaches in both areas, cognitive tutors and collaborative recommendation, assume that performance of every item should be measured independently and it provides too little data for these approaches to learn from data. We argue that relatively simple knowledge structure such as the information that several items belong to the same broader topic could be very helpful in this context. To explore this idea, we compare advanced log-driven prediction approaches such as Bayesian Knowledge Tracing [2], Performance Factor Analysis [12], advanced collaborative filtering approach (tensor factorization) [7], and FAST (Feature-Aware Student Knowledge Tracing) [3]. Our results show that the latter two approaches that can take into account knowledge structure do produce better performance.

2 The approaches

As we discussed in the introduction section, we believe that using extra information, such as the domain knowledge structure of the items should increase the accuracy of PSP. We experiment on five student modeling approaches in PSP: BKT [2], PFA [12], three dimensional and four dimensional Tensor factorization [7], and FAST [3]. BKT and PFA are the pioneer PSP methods that we choose as our baselines.

For predicting the performance of students without modeling the domain knowledge structure, we choose BKT, PFA. Also, as for the recommender system method for this purpose, we choose three-dimensional tensor factorization that can include the same level of information as BKT and PFA. The three dimensional tensor factorization model that we choose is Bayesian Probabilistic Tensor Factorization (BPTF) [14].

To include domain knowledge structure in PSP, we have to utilize the methods that can incorporate this information in their models. We select FAST as a new approach to this problem that can include both knowledge tracing and other models' information (such as PFA variables) in the form of additional features. Also, we model student behavior in a four-dimensional tensor that can include each of the domain knowledge structure aspects as one new dimension. Tensors are a very good fit for this problem because each type of information can be modeled as a new dimension of a tensor. We extended the tensor factorization model in [14] from a three dimensional tensor factorization to a four dimensional tensor factorization for this purpose.

Each of these methods has their positive and negative aspects; e.g., BKT can model the time sequence of student attempts explicitly, while PFA considers this information as a form of summary of the number of past successes and failures. PFA can handle multiple KCs, while BKT can only consider one KC. Tensor factorization methods predict personalized performance for each student, while BKT and PFA are not personalized in their original form. FAST has the positive characteristics of both BKT and PFA. In the following, we provide a brief description of each of the methods.

Bayesian Knowledge Tracing: The Bayesian Knowledge Tracing [2] model assumes a two-state learning model where each Knowledge Component (skill, or rule) is either in the learned or unlearned state. It uses a simple dynamic Bayesian network where the observable variable represents student performance (correct or incorrect) and the hidden variable represents student knowledge state. There are four parameters in BKT : the initial knowledge parameter ($p(L_0)$) represents the probability that the student knows a KC before practicing on any items associated with the KC; the learning rate parameter ($p(T)$) represents the probability that a student learns a KC by practicing; the guess parameter ($p(G)$) represents the probability when a student doesn't know a KC but answers the item correctly; the slip parameter ($p(S)$) represents the probability when a student knows the KC but answers the item incorrectly.

Performance Factor Analysis: Performance Factor Analysis [12] predicts the student's performance based on the easiness of the current KC(s), the student's prior correct responses and incorrect responses on the KC(s) associated with the current item using a standard logistic regression model. The correctness of response of a student on an item is modeled as the dependent variable here.

Feature-Aware Knowledge Tracing: FAST [3] is a method that combines PFA with Knowledge Tracing. It is able to infer students' knowledge, like BKT does, while allowing for arbitrary features, like PFA. FAST allows general features into Knowledge Tracing by replacing the generative emission probabilities (guess and slip probabilities) with logistic regression, so that these probabilities can change with time to infer a student's knowledge.

Tensor Factorization Methods: A tensor is a multi-dimensional or N-way array. A matrix is a 2-way tensor. Matrix factorization is a popular approach in the recommender systems field. In recommender systems, user ratings on items are represented in a matrix: one dimension of the matrix shows the users and the other dimension shows the items; each element of the matrix represents a user's rating on a specific item. In the educational data mining domain, to predict student performance, we can model users' success or failure on all items in a matrix: if a user succeeds in solving that item, the value for that element will be one and zero otherwise. Since there is only one element per student per question, we should consider only one value for the success or failure of each student on each item. This is problematic when a student has more than one attempt with different results on an item, such as in parameterized exercises. Thus, we should consider a method to incorporate time into the factorization model. To address this problem, we use a three dimensional tensor consisting of students, items, and attempts dimensions. Each element of this tensor represents the success/failure (one/zero) of a student on an item in a specific attempt. The task of predicting user performance here aims to find the success or failure of a student in each attempt of an item. Tensor factorization methods try to decompose a tensor into lower-dimensional space and predict the missing values of the tensor by approximating them using this lower-dimensional representation. In this paper, we use the three dimensional Bayesian probabilistic tensor factorization (3D-BPTF) introduced by Xiong et. al [14] to predict the success or failure of students.

The items considered in one dimension of the three dimensional tensor can be one type of KC, such as questions or topics. As a result, we cannot model various KC types in a three dimensional tensor. To deal with this problem, we represent the data in a four dimensional tensor and extend the 3D-BPTF to a four dimensional tensor factorization (4D-BPTF). In this model, the dimensions are: students, KC type 1, KC type 2, and attempts. For example, in this study we model the questions along with the topics associated with these questions. The tensor model of the data has the students, questions, topics, and attempts dimensions. Note that tensors are able to have as many dimensions as needed to model the data. But adding each dimension will add to the sparsity of the tensor.

3 The Dataset

Our dataset was collected from the online self-assessment system QuizJET [5], which provides parameterized questions for learning Java programming. Each parameterized question is generated from a template filling parameters inside the question with random (and reasonable) values to avoid providing the exact same question to the student. Students can try different versions of the same question multiple times until they acquire the knowledge or give up. The dataset was collected from Fall 2010 to Spring 2013 (six semesters). The subject domain is organized into reasonably coherent topics, each topic has several questions. Each question is assigned to one topic. We experimented on $27,302$ records of 166 students on 103 questions. The average number of attempts on each question is equal to three. Students have at least one attempt to at most 50 attempts in one question. Our dataset is imbalanced: the total number of successful attempts in the data equals to $18,848$ (69.04%) and the total number of failed attempts is 8454. We used user-stratified 5-fold cross-validation to split the data, so that the training set has 80% of the users (with all their records) randomly selected from original dataset, while the remaining 20% of the users were retained for testing. We performed a 5-fold cross-validation to perform the comparison in our studies. We ensured that all questions seen in the test set have at least one student attempt in the training set. In this way, all models are predicting unseen students on observed questions in each run. Simple statistics of our dataset are shown in Table 1.

Table 1: Dataset Statistics

	Average	Min	Max
#attempts per sequence	3	1	50
#attempts per question	265	25	582
#attempts per student	165	2	772
#different students per question	87	7	142
#different questions per student	54	1	101

4 The Studies

Cognitively-based approaches for knowledge prediction expect any success or failure to solve a problem to be attributed to one or more knowledge components that are practiced during problem solving. The most natural choice for this knowledge component is a topic that represents a coarse-grained level of information. With this approach, working with any question belonging to the same topic could be considered a chance to practice this topic. The problem of this approach is that problems belonging to the same topic might still be considerably different. As a result, the success or failure with one of the problems might carry too little evidence to predict performance on other problems in the same topic. An alternative approach is to consider each question to be a distinct subtopic and to use questions as knowledge components for modeling. We expect that including the domain knowledge structure, by having both questions and topic information, results in a better modeling for PSP. Since these approaches have both strong and weak aspects, we explored them in the hopes of finding the one that allows the best quality of prediction.

Each of the approaches models information of the domain (knowledge components) in different ways: BKT models each KC (either question or topic) separately; PFA models items (either questions or topics) with multiple KCs; 3-D tensor factorization considers items (either questions or topics) as KCs; 4-D tensor factorization includes both question and topic information (includes domain knowledge structure); and FAST considers each KC (topic or question) separately, in addition to the extra information (e.g. question information) in the form of features. Because of these characteristics, we expect FAST and 4D-BPTF to perform better than the aforementioned methods while having extra information about the domain knowledge structure. On the other hand, when we consider each question as a KC, and thus we do not have any information about the topics, we expect FAST and 3D-BPTF to perform similarly to other methods. The reason is that, in this case, the regression-based features of FAST contain only redundant information that can be captured by those models naturally. Also, 3D-BPTF does not have the extra dimension for topics. To examine the performance of these approaches and compare them using different information resources, we designed two studies with various granularity: one with question as the knowledge unit, and the other with topic as the knowledge unit (and extra question information for FAST and 4D-BPTF).

4.1 Study 1: Question as Knowledge Unit

The Procedure For BKT, FAST, and PFA, we treat a question (item) as a knowledge component (KC) in this set of experiments. By using question (item) level KCs, we would be able to capture a question's characteristic for predicting different attempts on the same question. To model the tensor, we used the three dimensions of student, question, and attempt.

We used existing tools implementing the above methods to perform our experiments. We used Expectation Maximization (EM) algorithm for BKT and set the initial parameters as follows: $p(L_0) = 0.5$ (for Initial Knowledge), $p(G) = 0.2$ (for Guessing), $p(S) = 0.1$ (for Slipping), $p(T) = 0.3$ (for Learning). For running PFA, we used the implementation of logistic regression in WEKA [4]. For 3D-BPTF, we used the Matlab code prepared by Xiong et. al.[3]. We experimented with different latent space dimensions for 3D-BPTF (5, 10, 20 and 30) and chose the best one, which has the latent space dimension as 10 to compare with other models.

Table 2: Results of the Methods with Question as Unit to Predict Student Performance

Methods	Accuracy	RMSE	TP	TN	FP	FN	Maj. precision	Min. precision	Maj. recall	Min. recall
FAST	73.64(0.8)	0.4209	3063.4	953.6	737.2	706.2	80.61	57.54	81.45	56.15
BKT	74.38(0.8)	0.4152	3527.6	534.8	1156.0	242.0	75.33	68.69	93.43	32.00
PFA	74.69(1.0)	0.4185	3381.4	701.4	989.4	388.2	77.34	64.16	89.56	41.63
3D-BPTF	74.26(0.9)	0.4189	3423.4	636.2	1054.6	346.2	76.42	64.59	90.60	37.88

The Results The results of our experiments are shown in Table 2. We used Accuracy, RMSE (Root Mean Squared Error), TP (True Positive), TN (True Negative), FP (False Positive), FN (False Negative), Maj. (Majority) and Min. (Minority) recall and precision to evaluate the methods. Numbers in parenthesis show the confidence interval with $P < 0.05$. We can see that the accuracy of all models is very close to each other. Among the models, BKT has slightly more true positives and false positives. It means that BKT tends to predict more positive values (successes) for the students. It over estimates the student's performance. Confusion matrix values of PFA and 3D-BPTF are in between FAST and BKT. Contrary to BKT, FAST has more true negatives and false negatives. It means that FAST tends to predict more failures for the students. BKT has the highest minority precision and significantly highest majority recall.

[3] http://www.cs.cmu.edu/ lxiong/bptf/bptf.html

FAST has the highest majority precision and highest minority recall. It means that if FAST predicts a success for a student and if BKT predicts a failure for students, their prediction is more likely to be true compared to the other methods.

4.2 Study 2: Topic as Knowledge Unit

The Procedure In our dataset, each question is related to only one topic, but each topic can include multiple questions. In this study, we treat a topic (item) as a knowledge component (KC) for BKT and PFA. In FAST, each topic is a KC but it includes the question that the topic is related to as an additional feature. For 4D-BPTF, the tensor dimensions are students, topics, questions, and attempts. To perform the methods, we used the same tools as in Study 1, and extended the 3D-BPTF to a four-dimensional tensor factorization (4D-BPTF). To compare the results, we also modeled a three dimensional tensor with students, topics, and attempts (ignoring the question information) and performed 3D-BPTF on that.

Table 3: Results of the Methods with Topic as Unit to Predict Student Performance

Methods	Accuracy	RMSE	TP	TN	FP	FN	Maj. precision	Min. precision	Maj. recall	Min. recall
FAST	75.34(0.7)	0.4134	3360.8	740.4	943.8	402.4	78.08	65.00	89.47	43.63
4D-BPTF	74.40(1.2)	0.4192	3374.8	783.6	917.2	506.4	78.63	60.71	86.76	46.30
BKT	71.93(1.1)	0.4355	3510.8	403.0	1281.2	252.4	73.33	61.68	93.38	23.81
PFA	71.16(0.8)	0.4392	3452.8	419.2	1265	310.4	73.21	57.90	91.95	24.49
3D-BPTF	69.66(1.3)	0.8317	3758	37	1647.2	5.2	69.16	91.06	100	2.13

The Results The results of our experiments are shown in Table 3. Numbers in parenthesis show the confidence interval with $P < 0.05$. As we can see, FAST and 4D-BPTF's performance do not have a significant difference from each other. Also, FAST and 4D-BPTF perform significantly better than all other approaches in terms of accuracy and RMSE. This is because of their ability to model and capture both topic-level and question-level information. While FAST has the topics as KCs, it is using question features, such as the number of successes and failures, in its model. In addition, 4D-BPTF models topics and questions explicitly as different dimensions of one tensor. Compared to the first study's results, FAST performs better while using topic-level KCs compared to using question-level KCs (Study 1). Also, 4D-BPTF performs better than 3D-BPTF both using question-level information (Study 1) and topic-level information (Study 2). BKT and PFA perform similarly to their results in Study 1 and 3D-BPTF on topics is slightly weaker than 3D-BPTF on questions in terms of accuracy. 3D-BPTF has a very high minority precision and majority recall. It means that it over estimates the students' ability to solve the problems. Eventually, we can conclude that having information on both questions and topics (in various granularity) can significantly improve the accuracy of predicting students performance in parameterized questions.

5 Discussion

As we have seen in the results presented above, each method's accuracy in predicting students performance depends on the input of the method. When we ignore the topic of questions as KCs, all models perform similarly. On the other hand, when we include topic information, in addition to the question information, in the models, the methods that can leverage this extra information perform much better than the ones that cannot consider that information. FAST and 4D-BPTF perform the best because they can include both question-level and topic-level information in addition to the time sequence of students. BKT and PFA cannot include the question-level information, but they still have the time sequence of attempts or the number of successes and failures. 3D-BPTF does not have any information for the questions and it cannot distinguish between the topics and questions.

Also, we can see that adding the extra topic data in the methods that cannot model this information well decreases the method's accuracy. FAST is the only model that performs better, compared to itself, in Study 2 versus Study 1. Also, 4D-BPTF performs better than 3D-BPTF in both question-level and topic-level models. BKT and PFA perform worse in Study 2 compared to Study 1.

We can conclude that, in the case of predicting students performance on multiple attempts, adding to the granularity of information is useful if the method we are using can model this extra information explicitly. Otherwise, this extra information may harm the accuracy of the results.

6 Conclusion and Future Work

In this study, we explored several advanced student modeling approaches in predicting student performance in solving parameterized exercises, particularly in the programming area. This is the first work that models students' data as a four-dimensional tensor (in 4D-BPTF).

We performed two sets of studies with different granularity using these methods: one considering questions as Knowledge Components (KCs) and the other one considering topics as KCs. We discovered that FAST and 4D-BPTF performs best in the second study. Also, we saw that the models do not differ significantly in the first study. We can conclude that, in the case of predicting students performance on parameterized exercises, adding to the granularity of information can be useful for the methods that can model and leverage this extra information explicitly. The methods that could not include both question-level information and topic-level KCs at the same time are not suitable for this granularity of data and did not perform well in the second study.

In addition, the success of using tensor factorization, which is one of the advanced techniques in the recommendation area, in both studies encourages more research on applying more recommendation techniques in PSP. Giving that factorization techniques do not need to know the exact Knowledge Components that influence students' performance, they reduce the manual effort in exercising authoring for student modeling, which is promising for providing student modeling in a larger scale.

Our first effort in this work in treating a question (item) as a KC and our second study in having a topic as a KC, in addition to question information, for FAST and 4D-BPTF proved effective. However, we haven't explored whether using more coarse-grained or fine-grained level KCs would give better prediction performance. Particularly, since PFA is designed for modeling multiple KCs, we need further experiments to compare these models when each item is associated with multiple KCs.

References

1. P. Brusilovsky and S. Sosnovsky. Individualized exercises for self-assessment of programming knowledge: An evaluation of quizpack. *ACM Journal on Educational Resources in Computing*, 5(3):Article No. 6, 2005.
2. A. T. Corbett and J. R. Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User modeling and user-adapted interaction*, 4(4):253–278, 1994.
3. J. P. González-Brenes, Y. Huang, and P. Brusilovsky. Fast: Feature-aware student knowledge tracing. In *NIPS Workshop on Data Driven Education*, 2013.
4. M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.
5. I.-H. Hsiao, S. Sosnovsky, and P. Brusilovsky. Adaptive navigation support for parameterized questions in object-oriented programming. In *ECTEL 2009*, volume 5794 of *LNCS*, pages 88–98. Springer-Verlag.
6. E. Kashy, M. Thoennessen, Y. Tsai, N. E. Davis, and S. L. Wolfe. Using networked tools to enhanse student success rates in large classes. In *FIE*, volume I, pages 233–237. Stipes Publishing L.L.C., 1997.
7. T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM*, 51(3):455–500, 2009.
8. Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.
9. G. Kortemeyer, E. Kashy, W. Benenson, and W. Bauer. Experiences using the open-source learning content management and assessment system lon-capa in introductory physics courses. *American Journal of Physics*, 76(438), 2008.
10. B. Minaei-Bidgoli, D. A. Kashy, G. Kortemeyer, and W. F. Punch. Predicting student performance: an application of data mining methods with an educational web-based system. In *FIE 2003*.
11. D. Parra and S. Sahebi. Recommender systems: Sources of knowledge and evaluation metrics. In J. V. et al. (Eds.), editor, *Advanced Techniques in Web Intelligence-2*, chapter 7, pages 149–175. Springer-Verlag, Berlin Heidelberg, 2013.
12. P. I. Pavlik, H. Cen, and K. R. Koedinger. Performance factors analysis-a new alternative to knowledge tracing. In *AIEd*, pages 531–538, 2009.
13. N. Thai-Nghe, T. Horvath, and L. Schmidt-Thieme. Context-aware factorization for personalized student's task recommendation. In *Int. Workshop on Personalization Approaches in Learning Environments*, volume 732, pages 13–18, 2011.
14. L. Xiong, X. Chen, T.-K. Huang, J. G. Schneider, and J. G. Carbonell. Temporal collaborative filtering with bayesian probabilistic tensor factorization. In *SDM*, volume 10, pages 211–222, 2010.