Predicting Student Performance in Solving Parameterized Exercises

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

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

Abstract. In this paper, we compare pioneer methods of educational data mining field with recommender systems techniques for predicting student performance. Additionally, we study the importance of including students' attempt time sequences of parameterized exercises. The approaches we use are Bayesian Knowledge Tracing (BKT), Performance Factor Analysis (PFA), Bayesian Probabilistic Tensor Factorization (BPTF), and Bayesian Probabilistic Matrix Factorization (BPMF). The last two approaches are from the recommender system's field. We approach the problem using question-level Knowledge Components (KCs) and test the methods using cross-validation. In this work, we focus on predicting students' performance in parameterized exercises. Our experiments shows that advanced recommender system techniques are as accurate as the pioneer methods in predicting student performance. Also, our studies show the importance of considering time sequence of students' attempts to achieve the desirable accuracy.

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. As a result, a single question's template is able to produce a large number of different questions. One of the benefits of this technology is in the self-assessment context: the same question can be used again and again with different parameters. This allows every student to achieve understanding and mastery.

On the practical side, this property and other benefits, such as re-usability and being cheating-proof, made parameterized exercises very attractive for the large-scale online learning context. In turn, it made platforms that supported parameterized questions such as LON-CAPA [8] or edX very popular for college-offered online learning and MOOCs.

On the research side, a range of studies have confirmed the value of parameterized questions as e-learning tools [5,9,1,4]. At the same time, Hsiao et. al's [4] experience with parameterized questions in the self-assessment context demonstrated that the important ability to try the same question again and again is not always beneficial, especially for students who are not good in 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 and correctly solve the same exercise with different parameters (which is at the time apparently easy for them) 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 as a recommender system can predict, for example, whether a user would or would not like a new movie. The ability to predict students' performance in the context of solving parameterized exercises could enable the system to intercept non-productive behavior and recommend a more efficient learning path. We also believe that the presence of a large volume of learning data that is now collected in online learning systems makes the task of performance prediction possible. In addition, we beleive that the repetition of exercises makes the attempt sequencing of students' activities an important factor to predict their performance. As a result, we expect the time-aware (or sequence-aware) approaches to perform better in this context. However, so far, there have been no attempts to explore approaches for predicting success in solving parameterized exercises. This paper attempts to bridge this gap by exploring a range of techniques for performance prediction. We compare advanced log-driven time-aware prediction approaches such as Bayesian Knowledge Tracing [2], Performance Factor Analysis [11], and tensor factorization (as an advanced collaborative filtering approaches [6]) with matrix factorization (as a baseline approach that does not model attempt sequences).

2 Background: Predicting Student Performance

The traditional approach to predict user experience with unknown items using the past experience of the user, along with a large community of other users, was developed in the field of collaborative recommender systems [10]. While collaborative filtering approaches were designed to predict user taste, not user performance, technically it is resolved to predicting 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 student's success/failure. In recent years, more modern approaches, such as matrix factorization [7] and tensor factorization [6] have been used in recommender systems. There are several works applying factorization techniques to student modeling, such as Thai et. al's tensor factorization [12]. But none of these works are focused on predicting user performance in parameterized questions at the question level.

Another approach for Predicting Student Performance (PSP) in problem solving is based on the idea of cognitive modeling. With cognitive modeling, 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) [11] has emerged as a powerful 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.

The problem of PSP in the context of solving parameterized problems is somewhat harder than predicting solving regular "solve-once" problems. Traditional modeling approaches are not fully adequate for 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. 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 [9].

3 The Approaches

As we stated in the introduction section, we expect the time-aware approaches perform better than time-ignorant approaches in PSP for parameterized exercises. Also, we expect advanced recommender systems approaches to perform as good as the pioneer methods in PSP. To study these expectations, we experiment on four student modeling approaches: BKT [2], PFA [11], Tensor Factorization [6], and Matrix Factorization [13]. As the previous work in PSP is focused on knowledge tracing and regression models, we choose a method of each: BKT and PFA. As for approaches of recommender systems, we choose a tensor factorization method that can include students' attempt sequence and a matrix factorization method. Each of these methods has their positive and negative aspects; e.g. BKT can model the time sequence of student attempts while PFA cannot model that explicitly; PFA can handle multiple knowledge components while BKT can only model one KC; and tensor factorization and matrix factorization methods predict a personalized performance for each student. We choose a Max baseline in addition to the above methods. This baseline predicts success (the majority class) for every attempt. Using this baseline, we explore how our models preform given our imbalanced data. 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 [11] predicts student's performance based on the easiness of the current Knowledge Component(s), 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. PFA does not model time sequences directly, but it considers them as the number of past successes and failures.

Matrix/Tensor Factorization: Matrix factorization is a popular approach in the recommender systems field. In the educational data mining domain, to predict student performance, we can model a user's attempt on all of the items as a one-dimensional binary array of length q (number of items). If a user succeeds in solving that item, the value for that element will be one and zero otherwise. Considering all of the items and all of the students, we can model all students' success or failure on all questions using an $s \times q$-matrix. Since different students might have different number of attempts on various items, we consider only the success or failure of the last attempt of the student. Some of the values of this matrix are unknown to us because some students might have never tried an item. The task of predicting user performance aims to find the values of these unknown elements of the matrix. In this paper, we use a Bayesian probabilistic matrix factorization (BPMF) method [13] to predict the success or failure of students in various questions.

However, a student might have more than one attempt with different results on an item. Thus, we should consider a method to incorporate time into the factorization model. One way of doing so is to use tensors. A tensor is a multi-dimensional or N-way array. A matrix is a 2-way tensor. In our problem, the sequence of one user's attempt on one item can be seen as a t-dimensional array consisting of zeros and ones. Zeros are representative of student's failure in that particular attempt of the item and ones are indicative of success. Consequently, if we want to model all the attempts each student has made on each item, we will end up with a three-dimensional tensor of the size $s \times q \times t$, which has binary values of failure or success. 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 Bayesian probabilistic tensor factorization (BPTF) introduced by Xiong et. al [13] to predict the success or failure of students.

4 The Dataset

Our dataset was collected from the online self-assessment system QuizJET [4], 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 of the 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 are 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

5 The Experiment

Our approach is to consider each question to be a distinct subtopic and use questions as knowledge components for modeling. Among the methods discussed above, BPTF, BKT, and PFA each consider the student's attempt sequence in a way: BKT models it explicitly as an HMM, BPTF has a smooth changing condition for students' attempts and PFA summarizes this information in the number of previous successful and unsuccessful attempts. On the contrary, BPMF does not consider this information. To examine the performance of these approaches and compare them using different information resources, we design the following experiment.

The Procedure. 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 use the three dimensions of student, question, and attempt. Each element of the tensor shows the success (1) or failure (0) of student in that question for the specific attempt. To model the matrix, we use the two dimensions of student and question. Each element of the matrix shows the success (1) or failure (0) of student in the last available attempt of that question.

We use existing tools implementing the above methods to perform our experiments. We use EM algorithm for BKT and set the initial parameters as follows: $p(L_0) = 0.5$, $p(G) = 0.2$, $p(S) = 0.1$, $p(T) = 0.3$. For running PFA, we use the implementation of logistic regression in WEKA [3]. For BPTF and BPMF, we utilize the Matlab code prepared by Xiong et. al.[1]. We experimented with different latent space dimensions for BPTF and BPMF (5, 10, 20 and 30) and chose the best one, which has the latent space dimension of 10.

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
BKT	74.38(0.8)	0.4152	3527.6	534.8	1156.0	242.0	75.33	68.69	93.43(0.9)	32.00
PFA	74.69(1.0)	0.4185	3381.4	701.4	989.4	388.2	77.34	64.16	89.56(1.1)	41.63
BPTF	74.26(0.9)	0.4189	3423.4	636.2	1054.6	346.2	76.42	64.59	90.60(1.4)	37.88
BPMF	71.73(0.5)	0.4365	3386.4	531	1159.8	383.2	74.39	58.46	89.95(1.6)	31.21
Max	69.04	0.5564	3769.9	0	1690.5	0	0.6904	0	100	0

The Results. The results of our experiments are shown in Table 2. Numbers in parenthesis show the confidence interval with $P < 0.05$. We can see that the accuracy of all models, except BPMF, are very close to each other. The BPMF model lacks the time sequencing of student attempts and performs poorly compared to the other three methods. All methods beat the Max baseline's accuracy. 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. BPTF has the next most true positives and false positives. PFA has more true negatives and

[1] `http://www.cs.cmu.edu/~lxiong/bptf/bptf.html`

false negatives than other approaches. It means that PFA tends to predict more failures for the students. BKT has the highest minority precision and significantly highest majority recall. PFA has the highest majority precision and highest minority recall. It means that if PFA 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.

6 Conclusion and Future Work

In this study, we explored several student modeling approaches in predicting student performance in solving parameterized exercises, particularly in the programming area. All the models we studied (BKT, PFA, BPTF, and BPMF) outperformed the max baseline, showing the feasibility of applying these student models to parameterized exercising systems which are different from the traditional step-by-step, fine-grained designed tutoring systems.

In our experiment, we saw that the sequencing information is an important factor in PSP for parameterized exercises and time-aware models perform better than the time-ignorant matrix factorization method. These time-aware methods do not differ significantly in results of PSP. This result encourages us to seek for more advanced approaches in this area, as future work.

In addition, the success of using BPTF, which is one of the advanced matrix factorization techniques in the recommendation area, 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 effort in this work in treating a question (item) as a KC for BKT and PFA. 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.

Also, since our study uses user-stratified cross-validation, which requires models to predict for new students, BPTF and BPMF encounter an unfavorable situation, since it is hard to give highly accurate prediction for the student with little or no information for that specific student. We will further explore these models' performance giving different amount of information of students or questions that the model is predicting for.

Obtaining reasonable accuracy of predicting performance for parameterized questions is necessary to investigate how to give recommendations to help students: whether to keep practicing on the current question or to move to another suitable question or to learn from reading an example.

References

1. Brusilovsky, P., Sosnovsky, S.: 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. Corbett, A.T., Anderson, J.R.: Knowledge tracing: Modeling the acquisition of procedural knowledge. User Modeling and User-Adapted Interaction 4(4), 253–278 (1994)
3. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The weka data mining software: An update. ACM SIGKDD Explorations Newsletter 11(1), 10–18 (2009)
4. Hsiao, I.-H., Sosnovsky, S., Brusilovsky, P.: Adaptive navigation support for parameterized questions in object-oriented programming. In: Cress, U., Dimitrova, V., Specht, M. (eds.) EC-TEL 2009. LNCS, vol. 5794, pp. 88–98. Springer, Heidelberg (2009)
5. Kashy, E., Thoennessen, M., Tsai, Y., Davis, N.E., Wolfe, S.L.: Using networked tools to enhanse student success rates in large classes. In: FIE, vol. I, pp. 233–237. Stipes Publishing L.L.C., (1997)
6. Kolda, T.G., Bader, B.W.: Tensor decompositions and applications. SIAM 51(3), 455–500 (2009)
7. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer 42(8), 30–37 (2009)
8. Kortemeyer, G., Kashy, E., Benenson, W., Bauer, W.: Experiences using the open-source learning content management and assessment system lon-capa in introductory physics courses. American Journal of Physics 76(438) (2008)
9. Minaei-Bidgoli, B., Kashy, D.A., Kortemeyer, G., Punch, W.F.: Predicting student performance: An application of data mining methods with an educational web-based system. In: FIE 2003 (2003)
10. Parra, D., Sahebi, S.: Recommender systems: Sources of knowledge and evaluation metrics. In: Velásquez, J.D., Palade, V., Jain, L.C. (eds.) Advanced Techniques in Web Intelligence-2. SCI, vol. 452, pp. 149–175. Springer, Heidelberg (2013)
11. Pavlik, P.I., Cen, H., Koedinger, K.R.: Performance factors analysis-a new alternative to knowledge tracing. In: AIEd, pp. 531–538 (2009)
12. Thai-Nghe, N., Horvath, T., Schmidt-Thieme, L.: Context-aware factorization for personalized student's task recommendation. In: Int. Workshop on Personalization Approaches in Learning Environments, vol. 732, pp. 13–18 (2011)
13. Xiong, L., Chen, X., Huang, T.-K., Schneider, J.G., Carbonell, J.G.: Temporal collaborative filtering with bayesian probabilistic tensor factorization. In: SDM, vol. 10, pp. 211–222 (2010)