MATRIX AND TENSOR FACTORIZATION FOR PREDICTING STUDENT PERFORMANCE

Nguyen Thai-Nghe, Lucas Drumond, Tomáš Horváth,
Alexandros Nanopoulos, and Lars Schmidt-Thieme
Machine Learning Lab, University of Hildesheim, 31141 Hildesheim, Germany
{*nguyen, ldrumond, horvath, nanopoulos, schmidt-thieme*}*@ismll.uni-hildesheim.de*

Keywords: Recommender Systems, Matrix Factorization, Tensor Factorization, Student Performance

Abstract: Recommender systems are widely used in many areas, especially in e-commerce. Recently, they are also applied in technology enhanced learning such as recommending resources (e.g. papers, books,...) to the learners (students). In this study, we propose using state-of-the-art recommender system techniques for predicting student performance. We introduce and formulate the problem of predicting student performance in the context of recommender systems. We present the matrix factorization methods, known as the most effective recommendation approaches, to implicitly take into account the latent factors, e.g. "slip" and "guess", in predicting student performance. Moreover, the knowledge of the learners has been improved over the time, thus, we propose tensor factorization methods to take the temporal effect into account. Experimental results show that the proposed approaches can improve the prediction results.

1 INTRODUCTION

Recommender systems are widely used in many areas, especially in e-commerce (Rendle et al., 2010; Koren et al., 2009). Recently, researchers have applied recommendation techniques in e-learning, especially technology enhanced learning (Manouselis et al., 2010). Most of the works focused on constructing recommender systems for recommending learning objects (materials/resources) or learning activities to the learners (Ghauth and Abdullah, 2010; Manouselis et al., 2010) in both formal and informal learning environment (Drachsler et al., 2009).

On the other side, educational data mining has also been taken into account recently to assist universities, teachers, and students. For example, to help the students improve their performance, we would like to know how the students learn (e.g. generally or narrowly), how quickly or slowly they adapt to new problems or if it is possible to infer the knowledge requirements to solve the problems directly from student performance data (Feng et al., 2009). In (Cen et al., 2006) it has been shown that an improved model for predicting student performance could save millions of hours of students' time (and effort) in learning algebra. In that time, students could move to other specific fields of their study or doing other things they enjoy. Moreover, many universities are extremely focused on assessment, thus, the pressure on "teaching and learning for examinations" leads to a significant amount of time spending for preparing and taking standardized tests. Any advances which allow us to reduce the role of standardized tests hold the promise of increasing deep learning (Feng et al., 2009). From an educational data mining point of view, a good model which accurately predicts student performance could replace some current standardized tests.

To address the student performance prediction problem, many works have been published. Most of them relying on traditional methods such as logistic regression (Cen et al., 2006), linear regression (Feng et al., 2009), decision tree (Thai-Nghe et al., 2007), neural networks (Romero et al., 2008), support vector machines (Thai-Nghe et al., 2009), and so on.

Recently, (Thai-Nghe et al., 2010) have proposed using recommendation techniques, especially matrix factorization, for predicting student performance. The authors have shown that using recommendation techniques could improve prediction results compared to regression methods but they have not taken the tempo-

ral effect into account. Obviously, in the educational point of view, we always expect that the students can improve their knowledge time by time, so the temporal information is critical for such prediction tasks.

On the other hand, in student performance prediction, there are two "crucial aspects" need to be taken into account, which are

1. the probabilities that the students do not know how to solve the problem (or do not know some required skills related to the problem) but guessing correctly (we call **"guess"** for short); and the probabilities that the students know how to solve the problem (or know all of the required skills related to the problem) but they make a mistake (we call **"slip"** for short); these problems are user-dependent, and,
2. their knowledge has been improved over the time, e.g. the second time a student is doing his exercises, his performance on average gets better, therefore, the sequence effect is important information.

Matrix factorization techniques, one of the most successful methods for rating prediction, are quite appropriate for the first problem since they implicitly encode the "slip" and "guess" factors in their latent factors and we do not need to explicitly take care of as in case of other methods like Hidden Markov Models (Pardos and Heffernan, 2010). Moreover, if we would like to incorporate the sequential (time) aspect or any other context such as skills in the second "crucial aspect", then tensor factorization techniques are suitable for solving this problem.

This work proposes the factorization approaches for predicting student performance. Concretely, the contributions of this work are:

- formulating the problem of predicting student performance in the context of recommender systems;
- proposing matrix factorization as well as biased matrix factorization techniques which implicitly encode the "slip" and "guess" factors in predicting student performance
- proposing tensor factorization techniques to take the temporal effect into account, e.g. the knowledge of the learners improves over time.
- comparing the proposed approaches with other baselines as well as traditional techniques such as logistic regression.

From the experimental results on two large data sets collected from the intelligent tutoring system, we show that the proposed approaches perform nicely among the other methods.

2 RELATED WORK

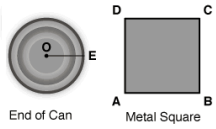
As surveyed in (Manouselis et al., 2010), many recommender systems have been deployed in technology enhanced learning. Concretely, (García et al., 2009) uses association rule mining to discover interesting information through student performance data in the form of IF-THEN rules, then generating the recommendations based on those rules; (Bobadilla et al., 2009) proposed an equation for collaborative filtering which incorporated the test score from the learners into the item prediction function; (Ge et al., 2006) combined the content-based filtering and collaborative filtering to personalize the recommendations for a courseware selection module; (Soonthornphisaj et al., 2006) applied collaborative filtering to predict the most suitable documents for the learners; while (Khribi et al., 2008) employed web mining techniques with content-based and collaborative filtering to compute the relevant links for recommending to the learners.

For predicting student performance, (Romero et al., 2008) compared different data mining methods and techniques to classify students based on their Moodle usage data and the final marks obtained in their respective courses; (Bekele and Menzel, 2005) used Bayesian networks to predict student results; (Cen et al., 2006) proposed a method for improving a cognitive model, which is a set of rules/skills encoded in intelligent tutors to model how students solve problems, using logistic regression; (Thai-Nghe et al., 2007) analyzed and compared some classification methods (e.g. decision trees and Bayesian networks) for predicting academic performance; while (Thai-Nghe et al., 2009) proposed to improve the student performance prediction by dealing with the class imbalance problem. (i.e., the ratio between passing and failing students is usually skewed). Recently, (Thai-Nghe et al., 2010; Toscher and Jahrer, 2010) proposed using collaborative filtering, especially matrix factorization for predicting student performance but they have not take the temporal effect into account.

Different from the literature, instead of using traditional classification or regression methods, we propose using matrix factorization technique to implicitly manage the "slip" and "guess" factors in predicting student performance, and using tensor factorization to incorporate the temporal effect into the models.

3 PREDICTING STUDENT PERFORMANCE

In the problem of predicting student performance, we would like to predict the student's ability in solving the tasks when interacting with the tutoring system. Figure 1a presents an example of the task[1]. Given the circle and the square as in this figure, the task for students could be *"What is the remaining area of the square after removing the circular area?"*



a. A task example

Task Skill

Row	Student	Problem	Step	Knowledge component	Opportunity Count	Correct at First Attempt	
1	S01	WATERING_VEGGIES	(WATERED-AREA Q1)	Circle-Area	1	1	Past performance
2	S01	WATERING_VEGGIES	(TOTAL-GARDEN Q1)	Rectangle-Area	1	1	
3	S01	WATERING_VEGGIES	(UNWATERED-ARE Q1)	Compose-Areas	1	1	
4	S01	WATERING_VEGGIES	DONE	Determine-Done	1	0	
5	S01	MAKING-CANS	(POG-RADIUS Q1)	Enter-Given	1	?	Predict
6	S01	MAKING-CANS	(SQUARE-BASE Q1	Enter-Given	2	?	
7	S01	MAKING-CANS	(SQUARE-AREA Q1	Square-Area	1	?	
8	S01	MAKING-CANS	(POG-AREA Q1)	Circle-Area	2	?	

b. A snaphot of the transaction log

Figure 1: Predicting student performance: A scenario (Picture source: pslcdatashop.web.cmu.edu/KDDCup)

To solve this task (question), students could do some smaller subtasks which we call solving-step. Each step may be required one or more skills (or we can call "knowledge components"), for example:

- Step 1: Calculate the circle area (the required skills for this step are the value of π, square, multiplication, and finally putting them together $area_1 = \pi * (OE)^2$)
- Step 2: Calculate the square area (skill: $area_2 = (AB)^2$)
- Step 3: Calculate the remaining (skill: $area_2 - area_1$)

Each solving-step is recorded as a transaction. Figure 1b presents a snapshot of the transactions. Based on the past performance, we would like to predict students' next performance (e.g. correct/incorrect) in solving the tasks. The following section will formally formulate this problem.

Problem Formulation

Computer-aided tutoring systems (CATS) allow students to solve some exercises with a graphical frontend that can automate some tedious tasks, provide some hints and provide feedback to the student (Massey et al., 1988). Such systems can profit from anticipating student performance in many ways, e.g., in selecting the right mix of exercises, choosing an appropriate level of difficulty and deciding about possible interventions such as hints. The problem of student performance prediction in CATS means to predict the likely performance of a student for some exercises (or part thereof such as for some particular steps) which we call the *tasks*. The task could be to solve a particular step in a *problem*, to solve a whole problem or to solve problems in a *section* or *unit*, etc.

CATS allow to collect a rich amount of information about how a student interacts with the tutoring system and about his past successes and failures. Usually, such information is collected in a clickstream log with an entry for every action the student takes. The clickstream log contains information about the

$$time, student, context, action$$

For performance prediction, such click streams can be aggregated to the task for which the performance should be predicted and eventually be enriched with additional information. For example, if the aim is to predict the performance for each single step in a problem, then all actions in the clickstream log belonging to the same student and problem step will be aggregated to a single transaction and enriched, for example with some performance metrics.

Part of the context describes the task the student should solve. In CATS tasks usually are described in two different ways: First, tasks are located in a topic hierarchy, for example

$$unit \supseteq section \supseteq problem \supseteq step$$

Second, tasks are described by additional metadata such as the skills that are required to solve the problem:

$$skill_1, skill_2, \ldots, skill_n$$

All this information, the topic hierarchy, skills and other task metadata can be described as attributes of the tasks. In the same way, also attributes about the student may be available.

More formally, let S be a set of student IDs, T be a set of task IDs, and $P \subseteq \mathbb{R}$ be a range of possible performance scores. Let M_S be a set of student metadata descriptions and

$$m_S : S \to M_S$$

be the metadata for each student. Let M_T be a set of task metadata descriptions and

$$m_T : T \to M_T$$

[1] Source: https://pslcdatashop.web.cmu.edu/KDDCup

be the metadata for each task. Finally, let $D^{train} \subseteq (S \times T \times P)^*$ be a sequence of observed student performances and $D^{test} \subseteq (S \times T \times P)^*$ be a sequence of unobserved student performances. Furthermore, let

$$\pi_p : S \times T \times P \to P, \quad (s,t,p) \mapsto p$$

and

$$\pi_{s,t} : S \times T \times P \to S \times T, \quad (s,t,p) \mapsto (s,t)$$

be the projections to the performance measure and to the student/task pair.

Then the problem of student performance prediction is, given D^{train}, $\pi_{s,t}D^{test}$, m_S, and m_T to find

$$\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_{|D^{test}|}$$

such that

$$err(p, \hat{p}) := \sum_{i=1}^{|D^{test}|} (p_i - \hat{p}_i)^2$$

is minimal with $p := \pi_p D^{test}$ (or some other error measures). In principle, this is a regression or classification problem (depending on the error measure).

Given $s \in S$ and $t \in T$, our problem is to predict p. Obviously, in a recommender system context, s,t and p would be *user, item* and *rating*, respectively. The recommendation task at hand is thus rating prediction.

4 TECHNIQUES FOR RECOMMENDER SYSTEMS

The aim of recommender system is making vast catalogs of products consumable by learning user preferences and applying them to items formerly unknown to the user, thus being able to recommend what has a high likelihood of being interesting to the target user. The two most common tasks in recommender systems are Top-N item recommendation where the recommender suggests a ranked list of (at most) N items $i \in I$ to a user $u \in U$ and rating prediction where the aim is predicting the preference score (rating) $r \in \mathbb{R}$ for a given user-item combination.

In this work, we make use of matrix factorization (Rendle and Schmidt-Thieme, 2008; Koren et al., 2009), which is known to be one of the most successful methods for rating prediction, outperforming other state-of-the-art methods (Bell and Koren, 2007) and tensor factorization (Kolda and Bader, 2009; Dunlavy et al., 2010) to take into account the sequential effect. We will briefly describe these techniques in the following subsections.

Notation: A matrix is denoted by a capital italic letter, e.g. X; A tensor is denoted by a capital bold letter, e.g. $\mathbf{Z}$; w_k denotes a k^{th} vector; w_{uk} is the u^{th} element of a k^{th} vector; and $\circ$ denotes an outer product.

4.1 Matrix Factorization

Matrix factorization is the task of approximating a matrix X by the product of two smaller matrices W and H, i.e. $X \approx WH^T$ (Koren et al., 2009). In the context of recommender systems the matrix X is the partially observed ratings matrix, $W \in \mathbb{R}^{U \times K}$ is a matrix where each row u is a vector containing the K latent factors describing the user u and $H \in \mathbb{R}^{I \times K}$ is a matrix where each row i is a vector containing the K factors describing the item i. Let w_{uk} and h_{ik} be the elements of W and H, respectively, then the rating given by a user u to an item i is predicted by:

$$\hat{r}_{ui} = \sum_{k=1}^{K} w_{uk} h_{ik} = (WH^T)_{u,i} \tag{1}$$

where W and H are the model parameters and can be learned by optimizing the objective function (2) given a criterion such as root mean squared error (RMSE):

$$\min_{W,H} (r_{ui} - \hat{r}_{ui})^2 + \lambda(||W||^2 + ||H||^2) \tag{2}$$

where λ is a regularization term which is used to prevent overfitting. In this work, the model parameters were optimized for RMSE using stochastic gradient descent (Bottou, 2004):

$$RMSE = \sqrt{\frac{\sum_{ui \in D^{test}} (r_{ui} - \hat{r}_{ui})^2}{|D^{test}|}} \tag{3}$$

An illustration of matrix decomposition is presented in Figure 2

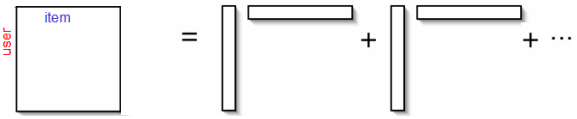


Figure 2: An illustration of matrix factorization

4.2 Tensor Factorization

This is a general form of matrix factorization. Given a three-mode tensor $\mathbf{Z}$ of size $U \times I \times T$, where the first and the second mode describe the user and item as in previous section; the third mode describes the context, e.g. time, with size T. Then $\mathbf{Z}$ can be written as a sum of rank-1 tensors:

$$\mathbf{Z} \approx \sum_{k=1}^{K} w_k \circ h_k \circ q_k \tag{4}$$

where $\circ$ is the outer product and each vector $w_k \in \mathbb{R}^U$, $h_k \in \mathbb{R}^I$, and $q_k \in \mathbb{R}^T$ describes the latent factors of

user, item, and time, respectively (please refer to the articles (Kolda and Bader, 2009; Dunlavy et al., 2010) for more details). An illustration of tensor decomposition is presented in Figure 3. The model parameters were also optimized for RMSE using stochastic gradient descent.

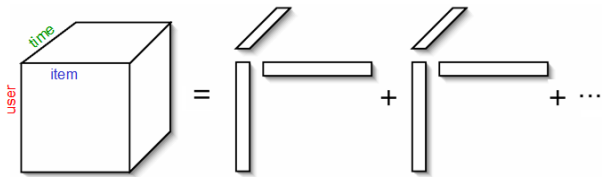


Figure 3: An illutration of tensor factorization

Student ID	Problem Hierarchy	Problem Name	Problem View	Step Name	Knowledge Components	Correct First Attempt	...
stu_de277734(	Unit CTA1_01, Section CTA	L1FB12	1	R1C1		1	...
stu_de277734(	Unit CTA1_01, Section CTA	L1FB12	1	R1C2		0	
stu_de277734(	Unit CTA1_01, Section CTA	L1FB12	1	R2C1		1	
stu_de277734(	Unit CTA1_01, Section CTA	L1FB12	1	R2C2	Identifying units	1	
stu_de277734(	Unit CTA1_01, Section CTA	L1FB12	1	R3C1	Define Variable	1	
stu_de277734(	Unit CTA1_01, Section CTA	L1FB12	1	R3C2	Write expressior	1	
stu_de277734(	Unit UNIT-CONVERSIONS-	UNITCONVE	1	MoreOrF	Compare units	0	
stu_de277734(	Unit UNIT-CONVERSIONS-	UNITCONVE	1	Conversi	Enter unit convei	1	
stu_de277734(	Unit UNIT-CONVERSIONS-	UNITCONVE	1	SelectFra	Select form of orl	1	
stu_de277734(	Unit UNIT-CONVERSIONS-	UNITCONVE	1	Numerat	Enter numerator:	1	
stu_de277734(	Unit UNIT-CONVERSIONS-	UNITCONVE	1	Denomir	Enter denominat	1	...
...	...						...

Figure 4: A snapshot of the data sets

5 DATA SETS AND METHODS

Two real-world data sets are collected from the Knowledge Discovery and Data Mining Challenge 2010[2]. These data sets, originally labeled "Algebra 2008-2009" and "Bridge to Algebra 2008-2009" will be denoted "Algebra" and "Bridge" for the remainder of our paper. Each data set is split into a train and a test partition as described in Table 1. The data represents the log files of interactions between students and computer-aided-tutoring systems. While students solve math related problems in the tutoring system, their activities, success and progress indicators are logged as individual rows in the data sets. A snapshot of the data sets is described in Figure 4.

Table 1: Data sets

Data sets	Size	#Attr.	#Records
Algebra train	3.1 Gb	23	8,918,054
Algebra test	124 Mb	23	508,912
Bridge train	5.5 Gb	21	20,012,498
Bridge test	135 Mb	21	756,386

The central element of interaction between the students and the tutoring system is the *problem*. Every problem belongs into a hierarchy of *unit* and *section*. Furthermore, a problem consists of many individual *steps* such as calculating a circle's area, solving a given equation, entering the result and alike. The field *problem view* tracks how many times the student already saw this problem. Additionally, a different number of *knowledge components* (KC) and associated *opportunity counts* is provided. Knowledge components represent specific skills used for solving the problem (where available) and opportunity counts encode the number of times the respective knowledge component has been encountered before. Both, knowledge components and opportunity counts are represented in a denormalized way, featuring all knowledge components and their counts in one column.

Target of the prediction task is the *correct first attempt* (CFA) information which encodes whether the student successfully completed the given step on the first attempt (CFA = 1 indicates correct, and CFA = 0 indicates incorrect). The prediction would then encode the certainty that the student will succeed on the first try.

5.1 Mapping Educational Data to Recommender Systems

There is an obvious mapping of users and ratings in the student performance prediction problem:

$$\textit{student} \mapsto \textit{user}$$
$$\textit{performance (correct first attempt)} \mapsto \textit{rating}$$

The student becomes the *user*, and the correct first attempt (CFA) indicator would become the rating, bounded between 0 and 1. With this setting there are no *users* in the test set that are not present in the training set which simplifies predictions.

For mapping the *item*, several options seemed to be available to us. Here we use two selections as *items* 1) Solving-step (a combination of problem hierarchy, problem name, step name, and problem view); and 2) Skills (knowledge components). Please refer to the article (Thai-Nghe et al., 2010) for more information about how to map these data to user/item in recommender systems. Information about the number of users, items, and ratings used in this study are presented in Table 2.

[2]http://pslcdatashop.web.cmu.edu/KDDCup/

Table 2: Mapping student performance data to user/item in recommender systems. Solving-step is a combination of problem hierarchy (PH), problem name (PN), step name (SN), and problem view (PV). Skill is also called knowledge component (KC)

	Algebra	Bridge
User	**#User**	**#User**
Student	3,310	6,043
Item	**#Item**	**#Item**
Solving-Step (PH, PN, SN, PV)	1,416,473	887,740
Skill (KC)	8,197	7,550
Rating	**#Rating**	**#Rating**
Correct first attempt	8,918,054	20,012,498

5.2 Mapping Educational Data to Regression Problem

As most of the columns available both in train and test were categorical, we needed to pre-process the data before we could regress on it. Of course, one could use "normalize to binary" strategy but in that case the datasets are quite large and very sparse (for example, the solving-step after binarizing will have 8,918,054 (20,012,498) rows and 1,416,473 (887,740) columns for Algebra (Bridge) data set). Thus, we mainly derived user/item averages to aggregate the variables as input for our regression models, as described in (Thai-Nghe et al., 2010). In the specific data sets from table 2, the variables we computed the respective averages on are: student ID, solving-step ID, and skill ID.

5.3 Matrix Factorization – Implicitly Encoding the "Slip" and "Guess" Factors

Recall that there are 2 latent factors that should be taken into account when predicting student performance: 1) "Guess": the probability that the students do not know how to solve the problem (or do not know any skill related to the problem) but still attempt to perform the task correctly by guessing, and 2) "Slip": the probability that the students know how to solve the problems (or know all of the required skills related to the problem) but they make a mistake.

Matrix factorization techniques, one of the most successful methods for item prediction, are appropriate for this issue because these "slip" and "guess" factors could be implicitly encoded in the latent factors of factorization models.

Besides the matrix factorization model as in equation (1), we also employ the biased matrix factorization model to deal with the problem of "user effect" and "item effect". On the educational setting the user and item bias are, respectively, the student and solving-step biases. They model how good a student is (i.e. how likely is the student to perform a task correctly) and how difficult the solving-step is (i.e. how likely is the step in general to be performed correctly).

The prediction function for user u and item i is determined by

$$\hat{r}_{ui} = \mu + b_u + b_i + \sum_{k=1}^{K} w_{uk} h_{ik} \quad (5)$$

where μ, b_u, and b_i are global average, user bias and item bias, respectively.

5.4 Tensor Factorization for Exploring the Temporal Effect

In section 5.3, we use matrix factorization models to take into account the latent factors "slip" and "guess" but not the temporal effect. Obviously, in education point of view: "The more the learners study the better the performance they get". Moreover, the knowledge of the learners will be cumulated over the time, thus the temporal effect is an important factor to predict the student performance. We adopt the idea from (Dunlavy et al., 2010) which applies tensor factorization for link prediction. Instead of using only two-mode tensor (a matrix) as in the previous section, we now add one more mode to the models - the time mode. We also take into account the "user bias" and "item bias". The prediction function now becomes:

$$\hat{r}_{uiT} = \mu + b_u + b_i + \left(\sum_{k=1}^{K} w_{uk} h_{ik} \Phi_{Tk} \right) \quad (6)$$

$$\Phi_{Tk} = \frac{\sum_{t=(T-T_{max}+1)}^{T} q_{tk}}{T_{max}} \quad (7)$$

where q_k is a latent factor vector representing the time, and T_{max} is the number of solving-steps in the history that we want to go back. This is a simple strategy which averages T_{max} performance steps in the past to predict the current step. We will call this approach TFA (Tensor Factorization - Averaging).

Another important factor is that "memory of human is limited", so the students could forget what they have studied in the past, e.g., they could perform better on the lessons they learn recently than the one they learn from last year or even longer. Moreover, we recognize that the second time the students do their exercises and have more chances to learn the skills, their performance on average gets better (we will explain more about this in Figure 6 of section 6.3). Thus, we could use a decay function which reduces the weight

θ when we go back to the history. We call this approach TFW (Tensor Factorization - Weighting)

$$\Phi_{Tk} = \frac{\sum_{t=(T-T_{max}+1)}^{T} q_{tk} e^{t \cdot \theta}}{T_{max}} \quad (8)$$

An open issue for this is that we can use forecasting techniques instead of weighting or averaging. We leave this solution for future work.

6 EVALUATION

6.1 Protocol

Before describing the experimental results, we present the protocol used for evaluation. First of all the data sets were mapped from the educational context to both recommender systems and regression contexts as described in sections 5.1 and 5.2. As a baseline method, we use the *global average*, i.e. predicting the average of the target variable from the training set. One of the challenging problem of recommender systems is to deal with the new-item problem. Matrix/tensor factorization cannot produce output for *"new items"*, thus, we provide global average scores for items that are not in the training data.

The proposed approaches were compared with other methods such as *user average*, *biased user-item* (Koren, 2010), as well as with traditional methods such as *logistic regression*.

6.2 Results of Matrix Factorization

Figures 5 presents the comparison of root mean squared error (RMSE) for Algebra data set. Clearly, when using both matrix factorization and biased matrix factorization on the student (as *user*) and the required skills (as *item*) to solve the step, the result is significantly improved compare to other methods such as biased user-item or logistic regression. Moreover, the proposed methods can deal with the "slip" and "guess" by implicitly encoding them as latent factors. (We also tried with linear regression, but the results of linear regression and logistic regression are very similar, we just report on logistic regression here).

6.3 Results of Tensor Factorization for Exploring Temporal Effects

Previous section shows how we encode the "slip" and "guess" factors to the model. In this section, we present the result of taking into account the temporal

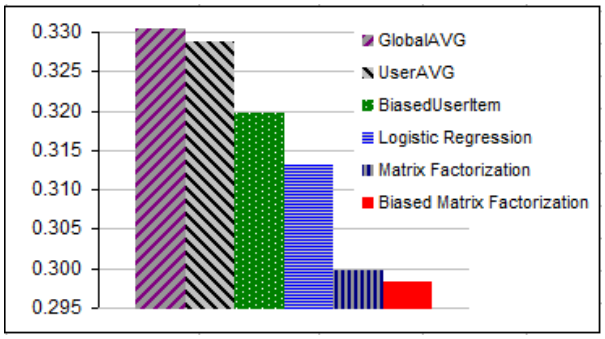


Figure 5: RMSE results of (biased) matrix factorization which factorize on student and the required skills in solving the step compared to other methods on Algebra data set. The lower the better.

information. Figure 6 describes the effect of the time on knowledge of the learners for Bridge data set (the trend of Algebra data set looks nearly the same). In this figure, the *x-axis* is the number of times that the students have chances to learn the skills (the "opportunity count" column in the data set), the *y-axis* is the ratio of the number of students solving the problem correctly. Clearly, we can see that their performance has been improved when they have more opportunities to learn the skills. This trend also reflects the educational factor that "the more the students learn, the better their performance they get".

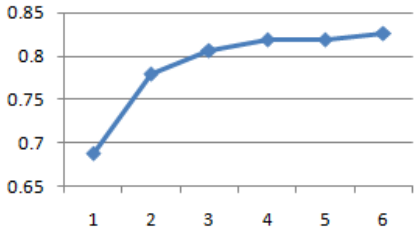


Figure 6: The effect of the time on knowledge of the learners (Bridge data set). The *x-axis* presents the number of times the student learns the knowledge components. The *y-axis* is the probability of solving the problem correctly.

Figure 7 presents the RMSE of tensor factorization methods which factorize on the student (as *user*), solving-step (as *item*), and the sequence of solving-step (as *time*) for the Bridge data set. The results of the proposed methods are also improved compared to the others. Compared with matrix factorization which does not take the temporal effect into account, the tensor factorization methods have also been improved. This result somehow reflects the natural fact that we mentioned before: "the knowledge of the student is improved over the time and human memory is limited". However, the result of tensor factorization by

weighting with a decay function (TFW) has a small improvement compared to the method of averaging on the time mode (TFA).

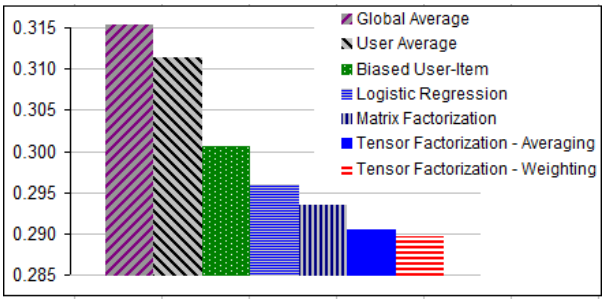


Figure 7: RMSE results of taken into account the temporal effect using tensor factorization (Bridge data set) which factorize on student, solving-step, and time. The lower the better.

For referencing, we also report the best hyperparameters that we found via cross-validation in Table 3.

7 DISCUSSION

To this end, one would raise the following question: *"how the delivered recommendations look like?"*. Actually, we have not explicitly answered for this question because of two reasons:

- First, we would like to discover how recommender systems (e.g. factorization techniques) can be applied for educational performance data, especially for *predicting student performance* but not for recommending learning objects.
- Second, we would like to focus on the first step in recommender systems to see how high quality of the rating score that the proposed methods can produce is. (Basically, every recommender system has "two steps": The first step is to generate the score, e.g rating; and the second step is to wrap around with an interface, e.g. a web page)

However, we can deliver recommendations for many problems. For example, in the scenario of section 3, when the students learn the formula of calculating the area of the circle and square, we could recommend them the similar skills such as rectangle or parallelogram, or alike. Another example is recommendation of similar grammar structures, vocabularies, or even a similar problem/section when student learning or doing exercises in an English course, etc.

Moreover, another question could be raised is that *"How good our approaches are compared to the others on the same KDD Challenge data?"*. We can have an overview on the RMSE score of the best student teams in Table 4. Note, that our purpose is not producing a system for the KDD Challenge but on getting the real datasets from this event and applying recommender systems, especially factorization techniques for predicting student performance. Although the other methods reached lower RMSE, they are more complex and require much effort on data preprocessing (e.g. feature engineering,..) as well as generating hundred of models and ensembling approaches. Factorization methods are simple to implement and need not so much human effort and computer memory to deal with large datasets (e.g, we just need 2 and 3 features for matrix and tensor factorization, respectively). More complex models using matrix factorization can also produce the better results as shown in (Toscher and Jahrer, 2010). This means that factorization approaches are promising for the problem of predicting student performance.

Table 4: RMSE Averaging on Algebra and Bridge - KDD Challenge 2010 - Student teams

Rank	Team Name	RMSE
1	National Taiwan University	0.27295
2	Zach A. Pardos	0.27659
3	SCUT Data Mining	0.28048
-	**Our approach (not submitted)**	0.29519
4	Y10	0.29801

The third issue should also be discussed is that *"what does the presented RMSE reduction mean in practical? e.g., how can it help the education managers, students, etc?"*. As in the Netflix Prize[3] it has been shown that a 10% of improvement on RMSE could bring million of dollars for recommender systems of a company. In this work, we are on educational environment, and of course, the benefit is not explicitly as in e-commerce but the trend is very similar. Moreover, in (Cen et al., 2006) it has been shown that an improved model (e.g. lower RMSE) for predicting student performance could save millions of hours of students' time and effort since they can learn other courses or do some useful activities. The better a model captures the student's skills, the lower will be it's error in predicting the student's performance, thus, an improvement on RMSE would show that our models are better capable of capturing how much the student has learned; and finally, a good model which accurately predicts student performance could replace some current standardized tests.

[3] http://www.netflixprize.com

Table 3: Hyperparameters are used for the experiments

Method	Data set	Hyperparameters
Matrix factorization	Algebra	learning-rate=0.01, #iter=60, K=64, λ=0.01
Biased matrix factorization	Algebra	learning-rate=0.001, #iter=80, K=128, λ=0.0015
Matrix factorization	Bridge	learning-rate=0.01, #iter=80, K=64, λ=0.015
Tensor factorization - Averaging	Bridge	learning-rate=0.01, #iter=30, K=32, λ=0.015, T_{max}=8
Tensor factorization - Weighting	Bridge	learning-rate=0.01, #iter=30, K=32, λ=0.015, T_{max}=8, θ= 0.4

8 CONCLUSION

We propose using state-of-the-art recommender system techniques for predicting student performance. We introduce and formulate this problem and show how to map it into recommender systems. We propose using matrix factorization to implicitly take into account two latent factors "slip" and "guess" in predicting student performance. Moreover, the knowledge of the learners improve time by time, thus, we propose tensor factorization methods to take the temporal effect into account. Experimental results show that the proposed approaches are promising.

In future work, instead of using averaging or weighting approached on the third mode of tensor, we could use forecasting approach to take into account the sequential effect. Moreover, each solving-step relates to one or many skills, thus, we could apply multi-relational matrix factorization to factorize this problem.

ACKNOWLEDGMENTS

The first author was funded by the "TRIG - Teaching and Research Innovation Grant" project of Cantho university, Vietnam. The second author was funded by the CNPq, an institute of the Brazilian government for scientific and technological development. Tomáš Horváth is also supported by the grant VEGA 1/0131/09. We would like to thank Artus Krohn-Grimberghe for preparing the data sets.

The MyMediaLite recommender system library (http://ismll.de/mymedialite) was used for some of the experiments.

REFERENCES

Bekele, R. and Menzel, W. (2005). A bayesian approach to predict performance of a student (bapps): A case with ethiopian students. In *Artificial Intelligence and Applications*, pages 189–194. Vienna, Austria.

Bell, R. M. and Koren, Y. (2007). Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In *Proceedings of the 7th IEEE International Conference on Data Mining (ICDM'07)*, pages 43–52, Washington, USA. IEEE CS.

Bobadilla, J., Serradilla, F., and Hernando, A. (2009). Collaborative filtering adapted to recommender systems of e-learning. *Knowledge-Based Systems*, 22(4):261–265.

Bottou, L. (2004). Stochastic learning. In Bousquet, O. and von Luxburg, U., editors, *Advanced Lectures on Machine Learning*, Lecture Notes in Artificial Intelligence, LNAI 3176, pages 146–168. Springer Verlag, Berlin.

Cen, H., Koedinger, K., and Junker, B. (2006). Learning factors analysis a general method for cognitive model evaluation and improvement. In *Intelligent Tutoring Systems*, volume 4053, pages 164–175. Springer Berlin Heidelberg.

Drachsler, H., Hummel, H. G. K., and Koper, R. (2009). Identifying the goal, user model and conditions of recommender systems for formal and informal learning. *Journal of Digital Information*, 10(2).

Dunlavy, D. M., Kolda, T. G., and Acar, E. (2010). Temporal link prediction using matrix and tensor factorizations. *ACM Transactions on Knowledge Discovery from Data*. In Press.

Feng, M., Heffernan, N., and Koedinger, K. (2009). Addressing the assessment challenge with an online system that tutors as it assesses. *User Modeling and User-Adapted Interaction*, 19(3):243–266.

García, E., Romero, C., Ventura, S., and Castro, C. D. (2009). An architecture for making recommendations to courseware authors using association rule mining and collaborative filtering. *User Modeling and User-Adapted Interaction*, 19(1-2).

Ge, L., Kong, W., and Luo, J. (2006). Courseware recommendation in e-learning system. In *In-*

ternational Conference on Web-based Learning (ICWL'06), pages 10–24.

Ghauth, K. and Abdullah, N. (2010). Learning materials recommendation using good learners' ratings and content-based filtering. *Educational Technology Research and Development, Springer-Boston*, pages 1042–1629.

Khribi, M. K., Jemni, M., and Nasraoui, O. (2008). Automatic recommendations for e-learning personalization based on web usage mining techniques and information retrieval. In *Proceedings of the 8th IEEE International Conference on Advanced Learning Technologies*, pages 241–245. IEEE Computer Society.

Kolda, T. G. and Bader, B. W. (2009). Tensor decompositions and applications. *SIAM Review*, 51(3):455–500.

Koren, Y. (2010). Factor in the neighbors: Scalable and accurate collaborative filtering. *ACM Trans. Knowl. Discov. Data*, 4(1):1–24.

Koren, Y., Bell, R., and Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *IEEE Computer Society Press*, 42(8):30–37.

Manouselis, N., Drachsler, H., Vuorikari, R., Hummel, H., and Koper, R. (2010). Recommender systems in technology enhanced learning. In *Kantor, P.B., Ricci, F., Rokach, L., Shapira, B. (eds.) 1st Recommender Systems Handbook*, pages 1–29. Springer-Berlin.

Massey, L. D., Psotka, J., and Mutter, S. A. (1988). *Intelligent tutoring systems : lessons learned / edited by Joseph Psotka, L. Dan Massey, Sharon A. Mutter, advisory editor, John Seely Brown*. L. Erlbaum Associates, Hillsdale, N.J.

Pardos, Z. A. and Heffernan, N. T. (2010). Using hmms and bagged decision trees to leverage rich features of user and skill from an intelligent tutoring system dataset. *KDD Cup 2010: Improving Cognitive Models with Educational Data Mining*.

Rendle, S., Freudenthaler, C., and Schmidt-Thieme, L. (2010). Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th International Conference on World Wide Web (WWW'10)*, pages 811–820, New York, USA. ACM.

Rendle, S. and Schmidt-Thieme, L. (2008). Online-updating regularized kernel matrix factorization models for large-scale recommender systems. In *Proceedings of the ACM conference on Recommender Systems (RecSys'08)*, pages 251–258, New York, USA. ACM.

Romero, C., Ventura, S., Espejo, P. G., and Hervs, C. (2008). Data mining algorithms to classify students. In *1st International Conference on Educational Data Mining (EDM'08)*, pages 8–17. Montral, Canada.

Soonthornphisaj, N., Rojsattarat, E., and Yim-ngam, S. (2006). Smart e-learning using recommender system. In *International Conference on Intelligent Computing*, pages 518–523.

Thai-Nghe, N., Busche, A., and Schmidt-Thieme, L. (2009). Improving academic performance prediction by dealing with class imbalance. In *Proceeding of 9th IEEE International Conference on Intelligent Systems Design and Applications (ISDA'09)*, pages 878–883. Pisa, Italy, IEEE Computer Society.

Thai-Nghe, N., Drumond, L., Krohn-Grimberghe, A., and Schmidt-Thieme, L. (2010). Recommender system for predicting student performance. In *Proceedings of the 1st Workshop on Recommender Systems for Technology Enhanced Learning (RecSysTEL 2010)*, volume 1, pages 2811 – 2819. Elsevier's Procedia CS.

Thai-Nghe, N., Janecek, P., and Haddawy, P. (2007). A comparative analysis of techniques for predicting academic performance. In *Proceeding of 37th IEEE Frontiers in Education Conference (FIE'07)*, pages T2G7–T2G12. Milwaukee, USA, IEEE Xplore.

Toscher, A. and Jahrer, M. (2010). Collaborative filtering applied to educational data mining. *KDD Cup 2010: Improving Cognitive Models with Educational Data Mining*.