Factorization Techniques for Predicting Student Performance

Nguyen Thai-Nghe, Lucas Drumond, Tomáš Horváth, Artus Krohn-Grimberghe, Alexandros Nanopoulos, and Lars Schmidt-Thieme

Abstract Recommender systems are widely used in many areas, especially in e-commerce. Recently, they are also applied in e-learning for recommending learning objects (e.g. papers) to students. This chapter introduces state-of-the-art recommender system techniques which can be used not only for recommending objects like tasks/exercises to the students but also for predicting student performance. We formulate the problem of predicting student performance as a recommender system problem and present matrix factorization methods, which are currently known as the most effective recommendation approaches, to implicitly take into account the prevailing latent factors (e.g. "slip" and "guess") for predicting student performance. As a learner's knowledge improves over time, too, we propose tensor factorization methods to take the temporal effect into account. Finally, some experimental results and discussions are provided to validate the proposed approach.

INTRODUCTION

Recommender systems are widely used in many areas, especially in e-commerce (Rendle et al., 2010). One of their main aims is to make vast catalogs of products consumable by learning user preferences and to apply them to items formerly unknown to the user. Thus they can learn which products have a high likelihood of being interesting to the target user. Recently, recommender systems have also been applied to e-learning, especially in technology enhanced learning (Manouselis et al., 2010).

On the other hand, educational data mining has also been taken into account recently to assist the students in the learning process. One of the main educational data mining tasks, for instance, is to predict student performance. It is applicable when we would like to know how the students learn (e.g. generally or narrowly),

University of Hildesheim, Germany, e-mail: {nguyen, ldrumond, horvath, artus, nanopoulos, schmidt-thieme}@ismll.de

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). Generally speaking, the prediction of student performance is the problem of predicting the student's ability (e.g., estimated by a score metric) in solving tasks when interacting with a tutoring system. Cen et al. (2006) have shown that an improved model for predicting student performance could save millions of hours of students' time and effort in learning algebra that they could otherwise have spent on other subjects or leisure. Moreover, many universities are extremely focused on assessment, thus, the pressure on teaching and learning for examinations leads to a significant amount of time spent for preparing and taking standardized tests. Any move away from standardized and non-personalized tests holds promise for 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 and yield truly personalized, adaptive test.

To address the student performance prediction problem, many works have been published. Most of them rely on traditional methods such as neural networks (Romero et al., 2008), Bayesian networks (Bekele and Menzel, 2005), logistic regression (Cen et al., 2006), support vector machines (Thai-Nghe et al., 2009) and so on.

In the recommender system context, predicting student performance can be considered as a rating prediction problem since student, task, and performance information could be treated as user, item, and rating, respectively, which are the main objects recommender systems learn from, nowadays. Recently, Thai-Nghe et al. (2010a); Toscher and Jahrer (2010) have proposed the use of 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 (Thai-Nghe et al., 2010b) but they have not taken the temporal effect into account. Obviously, in the educational point of view, we always expect that the students (or generally, the learners) can improve their knowledge over time, so temporal information is an important factor for such prediction tasks.

Furthermore, in predicting student performance, two crucial user-dependent aspects need to be taken into account:

1. The probability of a student to guesses correctly while not knowing how to solve the problem at hand or not having the required skills related to the problem (which we call "guess" for short); and the probability of a student to fail while knowing how to solve the problem or having all of the required skills related to the problem (which we call "slip" for short);
2. The increase in knowledge over time obviously has an effect on a student's performance, e.g. the second time a student does his exercises, the performance gets better on average, and therefore, the sequential effect is important information.

Factorization techniques are a promising approach for this problem since they i) implicitly take into account the latent factors such as "slip" and "guess", and ii) incorporate the sequential (time) aspect to their models. Moreover, the impor-

tance of the proposed approach is that besides predicting student performance, we can also recommend the similar tasks (exercises) to the students and can determine which tasks are notoriously difficult for a given student. The predicted student performance represents the estimation of a student's performance on a given exercise. For example, there is a huge database of exercises where students lose a lot of time solving problems which are too easy or too hard for them. When a system is able to predict their performance, it could recommend more appropriate exercises for them. Thus, we could filter out the tasks with predicted high performance / confidence since these tasks are too easy for them, or filter out the tasks with predicted low performance (too hard for them) or both, depending on the goals of the e-learning system. As mentioned in the literature, recommender systems for educational purposes are a complex and challenging research direction (Drachsler et al., 2009). The preferred learning activities of students might pedagogically not be the most adequate (Tang and McCalla, 2004) and recommendations in e-learning should be guided by educational objectives, and not only by the user's preferences (Santos and Boticario, 2010). Our approach which uses student performance prediction for the recommendation of e-learning tasks would be a promising approach to tackle the above problems since we can recommend the tasks for students based on their performance but not on their preferences.

Please note that in this chapter we have not focused on building a real system, e.g. how the system interface looks like or which particular objects are recommended, but on how to apply the state-of-the-art recommender system techniques to generate the most accurate performance estimation. More precisely, the purposes of this chapter are to

- formulate the problem of predicting student performance in the context of recommender systems;
- propose matrix factorization models for predicting student performance, which are extended from previous works (Thai-Nghe et al., 2010a; Toscher and Jahrer, 2010) and presented in a formalized and detailed way that can be helpful for the researchers who come from educational domains. We can empirically show that these factorization techniques can implicitly encode the "slip" and "guess" factors as well as "student effect" (e.g. how clever/good the student is in performing the task) and the "task effect" (e.g. how hard/easy the task is in general) in predicting student performance;
- propose tensor factorization techniques to take into account the temporal effect, e.g. how the knowledge of the learners improves over time;
- map the student performance data from two real data sets to recommender system context;
- experiment with the proposed approach and compare with some other baselines.

BACKGROUND

In the area of recommender systems for technology enhanced learning (Manouselis et al., 2010), almost all the works are focused on the construction of recommender systems for recommendation of learning resources (materials/ resources) or learning activities to the learners (Ghauth and Abdullah, 2010; Manouselis et al., 2010) in both formal and informal learning environments (Drachsler et al., 2009). More precisely, García et al. (2009) use association rule mining to discover interesting information through student performance data in the form of IF-THEN rules, followed by generating 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. This use of recommender systems corresponds to the so-called item recommendation (or item prediction) task in the recommender literature. But what so far is missing in e-learning systems is the exploitation of the recommender systems capability to predict scores - a task called rating prediction.

On the student performance prediction problem, many works have been published, too. Most of them rely on traditional methods. E.g., Romero et al. (2008) compared different data mining methods and techniques (like neural networks, decision trees, etc.) 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 using support vector machines (i.e., the ratio between passing and failing students is usually skewed).

Although some novel approaches have been proposed for predicting student performance recently, there are still several open issues. So far, the works have just focus on explicit modeling of the "slip" and "guess" latent factors, e.g. using Hidden Markov Models as Pardos and Heffernan (2010) did. Or they deal with thousands of features using feature engineering and hundreds of models stacked together in large ensembles as Yu et al. (2010) proposed. Thus, those suggested approaches need tremendous computational resources and human effort for data pre-processing. Yet, they do not take the temporal effect of the underlying process into account as in Thai-Nghe et al. (2010b); Toscher and Jahrer (2010).

Usually, the mentioned approaches work well when sufficient information is available about the students as well as tasks or learning resources. However, in real-

world applications we face the problem of information acquisition as the collection of attributes and metadata about users, tasks and resources is often very expensive or even not possible at all. The information easily obtainable is the results the students yielded on the various tasks they previously worked on. This information could be captured by the student, task, score tuple. Interestingly, this is exactly the kind of information, current recommender systems rely on, though in the recommender terminology the above mentioned tuple would be rephrased as user, item, rating. This perspective allows two distinct kinds of treatment. First, the so-called "implicit feedback" view where only the information that a specific user (student) as interacted with a given item (task). This can nicely be represented by a binary or even unary matrix on users and items storing those interactions. Second, the "explicit feedback" view where a numerical (preference) value can be obtained as the result of the interaction between user and item. This, too, can be represented as an interaction matrix on users and items. In both cases, however, the resulting matrix will be sparse, as most of the students will tackle only a (small) subset of the tasks / exercises available in the e-learning system. For recommender systems, the so called collaborative filtering approach prevails since the late nineties. Collaborative filtering is based on the assumption that similar users like similar things and, being content-agnostic, focuses only on the past ratings assigned. Factorization techniques are the most popular collaborative filtering techniques (Koren et al., 2009), used for both item and rating recommendations. Moreover, it has been shown that even a few ratings are often more valuable than meta data (Pilászy and Tikk, 2009), thus, the use of factorization techniques are in place for recommendation task. In case of sparse data with no additional metadata, the use of collaborative filtering techniques has been shown to be very effective.

Factorization techniques outperform other state-of-the-art collaborative filtering techniques. They belong to the family of latent factor models which aim at mapping users and items to a common latent space by representing them as vectors in that space. The dimensions of this space are called the factors. Here we should mention, that we usually don't now the exact "meaning" of these factors, we are just interested in the correlation between the vectors in that space. For example, imagine a well-known simplified movie rating example where users and movies are mapped to a two-dimensional latent space (Koren et al., 2009). Here, each user and each movie is represented by two factors. These factors can represent genre, seriousness, amount of action, quality of actors or any other concept. Even if we don't know what exactly the given two factors (dimensions of the latent space) represent, the "closeness" of particular user and movie vectors in that space expresses the preference of a given user on the given movie. Thus, we can expect with high certainty that the before mentioned slip and guess factors are somehow encoded in the latent factors of the computed factorization models and we do not need to explicitly take care of them as in case of other methods, e.g. Hidden Markov Models (Pardos and Heffernan, 2010). Other latent characteristics of students and tasks could also be implicitly encoded in the models such as how good the performance of a student is, how hard/easy the task is, etc. Furthermore, by using a tensor with three or more dimensions instead of a matrix we can represent the sequential (time) aspect or any other additional

context such as students' skills. As also mentioned in Pilászy and Tikk (2009), the performance of these techniques are very good in case of using just two or three features such as student ID, task ID and/or time. Thus, memory consumption and the human effort in pre-processing can be reduced significantly while the prediction quality is reasonable.

With the recommendation of learning resources, recommender systems already address an important problem in technology enhanced learning. Nonetheless, with rating prediction a highly appreciated capability of recommender systems is so far underexplored in e-learning environments. As we have already seen, all information necessary for recommender systems for both item and rating prediction is available in these systems and could be exploited. Furthermore, we have highlighted the power of recommender systems in addressing latent factors and their excellent runtime characteristics. The remainder of this chapter will proceed to show, how rating prediction can be mapped to predicting student performance.

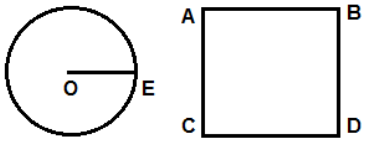
PREDICTING STUDENT PERFORMANCE

Figure 1a presents an example (Koedinger et al., 2010) of the task in predicting student performance. 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?" To solve this task (question), students could do some smaller subtasks which we call a "step". Each step may require one or more skills (or "knowledge components"), for example:

- Step 1: Calculate the circle area ($area_1 = \pi * (OE)^2$)
- Step 2: Calculate the square area ($area_2 = (AB)^2$)
- Step 3: Calculate the remaining ($area_2 - area_1$)

Each solving-step is recorded as a transaction. Figure 1b presents a snapshot of transactions. Based on the past performance, we would like to predict students' next performance (e.g. correct/incorrect) in solving the tasks. To do this we first need to consider the amount and the quality of information we have collected about the students and tasks as well as their interactions. Such data can be collected from some Student Information Systems (SIS) or from the Computer-Aided Tutoring Systems (CATS).

Computer-aided tutoring systems (CATS) allow students to solve some exercises with a graphical front-end 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. Thus, CATS are valuable environments for collecting data for prediction and for interacting with students in an intelligent way.



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
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	?
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	?

Past perform-ance

Predict

b. A snaphot of the transaction log

Fig. 1 Predicting student performance: A scenario. (Adapted from PSLC-Datashop.web.cmu.edu/KDDCup).

Problem formulation in the context of Computer-Aided Tutoring Systems (CATS)

The problem of student performance prediction in CATS is 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. In CATS, tasks usually are described in two different ways: First, tasks can be located in a topic hierarchy, for example

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

Second, tasks can be described by additional meta data 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 meta data can be described as attributes of the tasks. In the same way, also attributes about the student may be available.

CATS allow collecting 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 may contain many useful information, e.g. 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.

More formally, let S be a set of students, T be a set of tasks, and $P \subseteq \mathbb{R}$ be a range of possible performance scores. Let M_S be a set of student meta data descriptions and $m_S : S \rightarrow M_S$ be the meta data for each student. Let M_T be a set of task meta data descriptions and $m_T : T \rightarrow M_T$ be the meta data for each task. Finally, let $\mathscr{D}^{train} \subseteq (S \times T \times P)^*$ be a sequence of observed student performances and $\mathscr{D}^{test} \subseteq (S \times T \times P)^*$ be a sequence of unobserved student performances. Furthermore, let

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

and

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

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

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

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

such that

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

is minimal with $p := \pi_p(\mathscr{D}^{test})$. Some other error measures could also be considered.

In principle, this is a regression or classification problem (depending on the error measure). Specifically, we have to treat (i) two variables, e.g. the student ID and the task ID, both being nominal with many levels (1,000-100,000s), which can be *casted as rating prediction in recommender systems* since s,t and p would be *user, item* and *rating*, respectively (Thai-Nghe et al., 2010a), as illustrated in Figure 2. Thus, we could apply factorization techniques or other collaborative filtering techniques to this problem; and (ii) *potentially sequential effects* which describes how students gain experience over time.

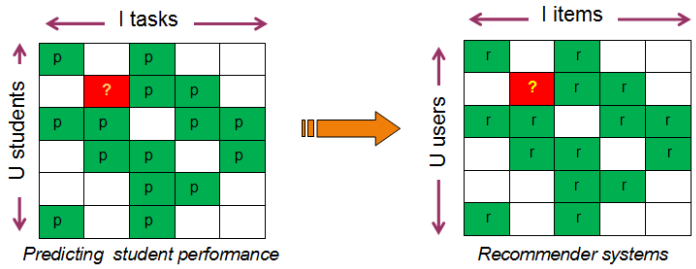


Fig. 2 An illustration of casting student/task/performance in PSP as user/item/rating in RecSys

FACTORIZATION TECHNIQUES

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.

Please note that the techniques presented in this chapter compute the prediction as a linear combination of factors. Although it may happen that in some cases linear combination of factors can be insufficient. In such cases, we can use non-linear extensions of factorization techniques (Lawrence and Urtasun, 2009). However, as showed in Takács et al. (2009), using the standard factorization models is enough to reach good prediction accuracy in an efficient and scalable way. The results of this approach in our experiments (provided later in this chapter) as well as the results of experiments in other domains (Koren et al., 2009) constitute some empirical evidence that assuming a linear interaction between the factors is a reasonable approach. The connection of factorization techniques to other methods, such as Neural Networks, can be seen e.g. in (Takács et al., 2009). We will describe the basic matrix and tensor factorization techniques to present the idea behind these approaches. Discussion of other advanced factorization techniques is out of the scope of this chapter.

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. We denote r as the actual value and $\hat{r}$ as the prediction value. From here to the end of the chapter, we will call student, task, and performance instead of user, item, and rating, respectively.

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). $W \in \mathbb{R}^{U \times K}$ is a matrix where each row u is a vector containing the K latent factors describing the student 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 task i. Let w_{uk} and h_{ik} be the elements of W and H, respectively, then the performance p given by a student u to a task i is predicted by:

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

The main issue of this technique is how to find the optimal parameters W and H (the elements of W and H) given a criterion such as root mean squared error (RMSE):

$$RMSE = \sqrt{\frac{\sum_{ui \in \mathscr{D}^{test}} (p_{ui} - \hat{p}_{ui})^2}{|\mathscr{D}^{test}|}} \tag{2}$$

Training Phase

In this approach, training the model is finding the optimal parameters W and H. One approach is that we first initialize these two matrices with some random values (e.g. from the normal distribution $\mathscr{N}(0, \sigma^2)$ with mean is 0 and standard deviation σ^2 = 0.01), and compute the error function

$$err = \sum_{(u,i) \in \mathscr{D}^{train}} e_{ui}^2 \tag{3}$$

where

$$e_{ui}^2 = (p_{ui} - \hat{p}_{ui})^2 = (p_{ui} - \sum_{k=1}^{K} w_{uk} h_{ik})^2 \tag{4}$$

then try to minimize this error by updating the values of W and H iteratively. Such a method is called gradient descent (Koren et al., 2009). To minimize equation (3), we need to know for each data point in which direction to update the value of w_{uk} and h_{ik}. Thus, we compute the gradient of the function (4):

$$\frac{\partial}{\partial w_{uk}} e_{ui}^2 = -2 e_{ui} h_{ik} = -2(p_{ui} - \hat{p}_{ui}) h_{ik} \tag{5}$$

$$\frac{\partial}{\partial h_{ik}} e_{ui}^2 = -2 e_{ui} w_{uk} = -2(p_{ui} - \hat{p}_{ui}) w_{uk} \tag{6}$$

After having the gradient, we update the values of w_{uk} and h_{ik} in the direction opposite to the gradient:

$$w'_{uk} = w_{uk} - \beta \frac{\partial}{\partial w_{uk}} e_{ui}^2 = w_{uk} + 2\beta e_{ui} h_{ik} \tag{7}$$

$$h'_{ik} = h_{ik} - \beta \frac{\partial}{\partial h_{ik}} e_{ui}^2 = h_{ik} + 2\beta e_{ui} w_{uk} \tag{8}$$

where β is the learning rate. We iteratively update the values of W and H until the error converges to its minimum or reaching a predefined number of iterations.

Regularization: To prevent overfitting, we modify the error function (4) by adding a term which controls the magnitudes of the factor vectors such that W and H would give a good approximation of X without having to contain large numbers. The function (4) now becomes:

$$(p_{ui} - \sum_{k=1}^{K} w_{uk} h_{ik})^2 + \lambda(||W||^2 + ||H||^2) \tag{9}$$

where λ is a regularization term. With this new error function, the values of w_{uk} and h_{ik} are updated by

$$w'_{uk} = w_{uk} + \beta(2e_{ui} h_{ik} - \lambda w_{uk}) \tag{10}$$

$$h'_{ik} = h_{ik} + \beta(2e_{ui} w_{uk} - \lambda h_{ik}) \tag{11}$$

Recall that there are 2 issues that should be taken into account when predicting student performance, namely the "guess" and "slip" factors expressing the probabilities that the students will guess correctly or make a mistake. Matrix factorization techniques are appropriate for handling these issues because the mentioned "slip" and "guess" factors could be implicitly scrambled in the latent factors of W and H.

Algorithm 1 describes details of training a matrix factorization model using stochastic gradient descent (we use stochastic gradient descent for all algorithms in this chapter since it has been shown that the computing cost of stochastic gradient descent has a huge advantage for large-scale problems (Bottou, 2004)). This method factorizes the student and the task that student wants to solve. First, the parameters W and H are initialized randomly from the normal distribution $\mathcal{N}(0, \sigma^2)$ with mean is 0 and standard deviation σ^2 = 0.01, as in lines 2-3. In each iteration, we randomly select an instance in the training set (u, i, p), then compute the prediction for this student and task, as in lines 5-9. We estimate the error in this iteration and update the values of W and H as in lines 11-14.

Algorithm 1 A matrix factorization for factorizing student and task using stochastic gradient descent with $Iter$ iterations, K factors, β learning rate, and λ regularization

procedure STUDENT-TASK-MATRIXFACTORIZATION($\mathscr{D}^{Train}, Iter, K, \beta, \lambda$)
  Let $u \in S$ be a student, $i \in T$ a task, $p \in P$ a performance score
  Let $W[|S|][K]$ and $H[|T|][K]$ be latent factors of students and tasks
  $W \leftarrow \mathscr{N}(0, \sigma^2)$
  $H \leftarrow \mathscr{N}(0, \sigma^2)$
  for $iter \leftarrow 1, \ldots, Iter * |\mathscr{D}^{Train}|$ **do**
    Draw randomly (u, i, p) from $\mathscr{D}^{Train}$
    $\hat{p} \leftarrow 0$
    for $k \leftarrow 1, \ldots, K$ **do**
      $\hat{p} \leftarrow \hat{p} + W[u][k] * H[i][k]$
    end for
    $e_{ui} = p - \hat{p}$
    for $k \leftarrow 1, \ldots, K$ **do**
      $W[u][k] \leftarrow W[u][k] + \beta * (e_{ui} * H[i][k] - \lambda * W[u][k])$
      $H[i][k] \leftarrow H[i][k] + \beta * (e_{ui} * W[u][k] - \lambda * H[i][k])$
    end for
  end for
  return $\{W, H\}$
end procedure

Prediction Phase

After having the optimal W and H, the performance of a student u in a given task i is predicted easily by

$$\hat{p}_{ui} = \sum_{k=1}^{K} w_{uk} h_{ik} \tag{12}$$

For the students or tasks in the test set but not in the train set, we return the average performance on the training set (we will discuss more about this later).

An Example

Figure 3 shows an example of how we can factorize the students and tasks. Suppose that we have 6 students and 5 exercises (tasks) which are presented in matrix X. Each task is to compute the values of y, e.g. $y = -2x$, given a specific value of x. These students have performed some tasks which are measured by correct (1) and incorrect (0) performance. Our problem is to predict the other tasks that they have not done (the empty values in X).

Note, that the presented techniques do not depend on specific tasks. Important is that the matrix represents a relation between two types of objects, which are in our case students and math tasks but any other types of objects can be also used, e.g.

students and courses, students and learning resources (as books or multimedia) as well as tasks in other disciplines.

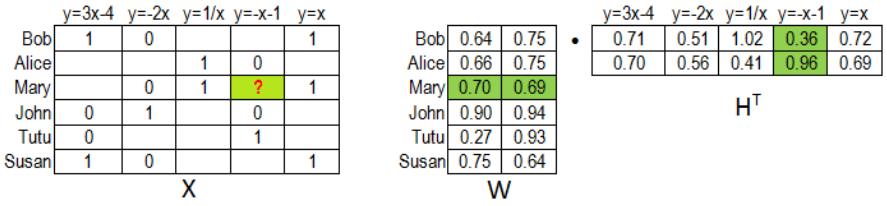


Fig. 3 An example of factorizing on students and tasks

After the training phase with $K = 2$ latent factors, we get the optimized parameters W and H as in this figure. Now, suppose that we would like to predict Mary's performance for the task $y = -x - 1$ (the cell with question mark). We can easily compute the prediction using equation (12)

$$\hat{p} = \sum_{k=1}^{K} w_{uk} h_{ik} = 0.7 * 0.36 + 0.69 * 0.96 = 0.91$$

From this prediction result, we can see that Mary can correctly answer her task with high confidence (0.91). In a similar way, we can predict the performance of other students in tasks which they have not done yet. Moreover, using this approach, besides *predicting student performance*, we can also *recommend the similar tasks (exercises) to the students*. For example, there is a huge database of exercises and that students lose most of their time solving problems which are too easy or too hard for them. By being able to predict their performance, the system could recommend just the appropriate exercises for them. The student performance represents how good the student will perform in a given exercise. Thus, we could filter out the tasks with high performance/confidence since these tasks are too easy for them, or filter out the tasks with low performance (too hard for them) or both, depending on the system goal.

Biased Matrix Factorization

Besides the basic matrix factorization model, we also employ the biased matrix factorization model to deal with the problem of "user effect" and "item effect", or we can call *"user bias"* and *"item bias"*, respectively (Koren et al., 2009). On the educational setting, the user and item bias are, respectively, the *student* and *task* biases. They model how good a student is (i.e. how likely is the student to perform a task correctly) and how difficult/easy the task is (i.e. how likely is the task in general to be performed correctly).

The performance prediction function for student u and task i is now determined by

$$\hat{p}_{ui} = \mu + b_u + b_i + \sum_{k=1}^{K} w_{uk} h_{ik} \tag{13}$$

where μ, b_u, and b_i are global average, student bias and task bias, respectively.

Algorithm 2 describes more details about this method. First, we compute the global average, student bias and task bias as in lines 2-8. Parameters W and H are again initialized randomly from the normal distribution $\mathcal{N}(0, \sigma^2)$ with mean is 0 and standard deviation σ^2 = 0.01. Then, we update the value of μ, b_u, b_i, W and H at each iteration as in lines 11-21. After getting these parameters, we can easily compute the prediction in the test set for any existing student and task with the same formula in line 13.

Algorithm 2 A biased matrix factorization for factorizing student and task using stochastic gradient descent with $Iter$ iterations, K factors, β learning rate, and λ regularization

1: **procedure** STUDENT-TASK-BIASED-MATRIXFACTORIZATION($\mathscr{D}^{Train}, Iter, K, \beta, \lambda$)
   Let $u \in S$ be a student, $i \in T$ a task, $p \in P$ a performance score
   Let $W[|S|][K]$ and $H[|T|][K]$ be latent factors of students and tasks
   Let $b_u[|S|]$ and $b_i[|T|]$ be student-bias and task-bias
2: $\mu \leftarrow \frac{\sum_{p \in \pi_p(\mathscr{D}^{Train})} p}{|\mathscr{D}^{Train}|}$
3: **for** each student u **do**
4: $b_u[u] \leftarrow \frac{\sum_i (p_{ui} - \mu)}{|\mathscr{D}_u^{Train}|}$
5: **end for**
6: **for** each task i **do**
7: $b_i[i] \leftarrow \frac{\sum_u (p_{ui} - \mu)}{|\mathscr{D}_i^{Train}|}$
8: **end for**
9: $W \leftarrow \mathcal{N}(0, \sigma^2)$
10: $H \leftarrow \mathcal{N}(0, \sigma^2)$
11: **for** $iter \leftarrow 1, \ldots, Iter * |\mathscr{D}^{Train}|$ **do**
12: Draw randomly (u, i, p_{ui}) from $\mathscr{D}^{Train}$
13: $\hat{p}_{ui} \leftarrow \mu + b_u[u] + b_i[i] + \sum_k^K (W[u][k] * H[i][k])$
14: $e_{ui} = p_{ui} - \hat{p}_{ui}$
15: $\mu \leftarrow \mu + \beta * e_{ui}$
16: $b_u[u] \leftarrow b_u[u] + \beta * (e_{ui} - \lambda * b_u[u])$
17: $b_i[i] \leftarrow b_i[i] + \beta * (e_{ui} - \lambda * b_i[i])$
18: **for** $k \leftarrow 1, \ldots, K$ **do**
19: $W[u][k] \leftarrow W[u][k] + \beta * (e_{ui} * H[i][k] - \lambda * W[u][k])$
20: $H[i][k] \leftarrow H[i][k] + \beta * (e_{ui} * W[u][k] - \lambda * H[i][k])$
21: **end for**
22: **end for**
23: **return** $\{W, H, b_u, b_i, \mu\}$
24: **end procedure**

Tensor Factorization

In previous section, we used matrix factorization models to take into account the latent factors "slip" and "guess" but not the temporal effect. Obviously, from the 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, thus, this method 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 student and task 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 by using Tucker decomposition (Tucker, 1966):

$$\mathbf{Z} \approx \sum_{r=1}^{R} \sum_{s=1}^{S} \sum_{p=1}^{P} \mathbf{C}_{rsp}\, w_r \circ h_s \circ q_p \tag{14}$$

or by using CANDECOM-PARAFAC (Harshman, 1970):

$$\mathbf{Z} \approx \sum_{k=1}^{K} w_k \circ h_k \circ q_k \tag{15}$$

where $\mathbf{C}_{rsp}$ is a core tensor, $\circ$ 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 student, task, and time, respectively (please refer to the articles (Kolda and Bader, 2009; Dunlavy et al., 2010) for more details).

To take into account the "student bias" and "task bias", the prediction function now becomes:

$$\hat{p}_{ui} = \mu + b_u + b_i + \left(\sum_{k=1}^{K} w_k h_k^T \Phi_k \right)_{u,i} \tag{16}$$

$$\Phi_k = \frac{\sum_{t=(T-T_{max}+1)}^{T} q_k(t)}{T_{max}} \tag{17}$$

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 5). 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)

$$\Phi_k = \frac{\sum_{t=(T-T_{max}+1)}^{T} q_k(t) e^{t \cdot \theta}}{T_{max}} \quad (18)$$

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

Dealing with cold-start problem

One of the challenging problem of collaborative filtering approaches like matrix factorization is to deal with the "new user" (new student) or "new item" (new task), e.g., those that are in the test set but not in the train set. We treat this problem with a simple strategy: providing the global average score for the new users or new items. Of course, more sophisticated methods can improve the prediction results (Preisach et al., 2010; Gantner et al., 2010) but we leave these solutions for future work.

Moreover, in the educational data mining scenario, the cold-start problem is not as harmful as in the e-commerce environment where the new users and new items appear every day or even hour. In educational environment, the new students and new tasks (or courses) appear only in every term/semester, and thus, the models need not to be re-trained continuously.

EMPIRICAL EVALUATION

An important problem in using recommender systems for predicting student performance is that which task to be considered as item. This section will discuss about this problem. At first, we will look in details 2 real world data sets which are collected from the Knowledge Discovery and Data Mining Challenge 2010 (pslcdatashop.web.cmu.edu/KDDCup).

These data sets, originally labeled "Algebra 2008-2009" and "Bridge to Algebra 2008-2009" will be denoted "Algebra" and "Bridge" for the remainder of this chapter. The Algebra (Bridge) data set has 23 (21) attributes, 8,918,054 (20,012,498) records for training, and 508,912 (756,386) records for testing. These data represent 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.

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*. Fur-

thermore, 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.

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.

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

$$\begin{aligned} \textit{student} &\mapsto \textit{user} \\ \textit{performance (correct first attempt)} &\mapsto \textit{rating} \end{aligned}$$

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.

Table 1 Mapping student performance data to user/item in recommender systems. Solving-step is a combination of problem hierarchy (PH), problem name (PN), problem group (PG) extracted from PN, step name (SN), and problem view (PV). Skill is also called knowledge component (KC). (*) items are used for the experiments

	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
PH, PN, SN	1,309,038	593,369
PH, PG, SN	848,218	188,001
PG, SN	776,155	155,808
PN, SN	1,254,897	566,843
SN	695,674	126,560
PN	188,368	52,754
PG	185,918	52,189
PH, PN	206,596	61,848
PH, PG	1,000	1,343
PH	165	186
PH, PN, PV	220,045	101,707
PH, PG, PV	3,203	5,537
PH, PV	780	1,526
Rating	**#Rating**	**#Rating**
Correct first attempt	8,918,054	20,012,498

For mapping the *item*, several options seemed to be available to us. From the official KDD Challenge dataset description it was immanent that an *item* was supposed to be the combination (concatenation) of *problem hierarchy* (PH), *problem name* (PN), *step name* (SN), and *problem view* (PV). Choosing PH-PN-SN-PV as the *item* had the drawback of incurring the new-item problem into our recommendation task: in the test sets, instances of PH-PN-SN-PV would occur that are unavailable in the training set, thus our models would not be able to learn much about them. Another problem with this approach is that it leads to huge sparsity and to a high number of new items on the test set. For instance, for the algebra dataset this configuration would lead to a total of 1,416,473 items (see Table 1). Since there are 8,918,054 examples on the training set for this dataset one could expect to see, on average, 6 observations per item (99.78% sparsity). Indeed, as shown in Thai-Nghe et al. (2010a) the different mappings yield different results.

Generally speaking, there are three important factors for the evaluation of the candidate items:

- the percentage of new users or new items in the test set
- the overall sparsity of the resulting matrix
- the percentage of missing values

Baselines and Model setting

Baselines: We use the global average as a baseline method, i.e. predicting the average of the target variable from the training set. The proposed methods are compared with other methods such as user average, biased-user-item (Koren, 2010). Please note that, as shown in Thai-Nghe et al. (2010a), matrix factorization already outperforms the traditional regression methods such as linear/logistic regression. Thus, in this chapter we have not considered these algorithms for the evaluation.
Hyper paramter setting: We do the hyper parameter search to determine the best hyper parameters for all methods, by optimizing the RMSE (Root mean squared error) on a holdout set. We will later report the hyper parameters for some typical methods (in Table 2). Please also note that we have not performed a significance test (t-test) because the real target variables of two data sets from KDD Challenge 2010 have not been published yet. In order to obtain the RMSE on the two data sets we have to submit the generated predictions to the KDD Challenge 2010 website. Thus, all the results reported in this study are the RMSE score from this website (it still opens for submission after the challenge). Of course, we could use an internal split (e.g. splitting the training set to sub-train and sub-test) but we wanted compared to the other KDD cup participants' approaches on the given datasets for better comparability (we will report later in the discussion section).

Implicitly Encoding Latent Factors Using Matrix Factorization

Figure 4 presents the comparison of root mean squared error for Algebra data set. In Figure 4a, we factorize on the student and the task, which is the solving-step. The result significantly outperforms any baseline method.

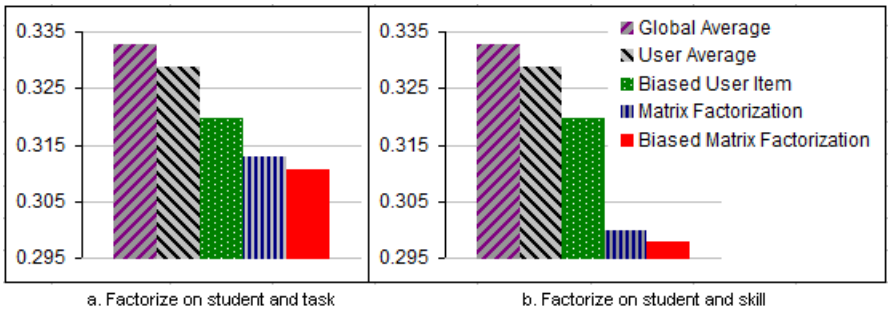


Fig. 4 RMSE results of matrix and biased matrix factorization which factorize on a) student and the solving-step (task) and b) student and the required skills in solving the step. Algebra data set.

Moreover, if we have more metadata, e.g., the required skills in solving the step, we see an additional improvement. Figure 4b clearly shows that 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 again improved compare to other methods and increases the gap between the baseline methods and the best factor model. These results somehow reflect that the proposed approach can deal with the "slip" and "uess" factors by implicitly encoding them as the latent factors of the model. The reason for the improvement of factorizing on the student/skill could be that, in this setting, there are less (tasks) items (8,197) and the same number of (performance) ratings, thus, on average, each (student) user has more ratings (8,918,054/8,197 $\approx$ 1,088 ratings/user), which means that the algorithms have more data to learn from.

Tensor Factorization for Exploring Temporal Effects

The previous section shows that the "slip" and "guess" factors could be implicitly encoded in the models. In this section, we present the result of taking into account the temporal information. Figure 5 describes the effect of the time on knowledge of the learners for both Algebra and Bridge data sets. 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), and 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 additional approaches to learn the skills. This

trend also reflects the educational factor that “the more the students learn, the better their performance gets”.

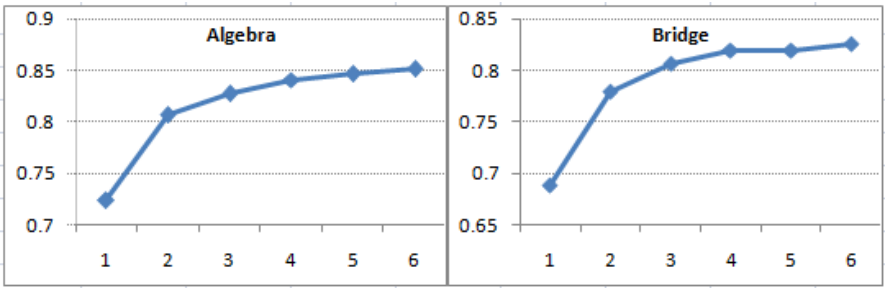


Fig. 5 The effect of the time on the knowledge of the learners. The x-axis presents the number of times the student learns the knowledge components. The y-axis is the probability of solving the problem correctly

Figure 6 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) on both data sets. The results of the proposed methods are also improved compared to those already presented. Compared with a matrix factorization which does not take the temporal effect into account, the tensor factorization methods perform better. 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).

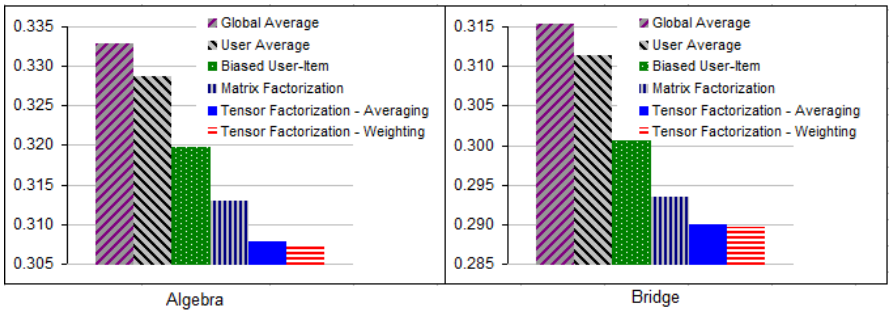


Fig. 6 RMSE of taken into account the temporal effect using tensor factorization which factorize on student, solving-step, and time

For referencing, we report the best hyper parameters found via cross validation and approximation of the running-time in Table 2, which produce the results in Figure 6. Please note that although the training time of TFW is high, e.g. ≈15 hours for

Bridge data set, in educational environment where the models need to be retrained on each term/semester/year, run times in this order of magnitude are not an issue, as we already discussed in the cold-start subsection.

Table 2 Hyperparameters and approximation of the running-time.

Method	Data set	Hyperparameters	Train (min.)	Test (sec.)
MF	Algebra	β=0.005, #iter=120, K=16, λ=0.015	16.83	0.15
Biased MF	Algebra	β=0.001, #iter=80, K=128, λ=0.0015	19.12	0.23
MF	Bridge	β=0.01, #iter=80, K=64, λ=0.015	40.15	0.34
TFA	Bridge	β=0.01, #iter=30, K=32, λ=0.015, T_{max}=8	296.25	2.84
TFW	Bridge	β=0.01, #iter=30, K=32, λ=0.015, T_{max}=8, θ= 0.4	904.40	22.56

DISCUSSION

To this end, one would raise the following question: "How do the delivered recommendations look like?" The delivered recommendations depend on the goal of the system. Let's discuss the recommendation for a single student:

- In case of rating prediction, which is the problem approached in this chapter, we are interested in student's performance for a given item. In our case, it means that we compute the expected performance on all tasks not solved by the student in the past. Then, we have several choices: we can provide the student with tasks she will be more likely to solve successfully or we can let the teacher decide about the right mix of tasks or simply present the prediction as a measure of estimated difficulty of that task for the student at hand.
- If we are most interested in those items the students is expected to have problems with and which we therefore want to present to him for further learning (item prediction is the recommender systems task at hand), we compute the expected score for each item the student has not tackled in the past. We sort these items according to the predicted score and deliver the student the top-k scored items as those have the highest likelihood of him failing this task.
- Furthermore, we can of course tackle classic problems such as providing the student with the top-k learning resources (courses, materials, tasks) he/she would be more likely interested in.

As shown above, there is a plethora of areas of application for recommender systems in learning environments. For instance, in the example of Figure 3, we could recommend to the students some similar exercises which are neither too hard nor too easy for them, thus, the system can help the student improve their knowledge by learning/doing similar tasks. Moreover, from the scenario of Figure 1, when the students solve a problem, we could also recommend them some similar problems which may help them consolidate and expand their knowledge. Another example

is the recommendation of similar grammar structures, vocabularies, or even a similar problem/section when the student is learning or doing exercises in an English course, etc.

Another important question could be raised is that "How good our approach is compared to the others competitors on KDD Challenge data?" The RMSE score of the best student teams is summarized from the leader-board as the following:

- First place: RMSE = 0.27295, National Taiwan University team (Yu et al., 2010)
- Second place: RMSE = 0.27659, Zach A. Pardos team (Pardos and Heffernan, 2010)
- Third place: RMSE = 0.28048, SCUT Data Mining team (Shen et al., 2010)
- *Our approach: RMSE = 0.29519 (not submitted)*
- Fourth place: RMSE = 0.29801, Y10 team

Please note that the focus of this work is not producing a system for the KDD Challenge but on getting the real datasets from an e-learning system 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 these together. On the other hand, factorization methods like those presented in this work 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 techniques are promising for the problem of predicting student performance.

One more issue should also be discussed is that "what does the presented RMSE reduction mean in practical? E.g., in helping the education managers, students..." As in the Netflix Prize (netflixprize.com) 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 as explicit 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 its 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.

FUTURE RESEARCH DIRECTIONS

An open issue is that recommender systems for educational purposes are more complex and challenging research direction (Drachsler et al., 2009) compared to the case of e-commerce in which recommender systems are widely being used. The reasons are that learners have specific learning goals that they want to achieve within a specified competence in a certain time, and that the preferred learning activities of learners might not be the pedagogically the most adequate (Tang and McCalla, 2004). As discussed in Santos and Boticario (2010), recommendations for technology enhanced learning scenarios have differences from those in other domains because recommendations in e-learning should be guided by educational objectives, and not only by the user's preferences. Thus, in future work, the proposed approach which uses student performance for recommendation would be promising to tackle the above problems since we can recommend the tasks for students based on their performance but not on their preferences.

Moreover, to improve the prediction results, one can apply more sophisticated methods to deal with the cold-start problems. Also, using ensemble methods on different models generated from matrix and tensor factorization can be investigated. Moreover, to take into account the sequential effect, instead of using averaging or weighting methods on the third mode of tensor, one could use forecasting techniques such as single exponential smoothing or Holt-Winter methods (Chatfield and Yar, 1988).

Another open issue is that each solving-step relates to one or many skills, thus, use multi-relational matrix factorization (Singh and Gordon, 2008; Lippert et al., 2008) could be developed to deal with this problem.

CONCLUSIONS

In this chapter, we have discussed on the problem of predicting student performance as well as personalized recommending tasks/exercises to students. We introduce state-of-the-art factorization techniques for predicting student performance. These techniques are useful in case of sparse data, and even in the case that we have no background knowledge about the students and the tasks. After introducing and formulating this problem in the context of recommender systems, we present factorization methods, which are currently known as the most effective recommendation approaches, by a formalized and detailed way that can be helpful for the researchers who come from educational domains.

Factorization techniques belong to the family of latent factor models, thus, they can implicitly take into account the important latent factors in predicting student performance such as "slip" (student knows the solution but making a mistake) and "guess" (student does not know the right solution but guessing correctly) as well as "student effect" (e.g. how clever/good the student is, in performing the task) and "task effect" (e.g. how hard/easy the task is). Moreover, as a natural fact, the knowl-

edge of the learners improves over the time, thus, we propose tensor factorization methods to take the temporal effect into account. Finally, some experimental results and discussions are provided to validate the proposed approach.

ACKNOWLEDGMENTS

The first author was funded by the "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.

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.

Chatfield, C. and Yar, M. (1988). Holt-winters forecasting: Some practical issues. *Special Issue: Statistical Forecasting and Decision-Making. Journal of the Royal Statistical Society. Series D (The Statistician)*, 37(2):129–140.

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.

Gantner, Z., Drumond, L., Freudenthaler, C., Rendle, S., and Schmidt-Thieme, L. (2010). Learning attribute-to-feature mappings for cold-start recommendations. In *Proceedings of the 10th IEEE International Conference on Data Mining (ICDM 2010)*. IEEE Computer Society.

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 *International 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.

Harshman, R. A. (1970). Foundations of the PARAFAC procedure: Models and conditions for an" explanatory" multi-modal factor analysis. *UCLA Working Papers in Phonetics*, 16(1):84.

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.

Koedinger, K., Baker, R., Cunningham, K., Skogsholm, A., Leber, B., and Stamper, J. (2010). A data repository for the edm community: The pslc datashop. In Romero, C., Ventura, S., Pechenizkiy, M., and Baker, R., editors, *Handbook of Educational Data Mining*, Lecture Notes in Computer Science. CRC Press.

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.

Lawrence, N. D. and Urtasun, R. (2009). Non-linear matrix factorization with gaussian processes. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 601–608, New York, NY, USA. ACM.

Lippert, C., Weber, S. H., Huang, Y., Tresp, V., Schubert, M., and Kriegel, H.-P. (2008). Relation-prediction in multi-relational domains using matrix-factorization. In *NIPS 2008 Workshop: Structured Input - Structured Output*.

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. winner - 2nd place. *KDD Cup 2010: Improving Cognitive Models with Educational Data Mining*.

Pilászy, I. and Tikk, D. (2009). Recommending new movies: even a few ratings are more valuable than metadata. In *Proceedings of the third ACM conference on Recommender systems*, RecSys '09, pages 93–100, New York, NY, USA. ACM.

Preisach, C., Marinho, L. B., and Schmidt-Thieme, L. (2010). Semi-supervised tag recommendation - using untagged resources to mitigate cold-start problems. In *14th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2010)*, volume 6118 of *Lecture Notes in Computer Science*, pages 348–357. Springer.

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.

Santos, O. C. and Boticario, J. G. (2010). Modeling recommendations for the educational domain. *Procedia Computer Science*, 1(2):2793–2800. Proceedings of the 1st Workshop on Recommender Systems for Technology Enhanced Learning (RecSysTEL 2010).

Shen, Y., Chen, Q., Fang, M., Yang, Q., Wu, T., Zheng, L., and Cai, Z. (2010). Predicting student performance: A solution for the kdd cup 2010 challenge. *KDD Cup 2010: Improving Cognitive Models with Educational Data Mining*.

Singh, A. P. and Gordon, G. J. (2008). Relational learning via collective matrix factorization. In *Proceeding of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2008)*, KDD '08, pages 650–658, New York, NY, USA. ACM.

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

Takács, G., Pilászy, I., Németh, B., and Tikk, D. (2009). Scalable collaborative filtering approaches for large recommender systems (special topic on mining and learning with graphs and relations). *Machine Learning Research*, 10:623–656.

Tang, T. and McCalla, G. (2004). Beyond learners interest: Personalized paper recommendation based on their pedagogical features for an e-learning system. In

Zhang, C., W. Guesgen, H., and Yeap, W. K., editors, *PRICAI 2004: Trends in Artificial Intelligence*, volume 3157 of *Lecture Notes in Computer Science*, pages 301–310. Springer Berlin / Heidelberg.

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. (2010a). 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., Gantner, Z., and Schmidt-Thieme, L. (2010b). Cost-sensitive learning methods for imbalanced data. *in Proceeding of IEEE International Joint Conference on Neural Networks (IJCNN'10)*.

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. winner - 3rd place. *KDD Cup 2010: Improving Cognitive Models with Educational Data Mining*.

Tucker, L. (1966). Some mathematical notes on three-mode factor analysis. *Psychometrika*, pages 279–311.

Yu, H.-F., Lo, H.-Y., Hsieh, H.-P., Lou, J.-K., G.McKenzie, T., Chou, J.-W., Chung, P.-H., Ho, C.-H., Chang, C.-F., Wei, Y.-H., Weng, J.-Y., Yan, E.-S., Chang, C.-W., Kuo, T.-T., Lo, Y.-C., Chang, P. T., Po, C., Wang, C.-Y., Huang, Y.-H., Hung, C.-W., Ruan, Y.-X., Lin, Y.-S., de Lin, S., Lin, H.-T., and Lin, C.-J. (2010). Feature engineering and classifier ensemble for kdd cup 2010. winner - 1st place. *KDD Cup 2010: Improving Cognitive Models with Educational Data Mining*.

ADDITIONAL READING

- Baker, R. S. and Yacef, K. (2009). The state of educational data mining in 2009: A review and future visions. Journal of Educational Data Mining (JEDM), 1(1):3-17.
- Castro, F., Vellido, A., Nebot, n., and Mugica, F. (2007). Applying data mining techniques to e-learning problems. In Kacprzyk, J., Jain, L., Tedman, R., and Tedman, D., editors, Evolution of Teaching and Learning Paradigms in Intelligent Environment, volume 62 of Studies in Computational Intelligence, pages 183-221. Springer Berlin Heidelberg.

- Cichocki, A., Mrup, M., Smaragdis, P., Wang, W., and Zdunek, R. (2008). Editorial: advances in nonnegative matrix and tensor factorization. Intelligent Neuroscience, 2008:2:1-2:3.
- Cichocki, A., Zdunek, R., Phan, A. H., and Amari, S.-i. (2009). Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-way Data Analysis and Blind Source Separation. Wiley Publishing.
- Duda, R. O., Hart, P. E., and Stork, D. G. (2001). Pattern Classification (2nd Edition). Wiley-Interscience, 2 edition.
- Ghauth, K. I. and Abdullah, N. A. (2010). Measuring learner's performance in e-learning recommender systems. Australasian Journal of Educational Technology, 26(6):764-774.
- Gunawardana, A. and Shani, G. (2009). A survey of accuracy evaluation metrics of recommendation tasks. Journal of Machine Learning Research, 10:2935-2962.
- Herlocker, J. L., Konstan, J. A., Terveen, L. G., and Riedl, J. T. (2004). Evaluating collaborative filtering recommender systems. ACM Transactions on Information Systems, 22(1):5-53.
- Koren, Y. (2010). Factor in the neighbors: Scalable and accurate collaborative filtering. ACM Transaction on Knowledge Discovery Data, 4(1):1-24.
- Kumar, V., Nesbit, J., Winne, P., Hadwin, A., Jamieson-Noel, D., and Han, K. (2007). Quality rating and recommendation of learning objects. In Jain, L., Wu, X., and Pierre, S., editors, E-Learning Networked Environments and Architectures, Advanced Information and Knowledge Processing, pages 337-373. Springer London.
- Luo, J., Dong, F., Cao, J., and Song, A. (2010). A context-aware personalized resource recommendation for pervasive learning. Cluster Computing, Springer-Netherlands, 13(2):213-239.
- Mairal, J., Bach, F., Ponce, J., and Sapiro, G. (2010). Online learning for matrix factorization and sparse coding. Journal of Machine Learning Research, 11:19-60.
- Manouselis, N., Vuorikari, R., and Van Assche, F. (2010). Collaborative recommendation of e-learning resources: an experimental investigation. Journal of Computer Assisted Learning, 26(4):227-242.
- Markowska-Kaczmar, U., Kwasnicka, H., and Paradowski, M. (2010). Intelligent techniques in personalization of learning in e-learning systems. In Xhafa, F., Caball, S., Abraham, A., Daradoumis, T., and Juan Perez, A., editors, Computational Intelligence for Technology Enhanced Learning, volume 273 of Studies in Computational Intelligence, pages 1-23. Springer Berlin / Heidelberg.
- 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.
- Morup, M. (2011). Applications of tensor (multiway array) factorizations and decompositions in data mining. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 1(1):24-40.
- Nishino, K., Shimoda, T., Iribe, Y., Mizuno, S., Aoki, K., and Fukumura, Y. (2010). Predicting e-learning course adaptability and changes in learning prefer-

ences after taking e-learning courses. In Setchi, R., Jordanov, I., Howlett, R., and Jain, L., editors, Knowledge-Based and Intelligent Information and Engineering Systems, volume 6277 of Lecture Notes in Computer Science, pages 143-152. Springer Berlin / Heidelberg.
- Rendle, S., editor (2010). Context-aware Ranking with Factorization Models. Springer.
- Ricci, F., Rokach, L., Shapira, B., and Kantor, P. B., editors (2011). Recommender Systems Handbook. Springer.
- Romero, C., Ventura, S., Pechenizkiy, M., and Baker, R. S. (2010). Handbook of Educational Data Mining. Chapman and Hall/CRC, Data Mining and Knowledge Discovery Series.
- Shen, Y., Chen, Q., Fang, M., Yang, Q., Wu, T., Zheng, L., and Cai, Z. (2010). Predicting student performance: A solution for the kdd cup 2010 challenge. KDD Cup 2010: Improving Cognitive Models with Educational Data Mining.
- Skillicorn, D. (2007). Understanding Complex Datasets: Data Mining with Matrix Decompositions. Chapman and Hall/CRC, Data Mining and Knowledge Discovery Series.
- Su, X. and Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques. Advance in Artificial Intelligence, 2009.
- Takács, G., Pilászy, I., Németh, B. and Tikk, D. (2009). Scalable collaborative filtering approaches for large recommender systems (special topic on mining and learning with graphs and relations). Machine Learning Research, 10:623-656.
- Tan, P.-N., Steinbach, M., and Kumar, V. (2005). Introduction to Data Mining, (First Edition). Addison-Wesley Longman Publishing Co., Inc., Boston.
- Tang, T. and McCalla, G. (2004). Beyond learners interest: Personalized paper recommendation based on their pedagogical features for an e-learning system. In Zhang, C., W. Guesgen, H., and Yeap, W. K., editors, PRICAI 2004: Trends in Artificial Intelligence, volume 3157 of Lecture Notes in Computer Science, pages 301-310. Springer Berlin / Heidelberg.
- Yu, Z., Nakamura, Y., Jang, S., Kajita, S., and Mase, K. (2007). Ontology-based semantic recommendation for context-aware e-learning. In Indulska, J., Ma, J., Yang, L., Ungerer, T., and Cao, J., editors, Ubiquitous Intelligence and Computing, volume 4611 of Lecture Notes in Computer Science, pages 898-907. Springer Berlin / Heidelberg.

KEY TERMS & DEFINITIONS

- Collaborative Filtering (CF): CF filters the information or patterns based on the collaboration among users, items, etc. CF generates the recommendations based on the ideas that ”similar users like similar things”
- Cold-start problem: A user/item has not been known by the models (new user/item)
- Computer-Aided tutoring system (CATS): CATS is any computer system that provides direct customized instruction or feedback to students (Wikipedia.org).

- Educational data mining (EDM): EDM is concerned with developing methods for exploring data that come from educational settings (Educationaldatamining.org).
- Matrix factorization (MF): MF is an approximation of a matrix by two (or more) low-rank matrices.
- Predicting student performance (PSP): PSP is the problem of predicting student's ability (e.g., estimated by a score metric) in solving the tasks when interacting with the tutoring system.
- Rating prediction: The prediction of the rating values (scores) for items (e.g. movies, books) that a user may like.
- Tensor: Tensor is a three (or more) dimensional matrix
- Tensor factorization (TF): TF is a generalization of MF.