Noname manuscript No.
(will be inserted by the editor)

Constrained Sparse Optimization for Tensor-Based Modeling Of Student Learning Dynamics

Alireza Farasat · Alexander G. Nikolaev

Received: date / Accepted: date

Abstract This paper advances the state of the art in Latent Factor Analysis-based student modeling and offers new methods for tackling the student performance prediction problem, using Probabilistic Tensor Factorization. The paper introduces new constrained tensor factorization problems that enable one to distill knowledge acquisition patterns from sparse observations of student performance in problem solving. This is accomplished by estimating students' conceptual material understanding levels, while recognizing both the temporal dynamics of knowledge acquisition and the differences in students' learning abilities. The introduced Homogeneous Student Model (HSM) accounts for knowledge gains over time by using a set of non-parametric constraints, and is useful when students can be assumed to progress as a group, e.g., a class. The Personalized Student Model (PSM) recognizes and reveals the differences in the learning curves of different students. The model-based Likelihood Maximization formulations resulting in non-convex, constrained optimization problems are solved using the Block Coordinate Descent (BCD) and Alternating Direction Method of Multipliers (ADMM). The paper compares the performance of the HSM, PSM and basic Static Student Model, all parameterized on the same training datasets, by measuring their prediction accuracy on the same testing datasets, achieved with both BDC and ADMM. The experiments showcase particularly robust prediction results obtained with the PSM using ADMM.

Keywords Probabilistic tensor factorization · non-convex optimization · Student modeling · Intelligent tutoring systems

1 Introduction

Machine learning techniques have been extensively applied in the education domain since the first conference on educational data mining in 2008 [38]. In this domain, such techniques and modeling tools as Logistic Regression, Decision Trees, Bayesian Networks, etc., are used to attack a variety of problems including student performance prediction, grouping students by ability, and directing students to appropriate learning materials and courses [2, 22, 30].

Student performance prediction is one of the most actively researched educational data mining tasks addressed with the help of machine learning tools [7, 41, 43]. The goal of student performance prediction lies in evaluating how much knowledge students have and how they can be expected to perform on

Alireza Farasat
University at Buffalo
E-mail: afarasat@buffalo.edu

Alexander G. Nikolaev
University at Buffalo
E-mail: anikolae@buffalo.edu

given assessment tasks. Observe that, since knowledge levels and skills tend to improve with time, the sequence in which students take on the assessment tasks is an important determinant in the inference problem formulations and outputs [39]. The KDD Cup 2010 featured student performance prediction as its problem of choice, challenging scientists to infer how students learn and adapt to new problems, and what knowledge levels/dimensions are required for correctly solving a given problem/question, based on observed student performance data. Indeed, accurate student models and performance predictors can save millions of hours of students' time and effort [6]. Developing powerful predictive models can also improve, or even lead to changing the format of, the current standardized tests such as SAT and GRE. Such efforts are primed to accelerate and personalize learning in our society, and ameliorate the stress associated with it [11, 39].

The number of efforts undertaken to improve student performance prediction demonstrate the importance of modeling in educational systems research. Most of the existing modeling studies exploit traditional classification/regression techniques, which fail to deal with the sparsity of observed data. Some very recent studies, however, report on the successes achieved through the use of matrix factorization techniques, previously adopted in recommender system research, for predicting student performance [27, 37, 39–41]. Although learning and problem-solving are multifaceted activities, which are based on complex cognitive processes fundamentally different from consumer logic, the collaborative filtering models were found capable of inferring latent dimensions of knowledge, and successfully exploit such inferred dimensions for making predictions. Being effective in particular in those settings where little observational data are available, the matrix factorization-based methods became the forefront of the state-of-the-art in student modeling [38, 39].

It is important to stress that the main driver behind the efforts toward predicting student performance is not the need to forecast students' future successes or failures on assignments, but the desired ability to identify an optimal way to teach students. Given this objective, a useful student model should account for the fact that learning happens over time – e.g., during students' interactions with course content, – and should be able to predict how the changes in the order of these interactions can affect learning outcomes. Further, it is also desirable to discover and explain *why* and *how* students acquire knowledge, and hence, models with explicit parametric knowledge space representations are particularly valuable.

This paper offers a student modeling framework, based on constrained *tensor* factorization, that meets the above outlined challenges. The paper presents three models that are able to handle highly sparse data and estimate students' conceptual material understanding levels, while recognizing both the temporal dynamics of knowledge acquisition and the differences in students' learning abilities. Two optimization algorithms are tailored to deal with the likelihood maximization problems for the estimation of the models' parameters.

Notably, the models built within the presented framework are non-parametric with respect to the interaction between students and learning objects. In other words, we do not have to assume that knowledge gain is necessarily a direct consequence of such interaction. This allows the presented models to be informative about which sequence of problems it is best for each student to take on *at each point of time* in their learning path, while remaining simple. To elaborate on this, note that simple models that explicitly parameterize the interaction between learners and learning objects may offer higher predictive accuracy but may fail when used for prescriptive purposes. Indeed, under the assumption that any given student gains the same amount of knowledge from the same learning object at any point of their journey into a new topic, such models would prescribe everyone to consume the learning materials in the decreasing order of the knowledge increments that these materials offer, irrespective of whether the learners are capable of consuming them or not. Our models do not have this drawback.

The rest of the paper is organized as follows. Section 2 reviews the state-of-the-art in student modeling. The new student models are introduced in Section 3. Section 4 presents the optimization

problem formulations along with gradient-based algorithms to solve them. Section 5 sets up computational experiments to evaluate the performance of the models. Section 6 reports the results. Section 7 concludes the paper.

2 An Overview of Advances in Predictive Student Modeling

Student performance prediction has been a topic of research in traditional education systems for many years. The research efforts in this domain have focused on several different aspects of this challenging problem. The works rooted in Item Response Theory (IRT) estimate the response that a student can be expected to provide to a particular item, as the quality of the student's answer to a problem/question [5,35]. Cen *et al.* use Latent Factor Analysis (LFA) as an approach to represent responses as a function of multiple skills that students may have fully or partially mastered, with the probability of mastering a skill being determined by the knowledge accumulation model [6]; this LFA model is an application of Item Response Theory [37]. Performance Factors Analysis (PFA), due to Pavlik *et al.*, is another student modeling approach that offers improvements over LFA. Similar to LFA, PFA employs logistic regression to compute the probability that a student correctly solves a given problem, given the counts of the student's prior successes and failures [32].

The more advanced student prediction models rely on matrix factorization methods [3,21,31]. Here, student performance prediction is treated as rating prediction – exactly as in recommender systems, with students, questions, and solution grades replacing users, items, and rating, respectively [40]. It has been shown that the matrix factorization techniques surpass other methods such as those based on global averages, user averages, item averages, user-item-baseline concept [20], regularized logistic regression [19] and Bayesian Knowledge Tracing models [1] (see [37] for more details).

Some variants of matrix factorization-based models are specifically tailored for student modeling [37], e.g., by explicitly incorporating the human element in the modeling – explicitly assuming that humans tend to forget things and that they may guess at correct answers into model formulations, – and also, by detecting similarities between different students. Further, accounting for different levels of skill mastery and the temporal/sequential nature of the observational data in knowledge acquisition pattern forecasting also improves the accuracy in student performance prediction [37].

A more recently emerging research stream aims at developing computationally effective matrix factorization methods [18,25], capable of precisely imputing any missing elements of a matrix comprising noisy or potentially corrupted observations [36]. The SPARse Factor Analysis (SPARFA) framework [27] represents the current state-of-the-art in these efforts; it uses probabilistic matrix factorization and outperforms other factor analysis methods in predicting student performance [24]. In SPARFA, the probability that a student answers a given question or solves a problem correctly is determined by the student's understanding of a set of underlying course concepts, on the concepts required to solve the problem, and on the problem's intrinsic difficulty [27]. An extension to the basic SPARFA model is a Kalman filter-based model that traces knowledge over time as students get exposed to learning resources such as textbook sections, lecture videos, etc. [25]. Also notable are some other machine learning-based techniques used for student modeling, in particular those based on Recurrent Neural Network (RNN) models with sigmoid units, and on the Long Short Term Memory model, developed as part of the inference approach called Deep Knowledge Tracing [33].

Reviewing the research gaps yet to be filled, the key deficiency of the vast majority of the above mentioned models – including SPARFA and other matrix factorization-based models – lies in their inability to account for the fact that learning takes place over time, in an irregular way, *while* the observations that inform the inference are being collected. The exceptions include the original Bayesian Knowledge Tracing model and its extensions [1,14,44], however, these models cannot handle multiple hidden and

interrelated knowledge component at once. Some recently proposed models – e.g., SPARFA-Trace [25] and RRN-based models – also work with longitudinal learning data, however, they operate under the assumption that knowledge accumulation occurs at distinct and known time instants – at the fixed moments when students interact with the learning materials. SPARFA-Trace, however, turns out to be computationally efficient only under the assumption that the learning materials of any difficulty impact students the same regardless of the timing that the students interact with them. As mentioned earlier, this significantly limits such models' ability to inform the optimal sequencing of learning materials. Further, many of the existing models do not feature well-interpretable parameters, which could allow instructors to effectively monitor student progress and understand its reasons. If the latter were possible, one could employ student models for the purpose of personalizing learning experiences.

3 Sparse Tensor Factorization for Student Modeling

Matrix and tensor decomposition is at the core of machine learning tools in many application areas, including recommender system design [12], feature extraction [10], signal processing [23], social tagging systems [34], computer vision and graphics, missing value estimation [29], and blind source separation [16], among others. This section provides a detailed description of the proposed tensor factorization-based approach to student modeling, along with explanations of the key concepts that the models' logic relies on; for more details of the mathematical properties of tensor factorization, see, e.g., [9, 16, 29]. An advantage of tensor-based modeling, used broadly in recommender system research, is its ability to incorporate multiple features, and importantly, time as the variables explaining observed user-item interactions [12]. This latter trait becomes particularly useful when one attempts to make inference from the observations of student performance, under the assumption that knowledge acquisition and skill development occur *while* students are being observed.

3.1 Tensor-based Modeling Approach: Preliminaries and Notations

Being an extension of the concept of *matrix*, an N-way *tensor* is a mapping from a linear space to another space with N distinct dimensions. In this paper, tensors, matrices and vectors are denoted with calligraphic capital letters (e.g., $\mathcal{Y} \in R^{I_1,\ldots,I_N}$), bold capital letters (e.g., $\mathbf{W}$) and bold lowercase letters (e.g., $\mathbf{d}$), respectively. For tensor $\mathcal{Y} \in R^{I_1,I_2,I_3}$ of size $I_1 \times I_2 \times I_3$, a compact format $\mathcal{Y} = [\mathcal{Y}_{ijk}]$ is used, where $\mathcal{Y}_{ijk}$ is an element of $\mathcal{Y}$ such that $i \in I_1$, $j \in I_2$ and $k \in I_3$.

Let $\mathcal{Y} \in \{0,1\}^{Q,N,T}$ denote a binary-valued data set of student performance observations of size $Q \times N \times T$, with entry $\mathcal{Y}_{ijt} = 1(0)$ signaling that student $j = 1,\ldots,Q$ answered question $i = 1,\ldots,N$ at time $t = 1,\ldots,T$ correctly(incorrectly); here, Q, N and T are the number of students, the number of questions and the number of time slots (e.g., weeks, days or hours) over which the data were collected, respectively. The sparsity of tensor $\mathcal{Y}$ depends on the number of available questions Q and the level of activity of each student in taking on some of these questions. It is typical to expect that many questions will be answered only by a small subset of students, resulting in incomplete data. Similar to SPARFA and other ITS model assumptions, it is assumed that a certain number K of latent knowledge components (KCs) are required to gain mastery of the subject matter that the students are exposed to. It is worth nothing that K is taken to be significantly smaller than Q and N (i.e., $K \ll N, Q$) [27]. The positive real value d_i reflects the difficulty level of question $i = 1,\ldots,Q$, to differentiate the questions in this respect. It is also reasonable to assume that students may have aptitude (learning ability): the positive real value θ_j quantifies the aptitude of student $j = 1,\ldots,N$. The vectors $\mathbf{d}$ and $\boldsymbol{\theta}$ contain the difficulty levels of all the questions and the aptitude levels of all the students, respectively. As in [27], W_{ki} denotes

how necessary the expertise in KC k is for solving question i: $W_{ki} = 0$ means that answering question i does not require KC k. The level of knowledge that student j has in KC k is denoted by C_{kj}. Finally, V_{kt} represents the relative level that learners tend to acquire in KC k by time t. In summary, matrix $\mathbf{W} = [\mathbf{w_1}\ \mathbf{w_2}\ \ldots\ \mathbf{w_Q}]$ relates questions to KCs, matrix $\mathbf{C} = [\mathbf{c_1}\ \mathbf{c_2}\ \ldots\ \mathbf{c_N}]$ relates students to KCs, and matrix $\mathbf{V} = [\mathbf{v_1}\ \mathbf{v_2}\ \ldots\ \mathbf{v_T}]$ relates KCs to time. These assumptions and notations will all be used below in Section 3.2, and with minor revisions, in Section 3.3.

3.2 Static and Homogeneous Student Models

An extension to Matrix Factorization (MF), Tensor Factorization adds time as another dimension explicitly accounted for in monitoring and predicting student performance. The goal of tensor factorization is to perform a higher order singular value decomposition (using the low-rank approximation logic) to build a predictive model that best fits the observed student performance outcomes $\mathcal{Y}$. In other words, tensor factorization is used to construct an approximation tensor, $\mathcal{T}$, such that $\mathcal{Y} \approx \mathcal{T}$ in those tensor elements that correspond to students' performance on the questions they actually took; once the factorization is complete, $\mathcal{T}$ also returns the sought-after estimates of the students' performance on the questions they have not yet taken. Several decomposition techniques can be employed to this end. One straightforward way to extend the concept of SVD to higher orders is to add another factor to the low-rank approximation model, in a method known as Canonical Polyadic decomposition [12]. For a latent factor model with three dimensions – students, questions and time, – the tensor reconstruction can be achieved as

$$\mathcal{T} = \sum_{k=1}^{K} \psi_k \mathbf{w_k} \otimes \mathbf{c_k} \otimes \mathbf{v_k},$$

with $\otimes$ denoting the outer product of vectors, and $\boldsymbol{\psi}$ denoting a vector of size K with elements $\psi_1 \geq \ldots \geq \psi_K$ (see Figure 1). It is safe to say that the decomposition relies just on $\mathbf{W}, \mathbf{C}$ and $\mathbf{V}$, since $\boldsymbol{\psi}$ can be subsumed by these tensors, and hence, can be excluded from the formulation.

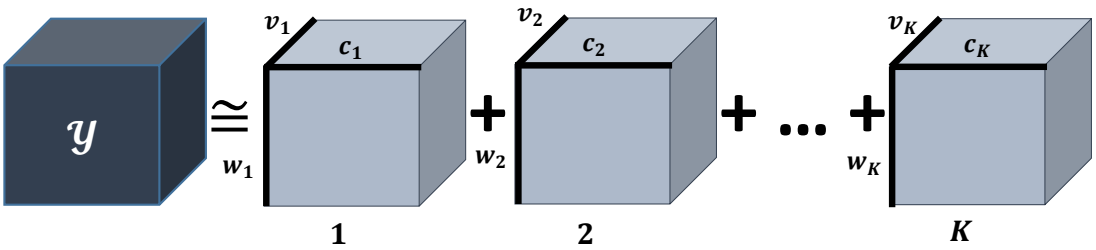

Fig. 1: Tensor Factorization Model based on CP decomposition.

A common issue with decomposition models, even 2-D models, is that multiple different decompositions may turn out to be equally good solutions to the optimization problem in the form $min\ \|\mathcal{Y} - \mathcal{T}\|_F$, unless this problem is well-posed [12]. In this work, in order to ensure that our tensor factorization problem is well-posed (see a discussion on this in [25]), it is assumed that only a limited number of knowledge components K – dimensions of the knowledge space – are required to represent a scientific subject area covered in a given course: i.e., $K \ll Q, N, T$.

Following the conventional probabilistic student modeling logic, we posit that there exists a distribution that describes the ability of each learner to perform well on a given task: the probability that learner j will answer question i at time t correctly depends on five factors: (i) the learner's knowledge represented in the space of latent, abstract concepts, (ii) the question's association with these concepts, captured as

weights, (iii) the intrinsic difficulty of the question, (iv) the learner's progress in acquiring knowledge by the time the question is taken, and (v) the learner's aptitude, i.e., the ability to apply the knowledge they have for solving new problems.

A model with interpretable parameters, corresponding to the above factors, is then developed to predict the performance of any student j on any question i at any time t, with 1 representing a correct response and 0 an incorrect response. Per the model, the reconstructed tensor is obtained as

$$\mathcal{T}_{ijt} = \sum_{k=1}^{K} W_{ki} C_{kj} V_{kt} - d_i + \theta_j \,, \quad \forall\, i = 1, \ldots, Q,\ j = 1, \ldots, N,\ t = 1, \ldots, T, \tag{1}$$

where d_i, and θ_j denote the difficulty of question i and aptitude of student j, respectively. Per the model, the response of student j taking on question i at time t follows a Bernoulli distribution,

$$\mathcal{Y}_{ijt} \sim Ber(\Phi(\mathcal{T}_{ijt})). \tag{2}$$

The parameter of the Bernoulli distribution in (2), is a *logit* function defined as

$$\Phi(x) = \frac{1}{1 + e^{-x}}.$$

Given $\mathbf{w_i}$, $\mathbf{c_j}$, $\mathbf{v_t}$, d_i and θ_j, the probability of observing response $\mathcal{Y}_{ijt}$ when student j takes on question i at time t can be expressed as

$$P(\mathcal{Y}_{ijt} | \mathbf{w_i}, \mathbf{c_j}, \mathbf{v_t}, d_i, \theta_j) = \Phi(\mathcal{T}_{ijt})^{\mathcal{Y}_{ijt}} [1 - \Phi(\mathcal{T}_{ijt})]^{1 - \mathcal{Y}_{ijt}}. \tag{3}$$

Let $\Omega_{obs} \subseteq \{1, \ldots, N\} \times \{1, \ldots, Q\} \times \{1, \ldots, T\}$ denote the set of given observations of students' performance in problem-solving. One can then use the Maximum Likelihood Estimation (MLE) approach to estimate the parameters in (1-3). As discussed in [26], care is required in formulating this parameter estimation problem, to avoid over-fitting and preserve interpretability and identifiability of matrices $\mathbf{W}$, $\mathbf{C}$ and $\mathbf{V}$. The assumptions below are in order to cope with such potential issues [27].

Low-rank Property: In traditional matrix factorization relying on SVD method, the hidden variable $K = min(M, N)$ is the rank of the SVD. However, in practice, a truncated SVD with $K < min(M, N)$ provides a reasonable approximation to the original matrix. In the educational data mining domain, K represents the number of knowledge components (concepts) that students are to learn in a given course. Indeed, it can be expected that the number of such knowledge components is significantly smaller than the number of students or questions.

Sparsity: Creating a question which tests one's knowledge regarding all or most of the concepts, covered in a course, at once is extremely difficult; therefore, one can postulate that $\mathbf{W}$ should contain many zeros, i.e., that it is sparse. This assumption remarkably increases the tractability of the corresponding MLE problem.

Non-negativity: Interpretability is a desired structural property of many machine learning models. With many pre-existing student models suffering from the lack of interpretability (as discussed in Section 2), it is desirable to tackle this issue in tensor factorization-based modeling. This can be achieved by restricting the model parameters to the positive space.

Under these assumptions, the MLE problem is expanded to include a set of constraints, to ensure that it is well-defined:

$$(P1): \max_{\mathbf{W},\mathbf{C},\mathbf{V},\mathbf{d},\boldsymbol{\theta}} \sum_{(i,j,t)\in\Omega_{obs}} log(P(\mathcal{Y}_{ijt}|\mathbf{w_i},\mathbf{c_j},\mathbf{v_t},d_i,\theta_j))$$

$s.t.$

$$\|\mathbf{w}_i\|_1 \leq \delta\,,\ \forall\, i=1,\ldots,Q, \tag{4}$$

$$\|\mathbf{w}_i\|_2 \leq \beta\,,\ \forall\, i=1,\ldots,Q, \tag{5}$$

$$\|\mathbf{C}\|_F = \xi\,, \tag{6}$$

$$\|\mathbf{V}\|_F = \eta\,, \tag{7}$$

$$W_{ki}, C_{kj}, V_{kt} \geq 0\,,\ \forall i=1,\ldots,Q,\ K=1,\ldots,K,\ j=1,\ldots,N,\ t=1,\ldots,T. \tag{8}$$

To meet the sparsity assumption, an upper bound is imposed on the number of non-zero elements in $\mathbf{W}$. One way to do this is to have $\|\mathbf{w_i}\|_0 \leq \delta$, where $\|.\|_0$ returns the count of non-zero entries in a vector; however, $\|.\|_0$ is non-differentiable, making our problem a hard combinatorial optimization problem. To mitigate this issue, we relax this sparsity-enforcing constraint by using the ℓ-1 norm in (4). Although the ℓ-1 norm still makes the problem's objective function non-smooth, there exist rather efficient optimization techniques (to be discussed later) that come as a remedy. The ℓ-2 norm-based constraint (5) on $\mathbf{W}$ prevents the elements of $\mathbf{W}$ from growing unbounded, and helps overcome the convergence issues [27]. Finally, using Frobenius norm, we suppress the arbitrary scaling of $\mathbf{C}$ and $\mathbf{V}$ with constraints (6-7). The constraints in (8) ensure the assumed non-negativity.

Problem $(P1)$ parameterizes the Static Student Model (SSM), which adds time as another dimension in conventional matrix factorization-based student modeling. This model is most basic, since it does not assume/imply any continuity in the knowledge acquisition dynamics, leaving much room for overfitting in the model parameterization stage.

In order to tackle this overfitting issue, and arrive at a more accurate and useful model, one can expand the constraint set in $(P1)$. Recognizing the fact that students gradually improve their knowledge over time, it is reasonable to assume – and require – that the inferred learning curves for the modeled students be increasing, or at least non-decreasing, functions in t.

Note that, while the interpretation of $\mathbf{V}$ alone is not straightforward, the product $\mathbf{C}^T\mathbf{V}$ gives a matrix that contains the knowledge levels of all the students across time. Thus, under the assumption that the knowledge attained by a student group should not deteriorate (a very reasonable assumption indeed – as time passes, the students tend to get better and better at handling the course material), one expects to have

$$\sum_{k=1}^{K} C_{kj}V_{kt+1} - \sum_{k=1}^{K} C_{kj}V_{kt} \geq -\epsilon\,,\ \ \forall\, j=1,\ldots,N,\ t=1,\ldots,T. \tag{9}$$

In (9), the small positive scalar ϵ helps account for a (temporary) loss of knowledge/skills that students might experience by forgetting some of the course material, e.g., during a Spring break or due to a class cancellation.

Problem $(P1)$ with constraints (4-9) parameterizes the Homogeneous Student Model (HSM). It is called homogeneous because it has students gaining knowledge simultaneously in time, each following the same pattern. More specifically, while the rate of the knowledge gain may vary, the students are assumed to learn as a group. This assumption, characteristic of HSM inference, is not unreasonable: indeed, students attend lectures together, take scheduled quizzes and exams together, and hence, are likely to read the same book chapters and improve their skills to about the same (relative) levels over about the same periods of time.

3.3 Personalized Student Model

With the HSM having a solid motivation behind it, one can still think of situations where significant differences may exist in how individual students approach learning: some may study continuously and get better more gradually, while others may study in spurts, e.g., in preparation for in-class tests. Further, in online courses, each student is likely to choose to learn at their own pace.

This section offers another tensor factorization-based model – Personalized student Model (PSM) – that is capable of capturing differences in different students' learning patterns. The cornerstone idea of PSM lies in tracking each student's progression separately, and focus on the accurate inference of questions' characteristics, while obtaining individual learning curves almost as a by-product.

Let tensor $\mathcal{X}$ of size $Q \times N \times T$ store the levels of knowledge that students have in the respective knowledge components at different time points considered for inference. We then factorize $\mathcal{Y}$ so that its approximation can be obtained bymultiplying $\mathbf{W}$ and $\mathcal{X}$. Figure 2 illustrates how this logic is applied to set up an inference problem; $\mathbf{W}$ captures the relations between questions and knowledge components, and $\mathcal{X}$ relates students, knowledge components and time.

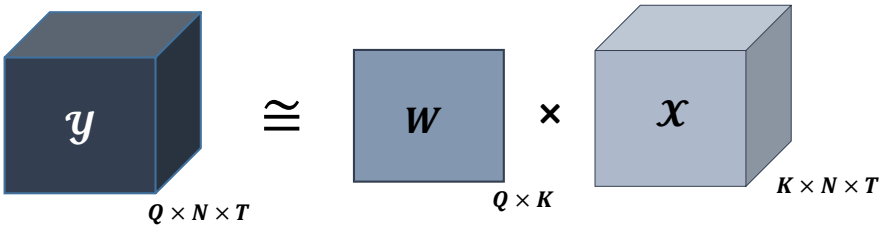

Fig. 2: Tensor Factorization Model based on multiplication of $\mathbf{W}$ and $\mathbf{X}$.

In PSM, one has

$$\mathcal{T}'_{ijt} = \sum_{k=1}^{K} W_{ik}\mathcal{X}_{kjt} - d_i + \theta_j\,,\ \forall\, i = 1,\dots,Q,\ j = 1,\dots,N,\ t = 1,\dots,T, \tag{10}$$

$$\mathcal{Y}_{ijt} \sim Ber(\Phi(\mathcal{T}'_{ijt})),$$

$$P(\mathcal{Y}_{ijt}|\mathbf{w_i}, \mathcal{X}_{:jt}, d_i, \theta_j) = \Phi(\mathcal{T}'_{ijt})^{\mathcal{Y}_{ijt}}[1 - \Phi(\mathcal{T}'_{ijt})]^{1-\mathcal{Y}_{ijt}}, \tag{11}$$

and the estimation of the parameters in (10-11) is accomplished by solving the following optimization problem with several sets of regularization constraints:

$$(P2): \max_{\mathbf{W},\mathcal{X},v,\boldsymbol{\theta}} \sum_{(i,j,t)\in\Omega_{obs}} log(P(\mathcal{Y}_{ijt}|\mathbf{w_i}, \mathcal{X}_{:jt}, d_i, \theta_j))$$

$s.t.$

$$\|\mathbf{w}_i\|_1 \leq \delta\,, \forall\, i = 1,\dots,Q, \tag{12}$$

$$\|\mathbf{w}_i\|_2 \leq \beta\,, \forall\, i = 1,\dots,Q, \tag{13}$$

$$\|\mathcal{X}_{::t}\|_F = \xi\,, \forall\, t = 1,\dots,T, \tag{14}$$

$$\mathcal{X}_{kjt} \leq \mathcal{X}_{kjt+1}\,, \forall\, k = 1,\dots,K,\ j = 1,\dots,N,\ t = 0,\dots,T-1, \tag{15}$$

$$W_{ik} \geq 0\,, \forall\, i = 1,\dots,Q,\ \ K = 1,\dots,K, \tag{16}$$

$$\mathcal{X}_{kj0} \geq 0\,, \forall\, k = 1,\dots,K,\ j = 1,\dots,N. \tag{17}$$

In $(P2)$, constraints (12) and (13) ensure the sparsity of matrix $\mathbf{W}$. Constraints (14) prevent $\mathcal{X}$ from an unbounded growth. Constraints (15) play an important role in increasing the expressive power of PSM, by ensuring the continuity and monotonicity of knowledge accumulation. Indeed, one can interpret the contents of $\mathcal{X}$ as the inferred student learning curves in all the knowledge components; importantly, recall that the elements of $\mathcal{X}$ across students are not necessarily similar.

Note that in general, tensor factorization can be a rather computationally expensive task, depending on the number of dimensions. However, the low rank property, $K \ll Q, N$, renders tensor $\mathcal{X}$ narrow, thus keeping $(P2)$ tractable.

4 Scalable Algorithms for Tensor-based Student Modeling

The conceptual appeal of the tensor factorization-based student models, and in particular their ability to distill learning curves from student performance data, needs to be matched by such algorithmic developments that would enable HSM and PSM use at scale, e.g., in MOOC-supporting Intelligent Tutoring Systems. The optimization problems $(P1)$ and $(P2)$ are multi-convex and non-smooth, complicating the likelihood maximization task [15, 29].

This section presents the technical approaches taken to treat the optimization problems for HSM and PSM. The section begins with the relaxed problem formulations and proceeds by describing two algorithms that are suitable for solving such multi-convex problems.

4.1 Algorithms for SSM and HSM Learning

Distinguished from convex optimization problems, multi-convex problems are global optimization problems which may contain a (large) number of local optima [15]. The constraints of (P1) are in the form of convex sets, and hence, one can use the method of Lagrangian multipliers to convert the original constrained problem into an unconstrained one; more specifically, only the non-negativity constraints remain:

$$(P1') \min_{\mathbf{W},\mathbf{C},\mathbf{V},\mathbf{d},\boldsymbol{\theta}} \Gamma = \sum_{(i,j,t)\in\Omega_{obs}} -log(P(\mathcal{Y}_{ijt}|\mathbf{w_i}, \mathbf{c_j}, \mathbf{v_t}, d_i, \theta_j)) + \lambda_1 \sum_{i=1}^{Q} \|\mathbf{w}_i\|_1 + \frac{\lambda_2}{2} \sum_{i=1}^{Q} \|\mathbf{w}_i\|_2 + \frac{\lambda_3}{2}\|\mathbf{C}\|_F + \frac{\lambda_4}{2}\|\mathbf{V}\|_F \quad (18)$$

$$s.t.:$$
$$W_{ki}, C_{kj}, V_{kt} \geq 0\,,\ \forall\, i = 1,\ldots,Q,\ K = 1,\ldots,K,\ j = 1,\ldots,N,\ t = 1,\ldots,T.$$

The partial gradients of the objective function in $(P1')$ can be expressed as follows:

$$\nabla\Gamma_{W_{ki}} = -\sum_{j=1}^{N}\sum_{t=1}^{T} C_{kj}V_{kt}(\mathcal{Y}_{ijt} - \frac{1}{1+e^{-\mathcal{T}_{ijt}}}) + \frac{\lambda_2 W_{ki}}{\|W\|_F}\,, \tag{19}$$

$$\nabla\Gamma_{C_{kj}} = -\sum_{i=1}^{Q}\sum_{t=1}^{T} W_{ki}V_{kt}(\mathcal{Y}_{ijt} - \frac{1}{1+e^{-\mathcal{T}_{ijt}}}) + \frac{\lambda_3 C_{kj}}{\|C\|_F}\,, \tag{20}$$

$$\nabla\Gamma_{V_{kt}} = -\sum_{j=1}^{N}\sum_{i=1}^{Q} C_{kj}W_{ki}(\mathcal{Y}_{ijt} - \frac{1}{1+e^{-\mathcal{T}_{ijt}}}) + \frac{\lambda_4 V_{ki}}{\|V\|_F}\,, \tag{21}$$

$$\nabla\Gamma_{d_i} = \sum_{j=1}^{N}\sum_{t=1}^{T} (\mathcal{Y}_{ijt} - \frac{1}{1+e^{-\mathcal{T}_{ijt}}})\,, \tag{22}$$

$$\nabla\Gamma_{\theta_j} = -\sum_{i=1}^{Q}\sum_{t=1}^{T} (\mathcal{Y}_{ijt} - \frac{1}{1+e^{-\mathcal{T}_{ijt}}})\,. \tag{23}$$

All the gradients for SSM can be computed per (19-23). For HSM, additional treatment is required for constraints (9). First, these constraints are simplified,

$$\sum_{k=1}^{K} C_{kj}V_{k,t+1} - \sum_{k=1}^{K} C_{kj}V_{kt} = \sum_{k=1}^{K} C_{kj}\Big(V_{k,t+1} - V_{kt}\Big)\,, \forall\, j = 1,\ldots,N,\, t = 1,\ldots,T. \tag{24}$$

Introducing new variables $\Delta V_{k,t+1} = V_{k,t+1} - V_{kt}$ and minding the non-negativity of C_{kj} (already enforced), constraints (24) can now be replaced by $\Delta V_{k,t+1} \geq -\epsilon$.

4.2 PSM Optimization Problem

The parameterization of PSM cannot be performed exactly as in (P_1') due to the difference between PSM and SSM in the form of the employed posterior probability functions. For PSM, one needs to treat the constraints $\mathcal{X}_{kjt} \leq \mathcal{X}_{kjt+1}, \forall\, k = 1,\ldots,K,\ j = 1,\ldots,N,\ t = 0,\ldots,T-1$. Similar to (24), let $\Delta\mathcal{X}_{kjt+1} = \mathcal{X}_{kjt+1} - \mathcal{X}_{kjt}$, and arrive at the optimization problem,

$$(P2')\ \min_{\mathbf{W,C,V}} \Gamma = \sum_{(i,j,t)\in\Omega_{obs}} -log(P(\mathcal{Y}_{ijt}|\mathbf{w_i}, \mathcal{X}_{:jt})) + \lambda_1\sum_{i=1}^{Q}\|\mathbf{w}_i\|_1 + \frac{\lambda_2}{2}\sum_{i=1}^{Q}\|\mathbf{w}_i\|_2 + \frac{\lambda_3}{2}\sum_{t=1}^{T}\|\mathcal{X}_{::t}\|_F$$

$s.t.:$

$$\Delta\mathcal{X}_{kjt+1} \geq 0\,,\ \forall\, k = 1,\ldots,K,\ j = 1,\ldots,N,\ t = 0,\ldots,T-1, \tag{25}$$

$$W_{ik} \geq 0\,,\ \forall\, i = 1,\ldots,Q,\ K = 1,\ldots,K, \tag{26}$$

$$\mathcal{X}_{kj0} \geq 0\,,\ \forall\, k = 1,\ldots,K,\ j = 1,\ldots,N. \tag{27}$$

The gradients of the objective function function in $(P2')$ are obtained similarly to that in (P_1'), however, since the factorization is different, additional work is required to calculate $\nabla\Gamma_W$:

$$\nabla\Gamma_{W_{ki}} = -\sum_{j=1}^{N}\sum_{t=1}^{T} \mathcal{X}_{kjt}(\mathcal{Y}_{ijt} - \frac{1}{1+e^{-\mathcal{T}_{ijt}}}) + \lambda_2\frac{W_{ki}}{\|W\|_F}\,, \tag{28}$$

$$\nabla\Gamma_{\mathcal{X}_{kjt}} = -\sum_{i=1}^{Q} W_{ki}(\mathcal{Y}_{ijt} - \frac{1}{1+e^{-\mathcal{T}_{ijt}}}) + \lambda_3\sum_{t=1}^{T}\frac{\mathcal{X}_{kjt}}{\|\mathcal{X}_{::t}\|_F}\,. \tag{29}$$

One can use (22) and (23) to calculate $\nabla\Gamma_{\mathbf{d}}$ and $\nabla\Gamma_{\boldsymbol{\theta}}$, respectively, exploiting the form of $\mathcal{T}$ expressed in (10).

To solve the optimization problems $(P1')$ and $(P2')$, any full gradient-based algorithm can be applied. However, in our case, the (re)calculation of the full gradient turns out to be very computationally expensive, due to multi-dimensionality. The next section discusses the application of two popular gradient based algorithms that estimate the gradient in part, instead of using its full version. Those method exhibit different levels of scalability and convergence, and their comparison in application to SSM, HSM and PSM deserves special attention.

4.3 Block Coordinate Descent

Block Coordinate Descent (BCD) methods [42] employ the fundamental ideas of Coordinate Descent (CD); CD has been successfully employed in large-scale machine learning problems, to solve Support Vector Machines [17], maximum entropy models [17], and Non-negative Matrix Factorization problems [8], among others. The main idea of CD is to use the gradient to update a single problem variable at each iteration (while keeping others unchanged), and move to the next variable at the next iteration, etc. Most applications utilize variations of CD – BCD methods – where groups (blocks) of variables are updated at each iteration, thus performing search along a coordinate hyperplane rather than a single coordinate direction [42]. Most concepts of single-coordinate descent methods can be generalized to the BCD without any fundamental changes [42].

BCD has been reported to be an efficient, scalable algorithm, particularly effective for solving large-scale convex optimization problems; however, the optimization problems $(P1')$ and $(P2')$ are non-convex and non-smooth due to the ℓ-1 regularization. However, one can adapt BCD to solve $(P1')$ by using projection methods to handle the non-smoothness in the non-negativity constraints. If these constraints are met, then the absolute value-dependent terms in the ℓ-1 norm can be dropped to render the remainder of the objective function differentiable. Another complicating aspect of dealing with $(P1')$ is that ℓ-2 regularization is typically computationally expensive; instead, to facilitate the differentiation, one can adopt the Frobenius norm as an approximation: indeed, $\|\mathbf{A}\|_2 \leq \|\mathbf{A}\|_F$.

The original BCD algorithm cyclically updates each block of variables by exactly solving one sub-problem after another; note, however, that the blocks can be updated in different orders (e.g., in a cyclic, randomized or parallel manner), and also, each sub-problem may not necessarily need to be solved to optimality. In our implementation, the indexes $i \in \{1, \ldots, Q\}$, $j \in \{1, \ldots, N\}$ and $t \in \{1, \ldots, T\}$ are randomly selected, and a block of variables grouped by k is updated in each iteration as

$$\mathbf{w}_i^{new} \leftarrow \max\{\mathbf{0}, \mathbf{w}_i^{old} - \beta \nabla \Gamma_{\mathbf{w}_i}\},$$

where $\mathbf{w}_i$ is a vector of the variables related to question i. To satisfy non-negativity constraints, a projection operation is used. To update $\mathbf{c}_j$, one performs the operation

$$\mathbf{c}_j^{new} \leftarrow \frac{1}{1+\beta\gamma}\Big(\max\{\mathbf{0}, \mathbf{c}_j^{old} - \beta \nabla \Gamma_{\mathbf{c}_j}\}\Big)$$

on the block variable $\mathbf{c}_j$. Here, the projection method is used to preserve non-negativity, and also, $\mathbf{c}_j$ is scaled to prevent an unbounded growth. Similarly, $\mathbf{v}_t$ is updated as

$$\mathbf{v}_t^{new} \leftarrow \frac{1}{1+\beta\gamma}\Big(\max\{\mathbf{0}, \mathbf{v}_t^{old} - \beta \nabla \Gamma_{\mathbf{v}_t}\}\Big).$$

Updating d_i and θ_j is straightforward. Algorithm (1) for SSM and HSM summarizes all the described BCD steps.

Algorithm 1 Block Coordinate Descent Algorithm-SSM

Input: Observed Tensor $\mathcal{Y}$, N, Q, T, K, $\lambda, \mu, \gamma, \beta$.
Output: Completed Tensor $\mathcal{Y}$, Matrices $\mathbf{W}$, $\mathbf{C}$ and $\mathbf{V}$.
BlockCoordinateDescent()

Initialize randomly $\mathbf{W}$, $\mathbf{C}$ and $\mathbf{V}$.
while (stopping criteria) **do**
  randomly select $i \in \{1, \ldots, Q\}$, $j \in \{1, \ldots, N\}$ and $t \in \{1, \ldots, T\}$;
  calculate $\nabla\Gamma_{\mathbf{w}_i}$, $\nabla\Gamma_{\mathbf{c}_j}$, $\nabla\Gamma_{\mathbf{v}_t}$, $\nabla\Gamma_{d_i}$ and $\nabla\Gamma_{\theta_j}$ using (19)-(23)
  update $\mathbf{w}_i^{new} \leftarrow \max\{\mathbf{0}, \mathbf{w}_i^{old} - \beta\nabla\Gamma_{\mathbf{w}_i}\}$
  update $\mathbf{c}_j^{new} \leftarrow \frac{1}{1+\beta\gamma}\Big(\max\{\mathbf{0}, \mathbf{c}_j^{old} - \beta\nabla\Gamma_{\mathbf{c}_j}\}\Big)$
  update $\mathbf{v}_t^{new} \leftarrow \frac{1}{1+\beta\gamma}\Big(\max\{\mathbf{0}, \mathbf{v}_t^{old} - \beta\nabla\Gamma_{\mathbf{v}_t}\}\Big)$
  update $d_i^{new} \leftarrow d_i^{old} - \beta\nabla\Gamma_{d_i}$
  update $\theta_j^{new} \leftarrow \theta_j^{old} - \beta\nabla\Gamma_{\theta_j}$
  calculate objective function using (18)
end
report AIC, AIC_c, BIC, MSE

4.4 Alternating Direction Method of Multipliers

The Alternating Direction Method of Multipliers (ADMM) belongs to the class of BCD optimization methods. The idea of ADMM is to combine the decomposability benefit, characteristic of dual ascent algorithms, with the fast convergence advantage, characteristic of the method of multipliers [4]. In an ADDM algorithm, some variables in each category (i.e., $\mathbf{W}, \mathbf{C}$ and $\mathbf{V}$) corresponding are updated in each subproblem, in an alternating fashion. Ideally, an optimal solution needs to be achieved for each sub-problem before moving on to the next one. Using ADMM typically results in faster convergence than BCD; however, BCD is more straightforward to implement and tends to offer better scalability for large-scale optimization problems. This trade-ff is worth exploring.

An ADMM implementation for SSM is given in Algorithm (2). To satisfy the non-negativity constraints in the proposed optimization problems, one again can rely on projection methods, e.g., the following projection to satisfy (15)),

$$\mathcal{X}_{::t+1}^{new} = \max\Big\{\mathcal{X}_{::t}, \mathcal{X}_{::t+1}^{new}\Big\}.$$

Algorithm 2 Alternating Direction Method of Multipliers-SSM

Input: Observed Tensor $\mathcal{Y}$, N, Q, T, K, $\lambda, \mu, \gamma, \beta$.
Output: Completed Tensor $\mathcal{Y}$, Matrices $\mathbf{W}$, $\mathbf{C}$ and $\mathbf{V}$.
AlternatingDirectionMethodMultipliers()

Initialize randomly $\mathbf{W}$, $\mathbf{C}$ and $\mathbf{V}$.
while (stopping criteria) **do**
  choose step sizes β_W, β_d, β_C, β_θ and β_V
  while (stopping criteria) **do**
    calculate $\nabla\Gamma_{\mathbf{w}}$ and $\Gamma_{\mathbf{d}}$ using (19) and (22)
    update $\mathbf{w}^{new} \leftarrow \max\{\mathbf{0}, \mathbf{w}^{old} - \beta_W\nabla\Gamma_{\mathbf{w}}\}$
    update $\mathbf{d}^{new} \leftarrow \mathbf{d}^{old} - \beta_d\nabla\Gamma_{\mathbf{d}}$
  while (stopping criteria) **do**
    calculate calculate $\nabla\Gamma_{\mathbf{c}}$ and $\nabla\Gamma_{\boldsymbol{\theta}}$ using (20) and (23)
    update $\mathbf{C}^{new} \leftarrow \frac{1}{1+\beta_C\gamma}\Big(\max\{\mathbf{0}, \mathbf{C}^{old} - \beta_C\nabla\Gamma_{\mathbf{C}}\}\Big)$
    update $\boldsymbol{\theta}^{new} \leftarrow \boldsymbol{\theta}^{old} - \beta_\theta\nabla\Gamma_{\boldsymbol{\theta}}$
  while (stopping criteria) **do**
    calculate $\nabla\Gamma_{\mathbf{V}}$ using (21) and (23)
    update $\mathbf{V}^{new} \leftarrow \frac{1}{1+\beta_V\gamma}\Big(\max\{\mathbf{0}, \mathbf{V}_t^{old} - \beta\nabla\Gamma_{\mathbf{V}}\}\Big)$
  calculate objective function using (18)
end
report AIC, AIC_c, BIC, MSE

5 Experimental Setup

This section describes the process of generating synthetic data for an experimental evaluation of the SSM, HSM, and PSM performance. There are several real-world datasets that researchers in the area of educational data mining use to test their algorithms [24,28,37]; however, we require a special experimental setup, where learners progress over time along some learning curves, to see if the presented models and algorithms can indeed recover those learning curves from the learner performance data. In other words, we set to generate the ground truth dataset representing a learning setting where knowledge acquisition patterns are known to exist, in order to make conclusions about the usefulness and inferential power of the developed inference methods.

Synthetic datasets of different sizes are prepared, to create the inference problem instances in three classes: (1) small-sized, (2) medium-sized, and (3) large-sized. Note that for each instance, the corresponding SSM or HSM optimization problem formulation has $(N+Q+T)K+(N+Q)$ variables, while the PSM formulation has $K(NT+Q)+(N+Q)$ variables.

Table 1: Experiments setup for testing Tensor Factorization Models.

Category	Experiment ID	Question	Student	Time (week)	KC	Sparsity	Estimated KC	Iterations	Num of Run
Small	1	100	20	6	3	25%	2-5	500	30
	2	100	20	6	3	50%	2-5	500	30
	3	100	20	6	3	75%	2-5	500	30
	4	100	20	6	3	85%	2-5	500	30
Medium	5	200	40	10	5	15%	2-5	200	25
	6	200	40	10	5	25%	2-5	200	25
	7	3200	40	10	5	50%	2-5	200	25
	8	200	40	10	5	75%	2-5	200	25
Large	9	500	100	15	8	10%	2-5	100	20
	10	500	100	15	8	15%	2-5	100	20
	11	500	100	15	8	25%	2-5	100	20
	12	500	100	15	8	50%	2-5	100	20

5.1 Synthetic Data Generation

In order to produce synthetic data exhibiting student performance improvement resulting from knowledge acquisition, the ideas of BKT, SPARFA and factorial hidden Markov modeling [13] are integrated in a complex simulation model of student learning. Let $\mathcal{X}_{kjt}$ denote student j's knowledge level in knowledge component k at time t, with the range comprised of four values: 0 (corresponding to the state of "no knowledge"), 1 (for weak knowledge), 2 (for shaky knowledge), or 3 (for solid knowledge).

Assuming that problem-solving, i.e., question-taking, as a form of practice can itself affect student knowledge, any student's knowledge improvement also depends of the characteristics of the questions they take. Let the binary indicator variable $\mathcal{U}_{ijt}$ be equal to one if question i is given to student j at time t, and zero otherwise. The Transition Probability Matrix (TPM) contains the probabilities that a student progresses/digresses from any one knowledge state to other state(s),

$$P(\mathcal{X}_{kjt}|\mathcal{X}_{kjt-1},\mathcal{U}_{ijt}) = \begin{bmatrix} \mathbf{1-\rho_{kj}} & \boldsymbol{\rho_{kj}} & \mathbf{0} & \mathbf{0} \\ \boldsymbol{\rho_F} & \mathbf{1-(\rho_F+\rho_{kj})} & \boldsymbol{\rho_{kj}} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\rho_F} & \mathbf{1-(\rho_F+\rho_{kj})} & \boldsymbol{\rho_{kj}} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{\rho_F} & \mathbf{1-\rho_F} \end{bmatrix}, \tag{30}$$

where ρ_F denotes the probability that a student (partially) forgets what they have learned, and ρ_{kj} denotes the probability of a knowledge state transition for student j in knowledge component k.

The transition probability values in (30) are modeled using two components,

$$\rho_{kj} = A_{kj} + \mathcal{B}_{kit} \times \mathcal{U}_{ijt}, \tag{31}$$

where A_{kj} stands for the general (expected) learning rate in knowledge component k for student j, and $\mathcal{B}_{ijt}$ indicates the knowledge gain that student j achieves through interaction with question i, at time t. Note an implicit assumption in this model: $\mathcal{X}_{kjt}$ is independent of $\mathcal{X}_{k'jt}$, $\forall\, k \neq k'$. This is a simplifying assumption; however, one should consider that fact that increasing the number of model parameters would make any resulting learning patterns less discernible and less interpretable. Assuming a Markovian property of the learning process, the transition probability in the full knowledge space is given by

$$P(\mathcal{X}_{:jt}|\mathcal{X}_{:jt-1}, \mathcal{U}_{ijt}) = \prod_{k=1}^{K} P(\mathcal{X}_{kjt}|\mathcal{X}_{kjt-1}, \mathcal{U}_{ijt}). \tag{32}$$

The next step is to model the uncertainty in the students' performance in answering questions correctly/incorrectly, depending on their knowledge states at the times each question is taken, and accounting for question difficulty and student aptitude levels. Modeling each problem-solving outcome by a binary random variable (returning 1 if the answer is correct and 0 otherwise), let $\mathcal{Y}_{ijt} = 1(0)$ denote a realization of such a binary variable – the performance indicator that takes the value of one if student j answered question i at time t correctly, and zero otherwise. Let $\mathcal{Y}_{ijt}$ be modeled as a Bernoulli random variable with parameter $\phi(\mathcal{T}_{ijt})$, where

$$\mathcal{T}_{ijt} = \sum_{k=1}^{K} W_{ik}\mathcal{X}_{kjt} - d_i + \theta_j, \tag{33}$$

with coefficient W_{ik} capturing the relation between the scope of question i and knowledge component k, and $\phi(\cdot)$: $\mathrm{R} \to [0, 1]$ standing for the *logit* function. Figure 3 provides a graphical representation for the resulting Factorial Hidden Markov Model (FHMM) [13]. The FHMM generates observations according to the probabilistic formula

$$P(\mathcal{Y}_{ijt}|\mathbf{w_i}, \mathcal{X}_{:jt}, d_i, \theta_j) = \Phi(\mathcal{T}_{ijt})^{\mathcal{Y}_{ijt}}[1 - \Phi(\mathcal{T}_{ijt})]^{1-\mathcal{Y}_{ijt}}. \tag{34}$$

Algorithm 3 summarizes all the steps involved in generating synthetic data using the FHMM with the components (30-34).

Algorithm 3 Synthetic Data Generator based on Factorial Hidden Markov Model

Input: Tensors $\mathcal{B}, \mathcal{U}$, matrices $\mathbf{W}$, $\mathbf{A}$, vectors $\mathbf{d}$, $\boldsymbol{\theta}$ and parameters N, Q, T, K.
Output: Tensors $\mathcal{Y}$ and $\mathcal{X}$.
SyntheticDataGenerator˙HHM()

for student $j = 1 : N$
  while $(t \leq T)$ **do**
    $\mathcal{U}_{ijt}$, $\mathcal{B}_{ijt} \leftarrow$ propose a question
    for knowledge component $k = 1 : K$
      calculate $P(\mathcal{X}_{kjt}|\mathcal{X}_{kjt-1}, \mathcal{U}_{ijt})$ using (30)
      given $\mathcal{X}_{kjt-1}$, $\mathcal{X}_{kjt} \leftarrow 0, 1, 2$ or 3 with probability $P(\mathcal{X}_{kjt}|\mathcal{X}_{kjt-1}, \mathcal{U}_{ijt})$
      calculate $\mathcal{T}_{ijt} \leftarrow \sum_{k=1}^{K} W_{ik}\mathcal{X}_{kjt} - d_i + \theta_j$
      calculate $P(\mathcal{Y}_{ijt}|\mathbf{w_i}, \mathcal{X}_{:jt}, d_i, \theta_j) \leftarrow \Phi(\mathcal{T}_{ijt})^{\mathcal{Y}_{ijt}}[1 - \Phi(\mathcal{T}_{ijt})]^{1-\mathcal{Y}_{ijt}}$
      choose $\mathcal{Y}_{ijt} = 1$ with $P(\mathcal{Y}_{ijt} = 1|\mathbf{w_i}, \mathcal{X}_{:jt}, d_i, \theta_j)$ and 0 otherwise
    end
    $t \leftarrow t + 1$
  end
end

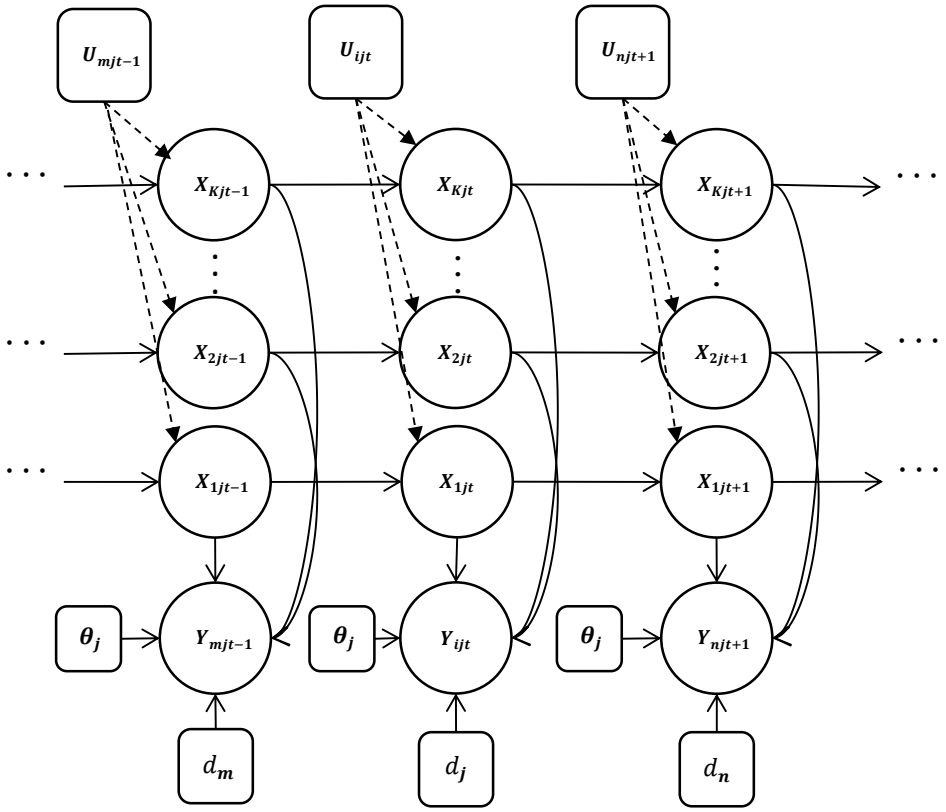

Fig. 3: Graphical Model representing the underlying dynamics of student j learning and the impact of each question on students' performance.

5.2 Implementation

The following notes describe the details related to the implementation of the developed BCD and ADMM algorithms for the parameterization of SSM, HSM and PSM.

Software: We used a custom code written in Python using NumPy, CVXPY and Pandas libraries. Double precision values were used in defining the matrices and tensors. Since the observation matrix $\mathcal{Y}$ is binary and highly sparse, the compressed column and row representations were adopted.

Hardware: Different hardware for various experimental setup was used. For the experiments 1 through 8, we used a desktop with Intel(R) Core(TM)i5 3GHz processor, 8GB RAM and 64 bit operating system. For the experiments 9 through 12, Dell machines with 12x2.40GHz Intel Xeon E5645 Processor Cores (12 cores per node) with 48 GB memory and Linux (RedHat Enterprise Linux 6.1 2.6.32 Kernel) operating systems were used.

Reporting: The main challenge in dealing with multidimensional non-convex optimization problems is a huge number of local optima causing iteration-based algorithms to get stuck without achieving convergence to a global optimum. Another issue with such large-scale optimization problems is that an optimal solution is typically unknown, particularly in real-world applications. Consequently, defining a stopping criterion is problematic, even if one can tweak it trying to tailor the algorithm to make it approach the optimum. In this paper, we use the pre-set number of iterations as the stopping criterion. To ensure that the randomness inherent in the algorithm initialization does not affect the performance evaluation, each experiment was run several times (see Table 1). In addition, the time to find the best solution in each test run was recorded and reported. In order to provide insights about the models' utility and help one to compare the models against each other, such measures such as Akaike Information Criteria (AIC), AIC corrected (AIC_c), Bayesian Information Criteria (BIC), Log likelihood, Frobenius

Norm and Mean Square Error (MSE) were reported. We used the following formulae to calculate the values of the respective metrics:

$$AIC = 2k - 2\Gamma, \quad AIC_c = AIC + \frac{2k(k+1)}{n-k-1}, \quad BIC = -2\Gamma + k\log(n),$$

where k is the number of parameters, Γ denotes the log-likelihood function, and n denotes the sample size.

The next section discusses the observed performance of BCD and ADMM executed on small-, medium- and large-sized problem instances to parameterize SSM, HSM and PSM. The MSE values, used as the main criterion in evaluating the models' performance, are reported for the comparisons of the probabilities of correct answers per the FHMM (used in the synthetic data generation) against the estimates of these probabilities, inferred from the observational data tensors with varied sparsity levels.

6 Computational Results

This section reports the results of the experiments conducted to assess and compare the performance of the SSM, HSM, and PSM.

Recall that in order to preserve both the models' simplicity (ensuring computational efficiency) and ability to inform decision-making (in terms of identifying the best sequences of tasks that students should take on), the presented tensor factorization approach to student modeling is non-parametric with respect to the interaction between students and learning objects. Consequently, the proposed models are not compared versus the models that explicitly compute the knowledge gain that is independent of time. We use the original SPARFA to benchmark the performance of SSM, HSM and PSM.

6.1 Comparison of BCD and ADMM

As discussed in Section 4, although both BCD and ADMM belong to the same family of iterative gradient-based algorithms, the differences between them that make one or the other more suitable for different applications. In many real-time (online) applications, the run time of an executed algorithm is of key importance, i.e., an implemented optimization method is expected to provide a *good-enough* solution in a short amount of time. Figure 4 compares the convergence rate of ADMM versus BCD on small-sized problem instances with a varied number of knowledge components. Because ADMM solves subproblems sub-optimally in all iterations, it has a higher convergence rate in terms of the number of iterations. On the other hand, each of its iterations takes more time.

Judging by the results of the performed experiments, ADMM comes out as an overall winner across all the test instances. Hence, all the experiments in the subsequent sections are performed using ADMM – for the SSM, HSM, and PSM parameterization.

6.2 Prediction Results for Small-sized Problem Instances

The small-sized instance set encompasses the instances with 20 students and 100 questions, which are representative of a setting typical in STEM – e.g., an advanced course taught in a Winter or Summer session. Table 2 summarizes the results across multiple student performance prediction instances, reporting various indexes allowing for a fair comparison of the models' performance. The results show that PSM outperforms SPARFA, SSM and HSM, i.e., achieves higher accuracy in prediction. Specifically, Figure 5 reports the MSE values for the instances with a varied number of knowledge components and 25% sparsity in the observation tensor. PSM exhibits a robust performance, as evidenced by the small variance

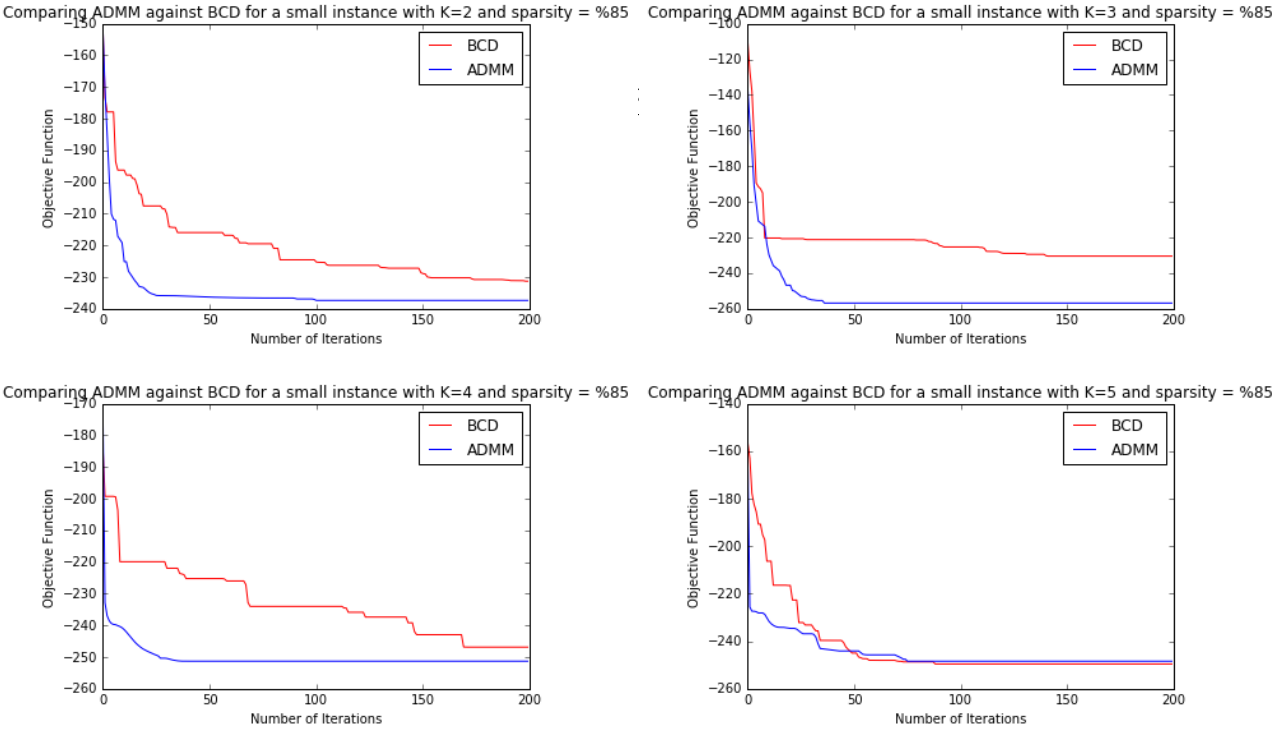

Fig. 4: Convergence rate of both BCD and ADMM algorithms on small-sized with different knowledge components in 200 iterations which shows ADMM converges with less number of iterations.

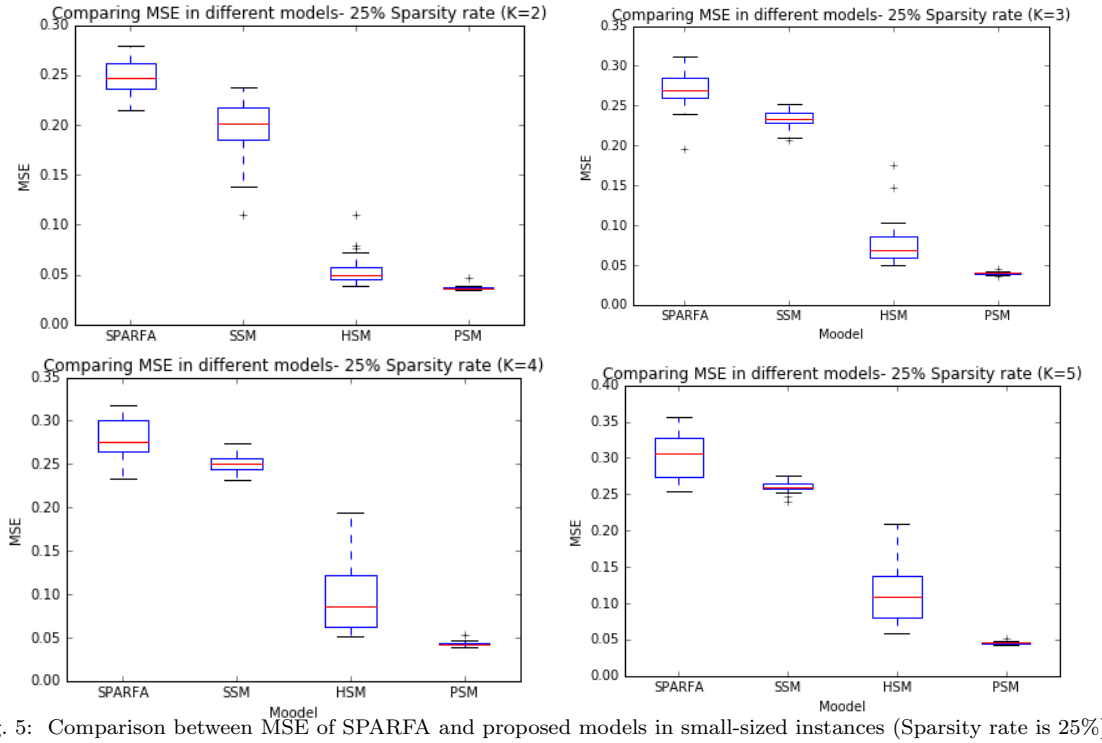

Fig. 5: Comparison between MSE of SPARFA and proposed models in small-sized instances (Sparsity rate is 25%) .

of MSE shown in the box plot. In the small-sized instances with $K = 2$ and $K = 3$, the performance of HSM and PSM are close; however, the averages of MSE are statistically significantly different.

6.3 Prediction Results for Medium-sized Problem Instances

The medium-sized test instances feature 200 questions and 40 students each, which is typical of a medium size STEM class taught in a Fall and Spring semester. Figure 6 depicts MSE values for the test datasets with sparsity rate 75% and a varied number of knowledge components. Again, PSM exhibits a superior performance compared to SSM and HSM. The MSE variance for PSM is also very small. The average

Table 2: Simulation Results of the small size problem using three models, different sparsity rates and knowledge component (hidden variables).

Model	Sparsity	KC	AIC	AIC˙c	BIC	Time(s)	Likelihood	Fro. Norm	MSE
SSM	25%	2	8679.249	8690.104	10542.2	145.895	-4087.625	48.268	0.195
		3	8718.492	8743.147	11512.918	75.858	-3981.246	52.83	0.233
		4	9660.322	9799.774	16165.865	47.271	-3950.161	53.615	0.24
		5	10147.78	10370.02	18279.708	69.988	-3973.89	54.718	0.25
	50%	2	5283.648	5317.222	8536.419	17.991	-2201.824	50.704	0.214
		3	5660.451	5737.399	10539.608	31.581	-2170.225	54.208	0.245
		4	6179.3	6318.751	12684.842	43.009	-2209.65	53.581	0.24
		5	6632.397	6854.638	14764.325	39.694	-2216.198	54.157	0.245
	75%	2	3110.927	3144.501	6363.698	2.274	-1115.463	50.989	0.217
		3	3520.042	3596.99	8399.199	28.566	-1100.021	54.272	0.246
		4	3923.637	4063.088	10429.179	18.181	-1081.818	53.95	0.243
		5	4411.602	4633.842	12543.53	33.268	-1105.801	54.722	0.25
	85%	2	2303.694	2337.268	5556.465	2.82	-711.847	50.269	0.211
		3	2641.75	2718.699	7520.907	21.594	-660.875	54.393	0.247
		4	3101.327	3240.778	9606.87	17.535	-670.664	53.736	0.241
		5	3530.748	3752.988	11662.676	4.944	-665.374	53.468	0.238
HSM	25%	2	9276.795	9287.65	11139.746	3.015	-4386.398	25.284	0.054
		3	9412.838	9437.493	12207.264	4.767	-4328.419	30.284	0.079
		4	9592.628	9636.911	13318.529	5.776	-4292.314	33.02	0.094
		5	9753.432	9823.365	14410.81	5.626	-4246.716	35.942	0.111
	50%	2	5349.594	5360.449	7212.545	5.585	-2422.797	29.762	0.075
		3	5438.839	5463.494	8233.265	2.761	-2341.419	34.475	0.101
		4	5904.906	5949.189	9630.807	50.791	-2448.453	24.579	0.05
		5	5901.79	5971.722	10559.167	8.648	-2320.895	40.015	0.136
	75%	2	2928.458	2939.313	4791.409	15.013	-1212.229	35.766	0.109
		3	3120.603	3145.259	5915.029	10.434	-1182.302	35.608	0.109
		4	3308.058	3352.341	7033.959	4.797	-1150.029	43.453	0.16
		5	3548.754	3618.687	8206.131	5.608	-1144.377	47.822	0.192
	85%	2	2067.214	2078.069	3930.165	6.323	-781.607	30.376	0.078
		3	2204.303	2228.959	4998.73	7.179	-724.152	34.175	0.101
		4	2454.114	2498.397	6180.015	4.366	-723.057	40.693	0.14
		5	2698.58	2768.512	7355.957	14.807	-719.29	48.016	0.198
PSM	25%	2	9356.938	9367.793	11219.889	86.625	-4426.469	20.999	0.037
		3	9610.989	9635.644	12405.415	67.407	-4427.494	21.912	0.04
		4	9862.182	9906.466	13588.084	106.764	-4427.091	22.709	0.043
		5	10130.819	10200.751	14788.196	165.745	-4435.41	23.435	0.046
	50%	2	5423.435	5434.29	7286.386	30.186	-2459.718	22.584	0.043
		3	5617.815	5642.471	8412.241	56.479	-2430.907	23.383	0.046
		4	5904.906	5949.189	9630.807	50.791	-2448.453	24.579	0.05
		5	6152.351	6222.283	10809.728	60.314	-2446.175	25.552	0.054
	75%	2	2992.996	3003.851	4855.947	17.98	-1244.498	23.622	0.047
		3	3212.147	3236.803	6006.573	19.052	-1228.073	23.201	0.045
		4	3479.487	3523.771	7205.389	23.221	-1235.744	25.516	0.054
		5	3738.483	3808.416	8395.86	27.574	-1239.242	27.551	0.063
	85%	2	2081.838	2092.693	3944.789	16.051	-788.919	25.87	0.056
		3	2239.775	2264.431	5034.201	24.134	-741.887	27.074	0.061
		4	2517.483	2561.766	6243.384	23.929	-754.741	28.813	0.069
		5	2777.209	2847.141	7434.586	31.47	-758.605	30.419	0.077

performance of HSM is also acceptable, however, the variance is high. Table 3 summarizes the results of the runs with all the three models over all the medium-sized instances. The time average to obtain a good-enough solution is reasonable for each model; all the three models can be quickly re-parameterized after new student performance observations are collected and added to the problem data.

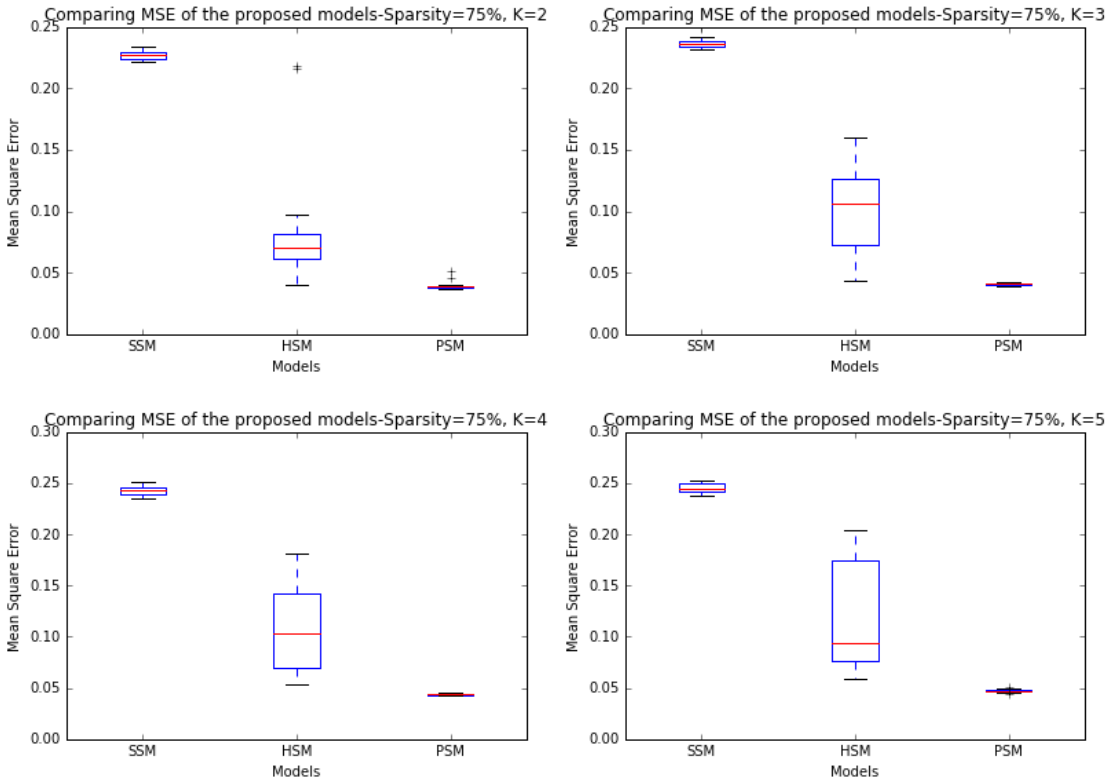

Fig. 6: Comparison between MSE of proposed models in medium-sized problem (Sparsity rate is 75%) .

6.4 Prediction Results for Large-sized Problem Instances

The large-sized instances encompass 500 questions and 100 students each. Table 4 summarizes the results of the corresponding experiments. PSM comes out on top in a majority of the experiments, with varied sparsity and number of knowledge components compared to SPARFA, SSHM and HSM (Fig 7).

7 Conclusion

This paper presents three tensor factorization-based student models that advance the state-of-the-art in Latent Factor Analysis. The proposed models extend the methods used for solving the student performance prediction problem, by framing it as a probabilistic sparse tensor factorization problem.

The key challenge of modeling and predicting students' performance lies in the estimation of their states of knowledge, while recognizing both the temporal dynamics of knowledge acquisition and the differences in individual learning abilities. Responding to this challenge, the proposed tensor factorization models, PSM and HSM, add constraints to the likelihood maximization problem for model parameterization, which allows for distilling knowledge accumulation patterns over time. The top-performing PSM features well interpretable parameters, and detects the differences in the learning curves characteristic of different students.

The Static Student Model – a basic model based on canonical tensor factorization – is employed as the baseline in evaluating the performance of Homogeneous and Personalized Student Models. HSM is an improvement over SSM in that it explicitly assumes the positive overall dynamics of knowledge acquisition; however, HSM does not distinguish between different students, i.e., it assumes the same learning trends for all students.

Personalized Student Model adopts a new constrained probabilistic tensor factorization approach that benefits from the shape of the matrices and the assumption that $K \ll Q, N, T$. PSM personalizes

Table 3: Simulation Results of the Medium-sized problem using three models, different sparsity rates and knowledge component (hidden variables).

Model	Sparsity	KC	AIC	AIC˙c	BIC	Time(s)	Likelihood	Fro. Norm	MSE
SSM	15%	2	51720.36	51726.662	56365.251	41.861	-25360.18	123.495	0.191
		3	51481.741	51495.955	58449.077	49.633	-24990.87	129.307	0.211
		5	51876.305	51916.02	63488.532	65.098	-24688.152	129.248	0.209
	25%	2	46338.534	46344.836	50983.425	59.443	-22669.267	115.235	0.167
		3	47008.48	47022.695	53975.817	72.995	-22754.24	114.693	0.165
	50%	2	32200.593	32206.895	36845.484	9.759	-15600.29	128.825	0.209
		3	34394.879	34477.79	51116.486	840.592	-15397.439	108.719	0.148
		4	34944.438	35092.956	57239.915	1156.122	-15072.219	122.462	0.188
		5	35223.168	35457.015	63092.514	1122.136	-14611.584	131.707	0.217
	75%	2	16843.459	16880.038	27991.197	10.188	-7221.729	134.812	0.227
		3	17838.036	17920.947	34559.643	8.001	-7119.018	137.612	0.237
		4	18934.271	19082.788	41229.747	9.534	-7067.135	139.363	0.243
		5	20241.866	20475.713	48111.212	23.348	-7120.933	140.022	0.245
HSM	15%	2	53686.504	53692.806	58331.395	13.148	-26343.251	111.741	0.158
		3	53179.076	53193.291	60146.412	17.737	-25839.537	126.765	0.203
		5	53693.702	53733.417	65305.929	34.406	-25596.851	122.182	0.187
	25%	2	46899.026	46905.328	51543.917	4.05	-22949.512	130.435	0.213
		3	47851.718	47865.933	54819.054	21.942	-23160.033	109.87	0.152
	50%	2	33805.685	33842.264	44953.423	624.673	-15702.84	94.0	0.110
		3	32672.503	32686.717	39639.839	18.991	-15586.251	101.231	0.129
		4	32838.284	32863.626	42128.066	25.508	-15419.14	114.632	0.168
		5	33088.197	33127.912	44700.425	37.338	-15294.099	111.113	0.157
	75%	2	17087.543	17093.845	21732.434	15.583	-8043.771	80.178	0.086
		3	17328.486	17342.701	24295.823	22.045	-7914.243	89.804	0.104
		4	17614.223	17639.565	26904.005	34.818	-7807.111	92.392	0.111
		5	18095.392	18135.107	29707.619	53.063	-7797.696	94.517	0.117
PSM	15%	2	58163.54	58200.12	69311.279	328.617	-27881.77	93.518	0.11
		3	546752.793	546849.584	615919.764	1007.521	-267376.397	394.482	0.208
		5	525277.366	525547.664	640555.651	280.498	-252638.683	430.52	0.247
	25%	2	50399.653	50436.232	61547.391	397.934	-23999.826	100.289	0.1267
		3	51811.578	51894.49	68533.185	704.491	-24105.789	101.814	0.13
	50%	2	32888.025	32894.327	37532.916	85.483	-15944.013	76.411	0.075
		3	33439.091	33453.306	40406.427	112.309	-15969.545	78.087	0.077
		4	33912.091	33937.433	43201.873	151.735	-15956.046	81.622	0.0864
		5	34254.91	34294.625	45867.138	146.107	-15877.455	81.856	0.087
	75%	2	17326.508	17332.81	21971.399	145.163	-8163.254	56.334	0.04
		3	17911.154	17925.368	24878.49	211.327	-8205.577	57.001	0.0417
		4	18232.683	18258.025	27522.465	314.742	-8116.342	59.05	0.044
		5	18727.232	18766.946	30339.459	420.525	-8113.616	61.324	0.047

conceptual learning trajectory for each student, and also acknowledges that students may learn different content at different rates. The corresponding optimization problem is attacked using the Alternating Direction Method of Multipliers which shows satisfactory performance in comparison to Block Coordinate Descent, particularly when the quality of obtained solutions matters. The simulation results illustrate that PSM outperforms SSM and HSM, and hence, appears a better choice for the use in intelligent tutoring. This model can be employed in the efforts to find optimal teaching policies and monitor student progress in learning.

While already proving itself highly capable of accurately predicting students' performance, PSM could be further improved to enable instructors to measure/track knowledge gains that students experience. Additionally, using a Bayesian approach based on Markov chain Monte Carlo sampling method, one can attempt to improve the presented models in terms of handling uncertainty. Although the point estimation method of the parameters of interest is computationally less expensive, a fully Bayesian approach would enable one to compute full posterior distributions of $\mathbf{C}, \mathbf{W}, \mathbf{V}, \mathcal{X}$, and thereby, compensate the increased computational complexity with accuracy improvements benefits, often sought-after in the context of learning and content analytics [27].

In addition, considering full posterior distributions could enable the computation of informative quantities such as credible intervals and posterior modes for all the model parameters. Moreover, knowing

Table 4: Simulation Results of the large-sized problem using three models, different sparsity rates and knowledge component (hidden variables).

Model	Sparsity	KC	AIC	AIC˙c	BIC	Time(s)	Likelihood	Fro. Norm	MSE
SSM	10%	2	514175.774	514179.819	528355.004	229.606	-255857.887	407.214	0.221
		3	511094.188	511103.292	532363.031	310.869	-253702.094	415.166	0.23
		4	4494268.81	494285.008	522627.268	308.655	-244674.405	436.418	0.254
		5	496029.117	496054.444	531477.189	351.132	-244939.558	432.949	0.25
	15%	2	484116.441	484120.485	498295.67	216.091	-240828.22	410.84	0.225
		3	488403.692	488412.796	509672.535	283.959	-242356.846	406.34	0.22
		4	481374.514	481390.711	509732.972	420.323	-238227.257	417.623	0.233
		5	474968.952	474994.279	510417.025	363.872	-234409.476	425.338	0.241
	25%	2	429330.372	429334.416	443509.601	192.793	-213435.186	404.444	0.218
		3	432443.819	432452.924	453712.663	253.448	-214376.91	401.761	0.215
		4	3408442.98	408459.177	436801.438	575.587	-201761.49	441.391	0.26
		5	407819.662	407844.989	443267.735	276.233	-200834.831	444.862	0.264
	50%	2	276894.218	276898.262	291073.447	80.779	-137217.109	413.082	0.228
		3	278877.231	278886.335	300146.074	115.8	-137593.615	412.201	0.227
		4	273707.644	273723.842	302066.102	256.938	-134393.822	444.707	0.264
		5	277969.74	277995.067	313417.813	295.091	-135909.87	434.777	0.253
HSM	10%	2	485733.714	485737.758	499912.943	60.816	-241636.857	423.958	0.24
		3	493836.421	493845.526	515105.264	86.101	-245073.21	427.841	0.244
		4	490247.312	490263.51	518605.77	94.76	-242663.656	436.371	0.254
		5	499815.053	499840.38	535263.126	97.409	-246832.527	433.823	0.251
	15%	2	459147.227	459151.272	473326.456	57.034	-228343.614	420.423	0.236
		3	465612.334	465621.438	486881.177	53.634	-230961.167	428.786	0.245
		4	469046.48	469062.678	497404.939	77.333	-232063.24	431.188	0.248
		5	470529.395	470554.722	505977.468	107.274	-232189.698	434.033	0.251
	25%	2	407597.46	407601.504	421776.689	57.18	-202568.73	423.297	0.239
		3	410613.839	410622.944	431882.683	56.184	-203461.92	428.311	0.245
		4	415284.658	415300.855	443643.116	68.837	-205182.329	429.131	0.246
		5	416731.35	416756.677	452179.423	56.92	-205290.675	435.497	0.253
	50%	2	279195.334	279199.378	293374.563	45.579	-138367.667	405.957	0.22
		3	279875.905	279885.01	301144.749	45.506	-138092.952	417.733	0.233
		4	280694.386	280710.583	309052.844	71.204	-137887.193	423.118	0.239
		5	283824.332	283849.659	319272.405	66.848	-138837.166	426.501	0.243
PSM	10%	2	567471.183	567514.089	613582.497	366.549	-279735.592	367.019	0.181
		3	546752.793	546849.584	615919.764	1007.521	-267376.397	394.482	0.208
		4	9505109.544	505282.073	597332.172	456.046	-244554.772	414.803	0.23
		5	525277.366	525547.664	640555.651	280.498	-252638.683	430.52	0.247
	15%	2	536414.318	536457.224	582525.632	808.557	-264207.159	364.037	0.177
		3	517226.589	517323.379	586393.56	818.516	-252613.294	393.279	0.207
		4	554256.855	554429.384	646479.483	915.612	-269128.428	391.52	0.205
		5	494151.76	494422.057	609430.045	207.825	-237075.88	435.152	0.253
	25%	2	466909.193	466952.099	513020.507	677.334	-229454.596	373.707	0.186
		3	450906.087	451002.878	520073.058	845.518	-219453.044	404.292	0.219
		4	447710.068	447882.597	539932.696	724.47	-215855.034	418.604	0.234
		5	436880.48	437150.778	552158.765	263.332	-208440.24	436.768	0.254
	50%	2	318848.362	318891.268	364959.676	1054.187	-155424.181	350.12	0.164
		3	318741.796	318838.587	387908.767	1036.587	-153370.898	356.1	0.17
		4	327489.436	327661.965	419712.064	2090.985	-155744.718	346.809	0.161
		5	329442.657	329712.954	444720.941	2395.198	-154721.328	354.312	0.168

the fact that MCMC methods search the full posterior space, they could potentially escape local minima. Finally, the hyperparameters used in Bayesian approach generally allow for more straightforward interpretation of the inferred model parameter values.

References

1. d Baker, R.S., Corbett, A.T., Aleven, V.: More accurate student modeling through contextual estimation of slip and guess probabilities in bayesian knowledge tracing. In: International Conference on Intelligent Tutoring Systems, pp. 406–415. Springer (2008)
2. Bekele, R., Menzel, W.: A bayesian approach to predict performance of a student (bapps): A case with ethiopian students. algorithms **22**(23), 24 (2005)
3. Bell, R.M., Koren, Y.: Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In: Seventh IEEE International Conference on Data Mining (ICDM 2007), pp. 43–52. IEEE (2007)

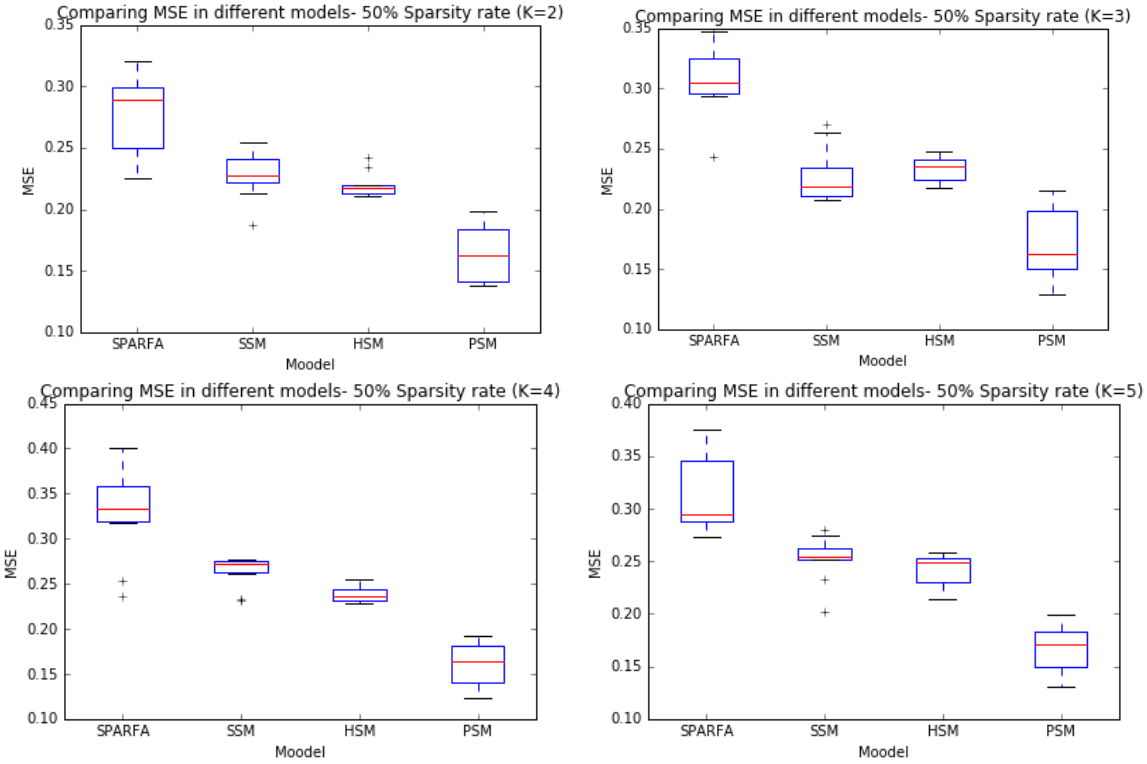

Fig. 7: Comparison between MSE of SPARFA and proposed models in large-sized instances (Sparsity rate is 50%) .

4. Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J.: Distributed optimization and statistical learning via the alternating direction method of multipliers. Foundations and Trends® in Machine Learning **3**(1), 1–122 (2011)
5. Brinton, C.G., Chiang, M.: Mooc performance prediction via clickstream data and social learning networks. In: 2015 IEEE Conference on Computer Communications (INFOCOM), pp. 2299–2307. IEEE (2015)
6. Cen, H., Koedinger, K., Junker, B.: Learning factors analysis–a general method for cognitive model evaluation and improvement. In: International Conference on Intelligent Tutoring Systems, pp. 164–175. Springer (2006)
7. Cetintas, S., Si, L., Xin, Y.P.P., Hord, C.: Automatic detection of off-task behaviors in intelligent tutoring systems with machine learning techniques. IEEE Transactions on Learning Technologies **3**(3), 228–236 (2010)
8. Cichocki, A., Anh-Huy, P.: Fast local algorithms for large scale nonnegative matrix and tensor factorizations. IEICE transactions on fundamentals of electronics, communications and computer sciences **92**(3), 708–721 (2009)
9. Comon, P.: Tensors: a brief introduction. IEEE Signal Processing Magazine **31**(3), 44–53 (2014)
10. Evgeniou, A., Pontil, M.: Multi-task feature learning. Advances in neural information processing systems **19**, 41 (2007)
11. Feng, M., Heffernan, N., Koedinger, K.: Addressing the assessment challenge with an online system that tutors as it assesses. User Modeling and User-Adapted Interaction **19**(3), 243–266 (2009)
12. Frolov, E., Oseledets, I.: Tensor methods and recommender systems. arXiv preprint arXiv:1603.06038 (2016)
13. Ghahramani, Z.: An introduction to hidden markov models and bayesian networks. International Journal of Pattern Recognition and Artificial Intelligence **15**(01), 9–42 (2001)
14. González-Brenes, J., Huang, Y., Brusilovsky, P.: General features in knowledge tracing to model multiple subskills, temporal item response theory, and expert knowledge. In: The 7th International Conference on Educational Data Mining, pp. 84–91. University of Pittsburgh (2014)
15. Gorski, J., Pfeuffer, F., Klamroth, K.: Biconvex sets and optimization with biconvex functions: a survey and extensions. Mathematical Methods of Operations Research **66**(3), 373–407 (2007)
16. Grasedyck, L., Kressner, D., Tobler, C.: A literature survey of low-rank tensor approximation techniques. GAMM-Mitteilungen **36**(1), 53–78 (2013)
17. Hsieh, C.J., Chang, K.W., Lin, C.J., Keerthi, S.S., Sundararajan, S.: A dual coordinate descent method for large-scale linear svm. In: Proceedings of the 25th international conference on Machine learning, pp. 408–415. ACM (2008)
18. Khajah, M.M., Huang, Y., González-Brenes, J.P., Mozer, M.C., Brusilovsky, P.: Integrating knowledge tracing and item response theory: A tale of two frameworks. Personalization Approaches in Learning Environments p. 7 (2014)
19. Komarek, P., Moore, A.W.: Making logistic regression a core data mining tool with tr-irls. In: Fifth IEEE International Conference on Data Mining (ICDM'05), pp. 4–pp. IEEE (2005)
20. Koren, Y.: Factor in the neighbors: Scalable and accurate collaborative filtering. ACM Transactions on Knowledge Discovery from Data (TKDD) **4**(1), 1 (2010)

21. Koren, Y., Bell, R., Volinsky, C., et al.: Matrix factorization techniques for recommender systems. Computer **42**(8), 30–37 (2009)
22. Kotsiantis, S.B., Pintelas, P.E.: Predicting students marks in hellenic open university. In: Fifth IEEE International Conference on Advanced Learning Technologies (ICALT'05), pp. 664–668. IEEE (2005)
23. Kurucz, M., Benczúr, A.A., Csalogány, K.: Methods for large scale svd with missing values. In: Proceedings of KDD cup and workshop, vol. 12, pp. 31–38. Citeseer (2007)
24. Lan, A.S., Studer, C., Baraniuk, R.G.: Quantized matrix completion for personalized learning. arXiv preprint arXiv:1412.5968 (2014)
25. Lan, A.S., Studer, C., Baraniuk, R.G.: Time-varying learning and content analytics via sparse factor analysis. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 452–461. ACM (2014)
26. Lan, A.S., Waters, A.E., Studer, C., Baraniuk, R.G.: Sparse factor analysis for learning and content analytics. arXiv preprint arXiv:1303.5685 (2013)
27. Lan, A.S., Waters, A.E., Studer, C., Baraniuk, R.G.: Sparse factor analysis for learning and content analytics. The Journal of Machine Learning Research **15**(1), 1959–2008 (2014)
28. Lindsey, R.V., Khajah, M., Mozer, M.C.: Automatic discovery of cognitive skills to improve the prediction of student learning. In: Advances in neural information processing systems, pp. 1386–1394 (2014)
29. Liu, J., Musialski, P., Wonka, P., Ye, J.: Tensor completion for estimating missing values in visual data. IEEE Transactions on Pattern Analysis and Machine Intelligence **35**(1), 208–220 (2013)
30. Minaei-Bidgoli, B., Kashy, D.A., Kortemeyer, G., Punch, W.F.: Predicting student performance: an application of data mining methods with an educational web-based system. In: Frontiers in education, 2003. FIE 2003 33rd annual, vol. 1, pp. T2A–13. IEEE (2003)
31. Paterek, A.: Improving regularized singular value decomposition for collaborative filtering. In: Proceedings of KDD cup and workshop, vol. 2007, pp. 5–8 (2007)
32. Pavlik Jr, P.I., Cen, H., Koedinger, K.R.: Performance factors analysis–a new alternative to knowledge tracing. Online Submission (2009)
33. Piech, C., Bassen, J., Huang, J., Ganguli, S., Sahami, M., Guibas, L.J., Sohl-Dickstein, J.: Deep knowledge tracing. In: Advances in Neural Information Processing Systems, pp. 505–513 (2015)
34. Rafailidis, D., Daras, P.: The tfc model: Tensor factorization and tag clustering for item recommendation in social tagging systems. IEEE Transactions on Systems, Man, and Cybernetics: Systems **43**(3), 673–688 (2013)
35. Reckase, M.: Multidimensional item response theory, vol. 150. Springer (2009)
36. Soni, A., Jain, S., Haupt, J., Gonella, S.: Noisy matrix completion under sparse factor models. IEEE Transactions on Information Theory **62**(6), 3636–3661 (2016)
37. Thai-Nghe, N.: Predicting student performance in an intelligent tutoring system. Department of Computer Science, University of Hildesheim, Hildesheim, Germany (2011)
38. Thai-Nghe, N., Busche, A., Schmidt-Thieme, L.: Improving academic performance prediction by dealing with class imbalance. In: 2009 Ninth International Conference on Intelligent Systems Design and Applications, pp. 878–883. IEEE (2009)
39. Thai-Nghe, N., Drumond, L., Horváth, T., Krohn-Grimberghe, A., Nanopoulos, A., Schmidt-Thieme, L.: Factorization techniques for predicting student performance. Educational Recommender Systems and Technologies: Practices and Challenges pp. 129–153 (2011)
40. Thai-Nghe, N., Drumond, L., Krohn-Grimberghe, A., Schmidt-Thieme, L.: Recommender system for predicting student performance. Procedia Computer Science **1**(2), 2811–2819 (2010)
41. Toscher, A., Jahrer, M.: Collaborative filtering applied to educational data mining. KDD cup (2010)
42. Wright, S.J.: Coordinate descent algorithms. Mathematical Programming **151**(1), 3–34 (2015)
43. Yu, H.F., Lo, H.Y., Hsieh, H.P., Lou, J.K., McKenzie, T.G., Chou, J.W., Chung, P.H., Ho, C.H., Chang, C.F., Wei, Y.H., et al.: Feature engineering and classifier ensemble for kdd cup 2010. In: Proceedings of the KDD Cup 2010 Workshop, pp. 1–16 (2010)
44. Yudelson, M.V., Koedinger, K.R., Gordon, G.J.: Individualized bayesian knowledge tracing models. In: Artificial Intelligence in Education, pp. 171–180. Springer (2013)